GraphSearch: Visualized Similarity Search in Heterogeneous Information Networks

Vu N. Nguyen, Xiao Xin Li
{vnnguye2, xli147}@illinois.edu
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA

ABSTRACT

We build an DBLP author search engine GraphSearch for this project based on the similarity measure PathSim by Sun et al. GraphSearch operates on the heterogeneous information network, logical network with multi-typed objects and multi-typed links, generated from DBLP's author, paper, term, and venue data. GraphSearch's offerings are three-fold: fast query, peer similarity search with semantics, and a network visualized interactive interface.

1. INTRODUCTION

Similarity search is one of the fundamental areas of social network analysis and search engine. It has become increasingly important with the advent of large-scale heterogeneous information networks such as social networks (Facebook) and bibliographic network (DBLP). Oftentimes, two objects are considered similar if they are linked by many paths in the network. However, these existing similarity measures only work for homogeneous information networks, which have only one node type and consequently one link type. This type of network, while simple to construct and store, is not expressive enough to capture all the subtleties for real information networks.

Heterogeneous information networks are ones which have multi-typed nodes and multi-typed links, denoting different relations. For example, consider the DBLP Computer Science Bibliography database. We can form a heterogeneous information network from this database with (at least) four node types: author, paper, term, and venue. Connecting these nodes together are different link types. An author node is connected to a paper node by the link 'writes' or, in the opposite direction, 'is written by'. A paper node, on the other hand, is connected to a term node by the link ‘contains’ or, in the opposite direction, ‘is contained in’. This network fully captures the meaning of DBLP database.

Research, in the modern time, almost always requires collaboration. A researcher might want to look for a potential collaborator who is most similar to her. She might want to look for researchers who share her interested topics, or she might want to look for researchers who have collaborated with her collaborators. The concept of meta path allows us to define similarity measure that carries these semantics. A meta path is a path consisting of a sequence of relations defined between different node types (i.e., structural paths at the meta level). For example, the path A-P-A means authors who write papers together, while A-P-T-P-A means authors who write papers containing similar terms.

With that in mind, we aim to build a search engine on top of the DBLP database that utilizes the strengths of heterogeneous information networks and meta paths. It should offer users a way to search for potential collaborators according to her chosen semantics. It should also be user-friendly and capable of displaying a visualization of the connections.

Meta path	Definition
A-P-A	Coauthorship
A-P-A-P-A	Authors who coauthor with similar authors
A-P-T-P-A	Authors who write similar topics
A-P-V-P-A	Authors who publish at similar venues

Table 1: Examples of meta path on DBLP-derived heterogeneous information network

2. RELATED WORKS

There have been many similarity measures developed for similarity search in information networks. Two of the most well-known measures are random walk used in Personalized-PageRank, pairwise random walk used in SimRank. One can easily extract a homogeneous information network from the DBLP database and apply these measures directly. She can also choose to work in the heterogeneous information network setting and meta path framework with these measure. However, it is well received that these measure are biased to either highly visible nodes (i.e., nodes associated with a large number of paths) or highly concentrated objects (i.e., nodes with a large percentage of paths going to a small set of nodes) [REF].

Sun et al. introduce PathSim, similarity measure that utilizes the heterogeneous information network and meta path framework in their 2011 paper “PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks”. PathSim is able to capture the subtle semantics of similarity among peer nodes in a network. In particular, given a query object, PathSim can identify nodes that not only are strongly connected but also share similar visibility in the network given the meta path.

Table 2 and 3 below demonstrate this point by comparing PathSim results to existing similar measures’ with an example. We are given a small heterogeneous information network with five authors and four venues. We want to rank authors who are most similar to Mike using meta path A-V-A. PathSim correctly identifies Bob as the most similar author, while existing measures (P-PageRank, SimRank, RandomWalk, PairwiseRandomWalk) incorrectly (and counter-intuitively) return Jim as the most similar author.

