Comparison Analysis of Link Prediction Algorithms for Recommender Systems*

* Class Project of Course: CPTS 591 (Elements of Network Science)

Shivani Sawant Md Mahedi Hasan Madhumitha Sivakumaran
Washington State University (WSU), Pullman, USA

***Abstract*—The link prediction method anticipates the likelihood of a future connection between two nodes in a given network. In this study, a comprehensive comparison among the link prediction algorithms is made for recommender systems by focusing on the exploratory data analysis and utilizing the insights to build the machine learning models for better accuracy. This study used a publicly available "Citations" network dataset for link prediction. Exploratory Data Analysis (EDA) shows that the data has three nodes: venue, articles, and authors, and found that 99% ariticles have eight citations or less; on the other hand, 99% articles cite six or fewer other articles. Three link prediction algorithms (common neighbours, preferential attachments, and total neighbours) were implemented, and the results show that common neighbour has the most promising performance in predicting the links.**

***Index Terms*—Link prediction, Recommender system, Random forest**

I. Introduction

In network theory, "link prediction" is the problem of predicting the existence of a link between two entities in a network. Link prediction can also be defined as a process of predicting future connections between pairs of unconnected links based on existing connections [5].To be more specific, a link prediction measure is used to forecast whether or not there is a future link between pairs of unconnected nodes or vertices. Let us consider a graph where it represents nodes as entities and edges as interactions or links. The graph, G = (V, E) consists of a set of nodes or vertices, V and a set of links or edges, E. For a given graph, each edge, e = (u, v) represents an interaction between u and v that takes place in a particular time, e(t). The following graph will help us to further explain the concept of link prediction [5],

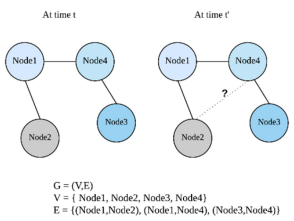


Fig. 1: Example for simple network graph of V and E.

Here, in Figure 1, Node1 is linked to Node2 and Node4 simultaneously. However, Node2 and Node4 are not linked to each other but Node1 is a common node of Node2 and Node4. Hence, link prediction should anticipate a connection between Node2 and Node4.

Some of the common example of link prediction are, predicting friendship links among users in social networks, predicting co-authorship links in a citation network, or may be predicting interactions between genes and proteins in a biological networks etc. In recent past, the link Prediction has become one of the most widely researched and implemented technique used in the social network analysis(SNA) [1]. People communicate and express their comments, likes and interests via social network as it provides a fast and easy way to share. As a result, it creates a complex connection within the social network and this can be immediately visualized as large graphs. With the dynamic and exponential rise in the number of new users, this graph expands over time, with some of the users come from unreliable and untrusted sources.Although connectivity increases the link between users, it also attracts unwanted users and links such as fake users and bots. This fact is true for any similar network analysis such as customer segmentation, recommender systems, viral marketing, biological interaction networks etc.

Alongside, it is quite evident that there are various approaches and methods available for link prediction in network analysis. The performances of different prediction methods are different and having different level of complexities in analysing the network data. This project also aims to compare and analyze the performance of some selected link prediction algorithms for the recommender systems using a publicly available network dataset. The aim of the recommendation task would be to (i) find prospective collaborators for authors, and (ii) find related papers about a topic. This task would make it easy for authors to find like-minded collaborators and also effectively search related background research. In this work, we have implemented three link prediction algorithms to a link prediction classifier.

II. Related Works

The area of research for link prediction is quite vast and also a numerous techniques are available for link prediction,

in general. It is apparent from the existing literature that a notable amount of research works aim to address the data sparsity in recommender systems by using either a novel link prediction algorithm or building on top of existing methods [3], [4]. In such cases, the application of link prediction tries to minimize the existing problems and increase the detection process. In one such study, the authors proposed a recommendation system that presented a solution for transfer link prediction problem across heterogeneous networks [6]. In different studies, different researcher showed the application of link prediction on anomaly detection for malicious users [7], community detection system [8], and information diffusion network on the viral events [9]. On the other hand, most of these works emphasize the results of their models using some specific types of dataset such as movie recommendation or friendship datasets.