	SIGMOD	VLDB	ICDE	KDD
Mike	2	1	0	0
Jim	50	20	0	0
Mary	2	0	1	0
Bob	2	1	0	0
Ann	0	0	1	1

Table 2: Adjacency matrix W_{AV}

	Jim	Mary	Bob	Ann
P-PageRank	**0.3761**	0.0133	**0.0162**	0.0046
SimRank	**0.7156**	0.5724	**0.7125**	0.1844
RW	**0.8983**	0.0238	**0.0390**	0
PRW	**0.5714**	0.4444	**0.5556**	0
PathSim	0.0826	**0.8000**	**1**	0

Table 3: Similarity between Mike and other authors

3. PROBLEM DEFINITION

We aim to design and build a web-based single page application that is capable of accepting queries (author name) and returns a network which visualizes the original author and his top-10 similar authors as author nodes, and their shared connectivity as nodes and links.

The application should be able to allow user to upload custom data files and store them in a database for quick retrieval. To better assist users to enter author name, the application should support auto text completion after user start typing author name as input. It should let the user specify her choice of meta path. Given the input, it should be able to, within reasonable time limit, calculate the similarity score for all authors that are in the database and return the highest ten such authors. The application should then find the shared items for each and every pair of authors. It finally should display the network with proper nodes, edges information and coloring.

The main challenge of the application is to perform the calculation quickly and accurately. Besides, the interface should be friendly and minimalistic to improve user experience. This web application is architected modularly so that it is easily extendable and maintained.

4. METHOD

4.1 Dataset

We use DBLP computer science bibliography dataset, which contains all bibliographic records. The dataset was downloaded from the DBLP website as text file. We converted them to four types of files in appropriate format: definitions, index, relations, and matrix. Definition data files, including author.json, terms.json, paper.json and venue.json, contain id and name. Index data files, including authorIndex.json, termIndex.json, paperIndex.json, and venueIndex.json, contain an array with id in the index position. We also have 6 relation data files: author2paper.json, paper2author.json, paper2term.json, paper2venue.json, term2paper.json, venue2paper.json. They map a definition ID to an array of related definition IDs. Finally, we have three large text files which contain the matrix data: APA.txt, APTPA.txt, and APVPA.txt.

4.2 Build Web App Using MEAN Stack
JavaScript has exploded in recent years and has moved from the front-end on the server-side and database. MEAN is an acronym made up of commonly used technologies for an all JavaScript web stack. The key components are MongoDB as database, ExpressJS as web framework, AngularJS as front-end framework, and NodeJS as server platform. MEAN stack helps creating a full JavaScript web application in a shorter development time. MongoDB provides the benefits of flexible schema and allow us to run MapReduce job to efficiently calculate similarity scores.

5. PROCESS

5.1 mean.io
We use mean.io as our MEAN stack framework. It is an opinionated fullstack javascript framework, which simplifies and accelerates web application development. It contains Mongodb, Angular.js, Express.js, and Node.js. It allows us to quickly start our project by providing a seed project as a starting point.

5.2 Design MongoDB Schema
There are 4 types of object: definition, index, relation, and matrix. There are 4 definition models: author, paper, term, and venue, with id and name in the schema. There are 4 index models: authorIndex, paperIndex, termIndex, and venueIndex, with id and name in the schema. We also maintain the relations from paper to other objects and from other objects to paper. There are 3 types of matrix: apa, aptpa, and apvpa, with row, column, and value in the schema. All these data are loaded from the data files to mentioned above.

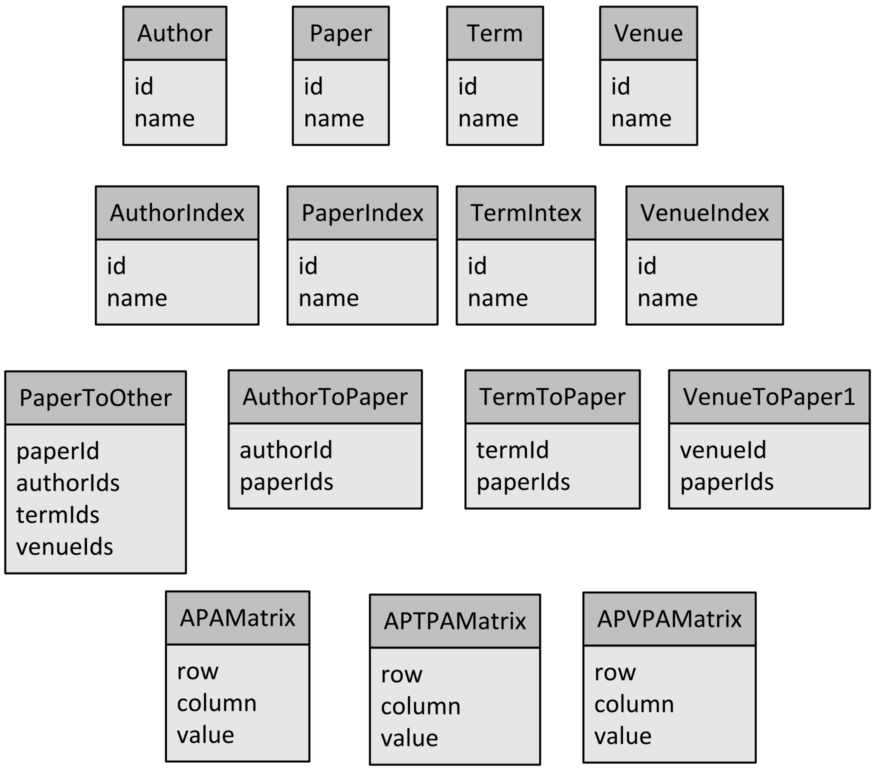


Figure 1. Database Schema Design

5.3 REST APIs
We built a set of rest services to load data and trigger the map reduce process to compute similarity. We also implemented rest services to upload and download files. All available rest services are listed in the table below:

API	URL	HTTP Method
Load definition and index data	/api/graph/data	POST
Load APA relation data	/api/graph/data/matrix/apa	POST
Load APTPA relation data	/api/graph/data/matrix/aptpa	POST
Load APVPA relation data	/api/graph/data/matrix/apvpa	POST
Run mapreduce process to compute similarity by author id	/api/graph/result/:id	GET
Get all available authors	/api/graph/author	GET
Upload file to Mongodb GridFS	/api/graph/uploads	POST
Download file from Mongodb GridFS	/api/graph/download	GET