Anther vast area of link prediction research is the application of link prediction in social networks [5]. This prediction is essential in social networks to infer social interactions or to suggest possible friends to the users. Rapid growth of social network data and especially with the significant advancement in complex social network modeling trigger link prediction analysis to be more challenging. However, it is mentioned that the link prediction applications namely, recommendation system, anomaly detection, influence analysis and community detection become more strenuous due to network diversity, complex and dynamic network contexts.

The approaches and methods that are available for link predictions are often classified as similarity based, probabilistic, algorithmic, and hybrid approaches [**?**], and there has been quite a number of methods found in application on each of those approaches.However, there has been a little work found that are done to compare the traditional link prediction algorithms using the citations dataset. A comprehensive comparison analysis will help to identify the limitations and advantages of the existing link prediction algorithms.

III. Problem Definition

In network theory, link prediction is the problem of predicting the existence of a link between two entities in a network". Identifying the link between two networks is often an interest of research. As a result, it leads us to some crucial questions; can we identify those connections? how strong are the connections? how does the engagement between nodes help to predict the future nodes? In this project, we specifically intend to focus on predicting the links in recommender systems for a publicly available citations dataset and get the answer of those questions.Successful implementation of the work would make it easy for authors to find like-minded collaborators and also effectively search related background research.

IV. Data and Data Sources

The dataset used in this project is a publicly available "Citations" network dataset that contains research citations extracted from DBLP, ACM, and MAG [2]. It contains 629,814 papers, and 632,752 citations. Each paper is associated with a set of entities including title, year, abstract, authors, and venue. We have used the citation papers of 2021. The dataset was of "json" format.

In order to give an essence of the citations network dataset, we have included the following graph.

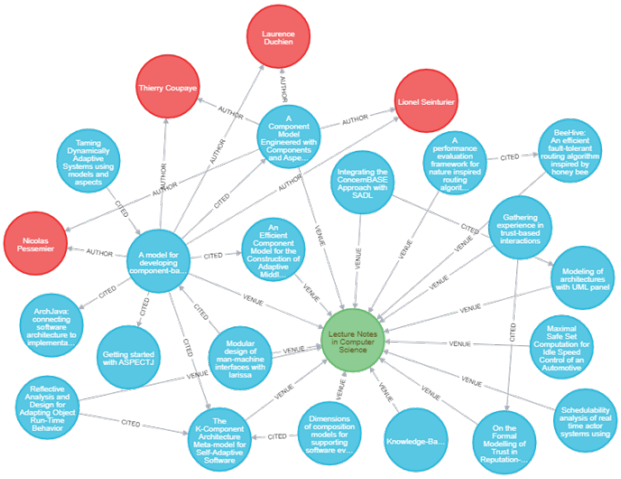


Fig. 2: Sample data visualization

V. Link Prediction Algorithms

This work focuses on both application and the evaluation of different existing models/algorithms for link prediction. We extensively investigate the parsimony of the models for making prediction. In the process, the following link prediction and machine learning algorithms were used during the course of this project:

1) Common Neighbors
2) Preferential Attachment
3) Total Neighbors
4) Random Forest Binary Classifier

To study the ability of link prediction algorithms for generating recommendations, three algorithms were compared and analyzed to predict if two given authors will collaborate in the future. Here is a detailed description of the algorithms:

Common neighbors algorithm determines the closeness of a given pair of nodes in a graph. It works on the idea that any two strangers who have a common friend are more likely to be introduced in future compared to those who do not have common friends. Mathematically, it can be computed as follows:

$$\mathbf{CN}(\mathbf{x}, \mathbf{y}) = |\mathbf{N}(\mathbf{x}) \cap \mathbf{N}(\mathbf{y})| \tag{1}$$

where, $\mathbf{N}(\mathbf{x})$ denoted the set of nodes adjacent to node x and $\mathbf{N}(\mathbf{y})$ denotes the set of nodes adjacent to node y. Furthermore, A value of zero denotes that a given pair of two nodes are not close, and higher values represent that the nodes are closer.

Preferential Attachment was first popularized by Albert-László Barabási and Réka Albert. It works based on the idea that users that have many existing friends tend to make more connections in the future. In the context of link prediction, it suggests that a node that is more connected, is more likely to receive new links in the future. To measure the likeliness of two nodes connecting, we calculate the product of the number of friends each node has. Mathematically, it is represented as follows:

$$\mathbf{PA}(\mathbf{x}, \mathbf{y}) = |\mathbf{N}(\mathbf{x})| \times |\mathbf{N}(\mathbf{y})| \tag{2}$$

Here, a value of zero denotes that the two nodes are not close, and higher values represent that the nodes are closer.