Table 4: Available RESTful Services

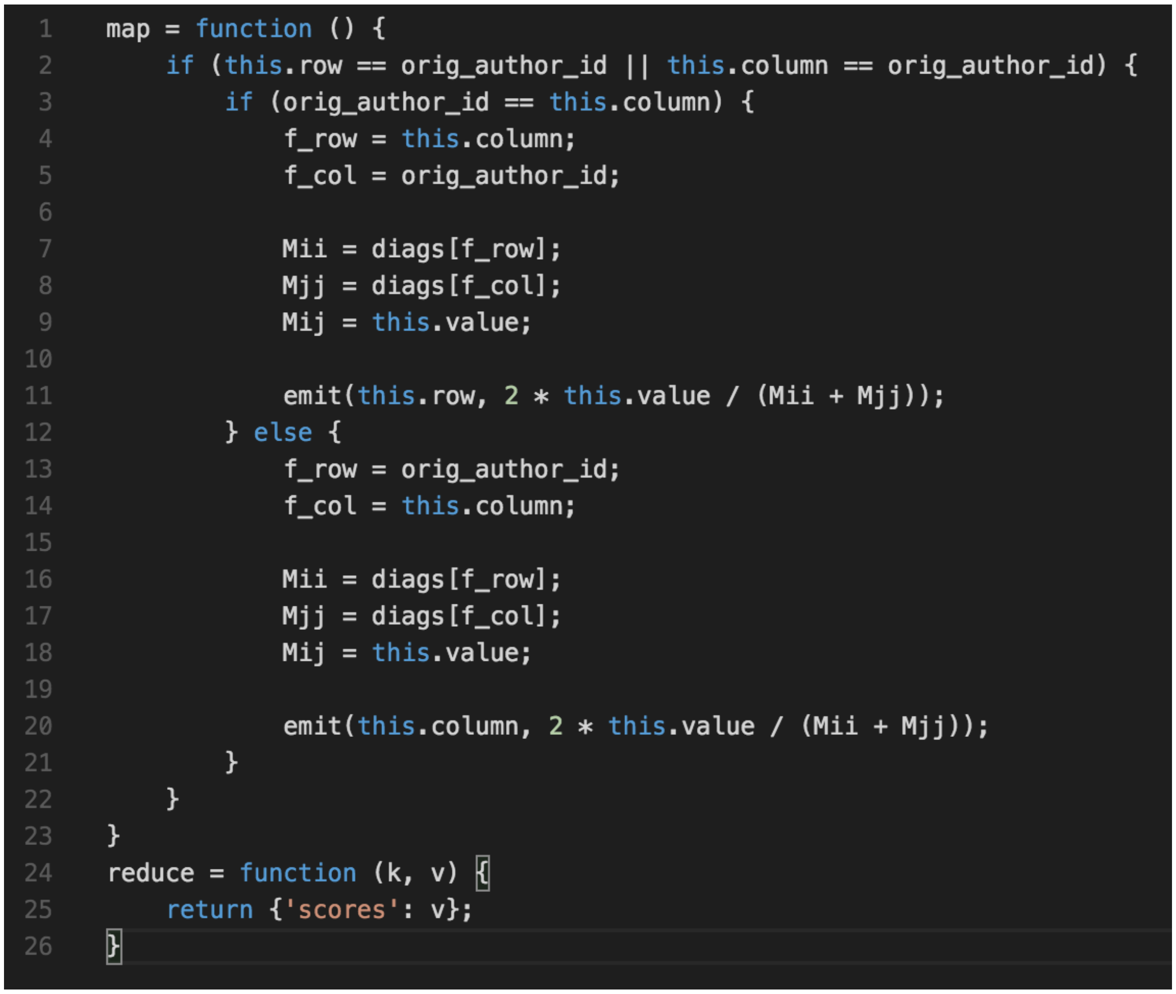
5.4 JavaScript Libraries

5.4.1 Twitter typeahead.js
We implemented typeajead for author name input so users know which authors are available in our system. We use a jQuery library from Twitter called typeahead.js, which is a flexible JavaScript library that provides a strong foundation for building robust typeaheads. We first retrieve author data using a rest api, which will query the database and return all authors. We then use this list of authors in the typeahead library as input data.

5.4.2 VisJS for result visualization
Vis.js is a dynamic, browser based visualization library that can handle large amounts of dynamic data and allow manipulation of and interaction with the data. We use vis.js to display the similarity relation between authors, papers, and venues. Vis.js expects the input data to contain nodes and edges, and it will render the graph based on the node and edge definitions. The definition of nodes and edges are from the response of the rest API which runs a map reduce job to compute similarity by author id.

5.4.3 AngularFileUpload and skipper for file upload
Angular File Upload is a module for the AngularJS framework. It supports drag-n-drop upload, upload progress, validation filters and a file upload queue. It supports native HTML5 uploads and works with any server side platform which supports standard HTML form uploads. Once user drop or select a file and click upload, the file uploader will call the file upload API. The file upload API uses the node.js library called skipper to stream the file directly to MongoDB GridFS.

5.5 Map Reduce
In order to calculate the similarity score for all pairs of authors (*a*, *b*) where *a* is a chosen author, we need to do around 5000 calculations. Interestingly, Map reduce is the perfect paradigm for this calculation. It allows parallel computation of the similarity score, thus reducing processing time. In the last step, we just need to utilize MongoDB's sort query to return the top 10 similar authors.

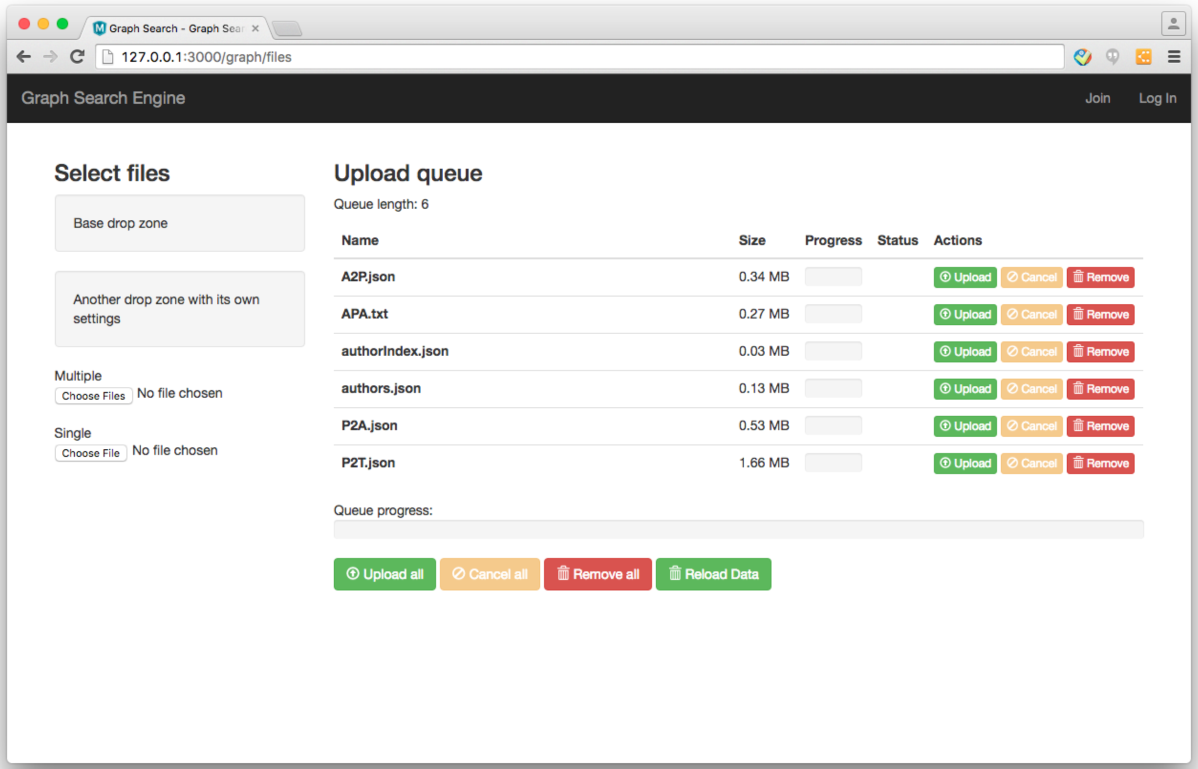


```
map = function () {
    if (this.row == orig_author_id || this.column == orig_author_id) {
        if (orig_author_id == this.column) {
            f_row = this.column;
            f_col = orig_author_id;

            Mii = diags[f_row];
            Mjj = diags[f_col];
            Mij = this.value;

            emit(this.row, 2 * this.value / (Mii + Mjj));
        } else {
            f_row = orig_author_id;
            f_col = this.column;

            Mii = diags[f_row];
            Mjj = diags[f_col];
            Mij = this.value;

            emit(this.column, 2 * this.value / (Mii + Mjj));
        }
    }
}
reduce = function (k, v) {
    return {'scores': v};
}
```

Figure 2: Map reduce functions

6. EVALUATION

6.1 File Upload
User can upload custom data to the system. We currently support DBLP dataset only. The following screenshot demonstrates the file upload page. Files will be streamed into MongoDB's GridFS directly. After uploaded the data files, user can click the 'Reload Data' button to send a POST request to the server to reload the database with new data files.



Select files

Base drop zone

Another drop zone with its own settings

Multiple
Choose Files No file chosen

Single
Choose File No file chosen

Upload queue

Queue length: 6

Name	Size	Progress	Status	Actions
A2P.json	0.34 MB			Upload Cancel Remove
APA.txt	0.27 MB			Upload Cancel Remove
authorIndex.json	0.03 MB			Upload Cancel Remove
authors.json	0.13 MB			Upload Cancel Remove
P2A.json	0.53 MB			Upload Cancel Remove
P2T.json	1.66 MB			Upload Cancel Remove

Queue progress:

Upload all Cancel all Remove all Reload Data

Figure 3: File Upload UI

6.1 Search by Author

First, user will need to enter an author name in the text box. We implemented typeahead to show user a list of authors we current have in our database. We current have information about 5000 authors. After entering a valid author name, user can click the search button to start the search.

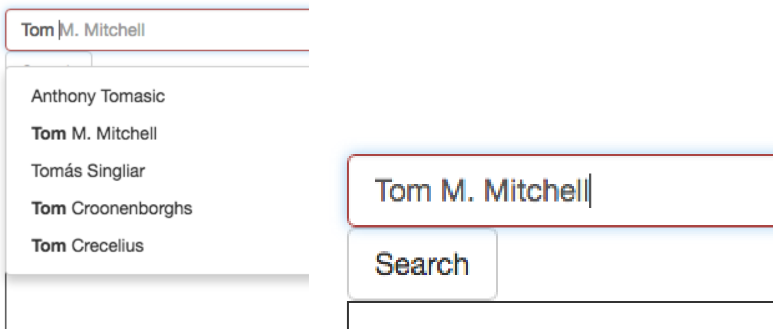
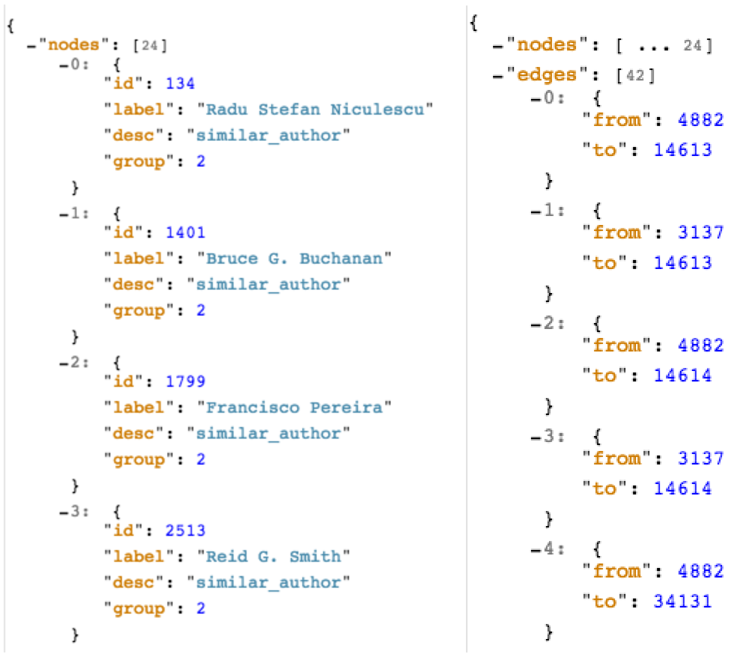


Figure 4. Author Search

This will send a request to the server to trigger a map reduce job to find the top 10 similar authors and their connections. The response contains an array of nodes and an array of edges, as shown below.

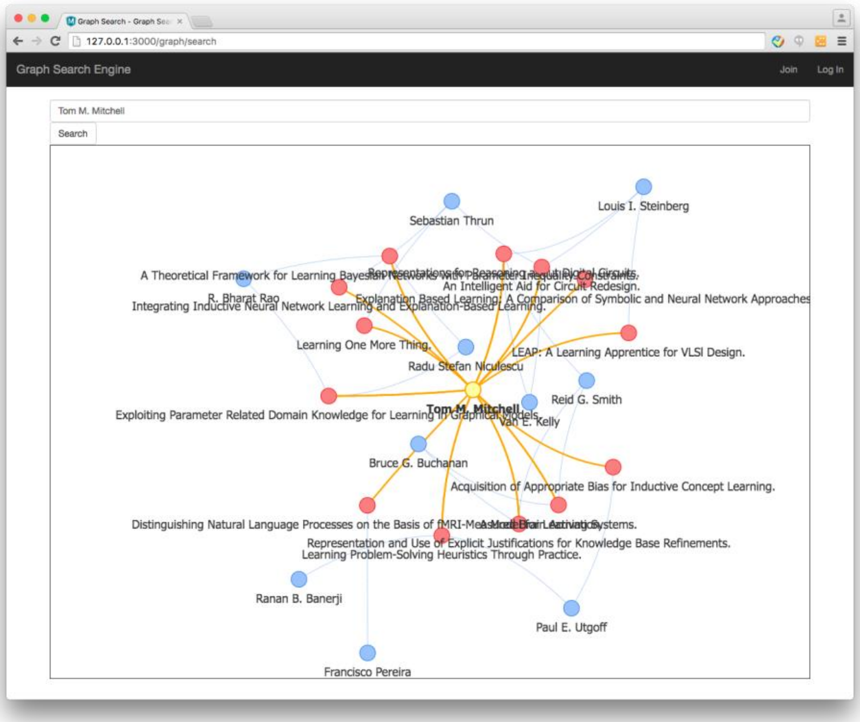


```
{
 -"nodes": [24]
   -0:  {
          "id": 134
          "label": "Radu Stefan Niculescu"
          "desc": "similar_author"
          "group": 2
        }
   -1:  {
          "id": 1401
          "label": "Bruce G. Buchanan"
          "desc": "similar_author"
          "group": 2
        }
   -2:  {
          "id": 1799
          "label": "Francisco Pereira"
          "desc": "similar_author"
          "group": 2
        }
   -3:  {
          "id": 2513
          "label": "Reid G. Smith"
          "desc": "similar_author"
          "group": 2
        }
```

```
{
 -"nodes": [ ... 24]
 -"edges": [42]
   -0:  {
          "from": 4882
          "to": 14613
        }
   -1:  {
          "from": 3137
          "to": 14613
        }
   -2:  {
          "from": 4882
          "to": 14614
        }
   -3:  {
          "from": 3137
          "to": 14614
        }
   -4:  {
          "from": 4882
          "to": 34131
        }
```

Figure 5. Top 10 authors response from server

We then render the result using vis.js. The image below demonstrates the result rendered by vis.js.



Graph Search - Graph Sea
127.0.0.1:3000/graph/search
Graph Search Engine
Join
Log In
Tom M. Mitchell
Search
Louis I. Steinberg
Sebastian Thrun
A Theoretical Framework for Learning Bayesian Networks with Parameter Inequality Constraints.
Representations for Reasoning about Digital Circuits.
An Intelligent Aid for Circuit Redesign.
R. Bharat Rao
Integrating Inductive Neural Network Learning and Explanation-Based Learning.
Explanation Based Learning: A Comparison of Symbolic and Neural Network Approaches
Learning One More Thing.
LEAP: A Learning Apprentice for VLSI Design.
Radu Stefan Niculescu
Reid G. Smith
Exploiting Parameter Related Domain Knowledge for Learning in Graphical Models
Tom M. Mitchell
Van E. Kelly
Bruce G. Buchanan
Acquisition of Appropriate Bias for Inductive Concept Learning.
Distinguishing Natural Language Processes on the Basis of fMRI-Measured Brain Activation.
A Model-Driven Learning Systems.
Representation and Use of Explicit Justifications for Knowledge Base Refinements.
Learning Problem-Solving Heuristics Through Practice.
Ranan B. Banerji
Paul E. Utgoff
Francisco Pereira

Figure 6. Top 10 authors result

7. CHALLENGES AND TAKEAWAY

We had many challenges when we try to deploy the application to a cloud server. We planned to use OpenShift cloud server at first, but we had a problem that there is not enough space to store our data files. Then we decided to swift to Google Cloud Platform, which will give us more flexibility and control. However, there are limited documentations and tutorials on deploying MEAN stack application to Google Cloud Platform. We are able to push to application to a Google Cloud Platform, but the server returns errors when we try to access the website.

After completing this project, we understand how SimPath algorithm works. We also gain experience on how to run map reduce job on a node.js application and send results back to the client via REST services. This give us better understand of how map reduce can be use in web application.

8. CONCLUSION AND FUTURE WORKS

In this project we introduce GraphSearch, a similarity search engine that operates on heterogeneous information network under the meta path framework. The engine utilizes PathSim similarity measure, MapReduce paradigm through MongoDB, and the MEAN web development framework to bring users a user-friendly interface, fast query and network visualization.

The search engine is easily extendable. Within the DBLP database, we can extend it to search for similar terms, or papers, or venues. We can also streamline and modify the importing and data preprocessing pipeline to work with other databases, such as the Internet Movie Data Base or networks in the medial fields (diseases/vaccines/etc.) We can also allow user to log in and save their searches. Another direction is to integrate it with DBLP in real time so new data can be incrementally updated to network, enabling users to query the most recent snapshot of DBLP. We can also extract out our SimPath computation process and build a pluggable node.js library. We will continue working on this project and make it a reliable website for users who are interested in similarity search of their custom dataset.

9. REFERENCES

[1] G. Jeh and J. Widom. SimRank: A Measure of Structural-Context Similarity. In KDD'02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 538-543. ACM Press, 2002.
[2] Y. Sun, J. Han, X. Yan, P. S. Yu, and T. Wu.Pathsim: Meta path-based top-k similarity searching heterogeneous information networks. PVLDB, 4(11):992-1003, 2011.
[3] MongoDB, https://www.mongodb.com
[4] Express.js, http://expressjs.com
[5] AngularJS, https://angularjs.org
[6] Node.js, https://nodejs.org
[7] vis.js, http://visjs.org
[8] typeahead.js, https://twitter.github.io/typeahead.js