Total Neighbors works similar to the idea of common neighbors and preferential attachment in the sense that mode connected a node is, the more likely it is to receive new links, however it computes the closeness of these nodes using the unique number of neighbors they have. Hence the closeness of two nodes is mathematically computed as follows:

$$\mathbf{TN}(\mathbf{x}, \mathbf{y}) = |\mathbf{N}(\mathbf{x}) \cup \mathbf{N}(\mathbf{y})| \tag{3}$$

where $\mathbf{N}(\mathbf{x})$ denoted the set of nodes adjacent to node x,and $\mathbf{N}(\mathbf{y})$ denotes the set of nodes adjacent to node y. A value of zero denotes that a given pair of two nodes are not close, and higher values represent that the nodes are closer.

Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. Steps involved in random forest algorithm:

1) Step 1: In Random forest n number of random records are taken from the data set having k number of records.
2) Step 2: Individual decision trees are constructed for each sample.
3) Step 3: Each decision tree will generate an output.
4) Step 4: Final output is considered based on Majority Voting or Averaging for Classification and regression respectively.

Finally, to implement the same, we have used various libraries in Python including “Networkx” for exploratory data analysis, “scikit-learn” for machine learning classification algorithms, and lastly “Neo4j” for graph algorithms and visualization.

VI. Implementation

A. Data Processing

We used the “Neo4j” platform, a graph database management and Python libraries to visualize and plot the distributions and graphs. To combine the “Neo4j” database and python libraries ,we imported the “py2neo” library which is used to import the data into “Neo4j”. “py2neo” is a client library and toolkit for working with “Neo4j” from within Python applications. We used a special type of query named “Cypher” to interact with the database.

We did some data pre-processing to remove the inconsistency in the data. First we loaded the data into database by unwinding the “json” files and then removed the duplicates, deleted all articles with no titles.Finally, the data was ready to proceed with Exploratory Data Analysis and then apply machine learning models.

Exploratory Data Analysis is performed to gain more insight on the dataset, which is discussed in Results and Discussion,in detail.

B. Implementation of Algorithms

After EDA, we implemented the algorithms for predicting links, specifically we implemented;

1) the link-prediction algorithms to predict the link between two nodes
2) and use the scores fro the algorithms as features to train a “binary classifier”

The following schematic diagram shows the steps followed to implement the link prediction algorithms.

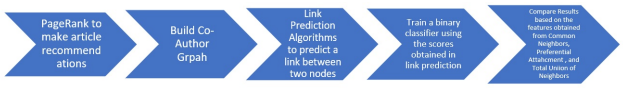


Fig. 3: Steps in implementing the link-prediction algorithms

To build a link prediction classifier, we first started by creating a co-author graph which is based on people collaborating on similar papers. To implement this, Neo4j’s “cypher” queries were used. In addition to that, the results were ordered by the year of the author’s first collaboration. Next step was to figure out ways to use this co-author graph to predict future relationships/ collaborations between authors. For this task, three link prediction algorithms, namely Common Neighbors, Preferential Attachment, and Total Neighbors were chosen to test their effectiveness for our task. The aim was to compare and analyze the performance of all three and make inferences about their abilities in link prediction. Normally, we could have computed the individual scores of all three aforementioned algorithms and made prediction inferences using a threshold value, however, our hypothesis being tested

here was that the prediction scores computed using these algorithms can be used as features for a machine learning binary classifier model for better prediction accuracy.

The aim of this binary classifier is to predict if a given pair of nodes(authors) will have a link(collaboration) or not. To proceed with this supervised learning task, we first split our data into a training and testing set. The challenge in this task was to avoid data leakage that could have occurred with a random split of data. To handle this, we split sub-graphs by using "year" of publication as a constraint to split the graph. This query resulted in a 52-48 train-test split. Next task was to make sure we have negative examples in our train test datasets to have a 0 label indicating that there exists no relationship between the two nodes (authors). To do this, a "cypher" query was written and executed to find authors that are two to three hops away. Finally, this was collated in a new dataframe consisting of three columns, namely "node1", "node2", and "label" one for each training and testing set. Next, we computed the scores of closeness by using each of the three link prediction scores and appended those to our train test data frames. The final structure of our data frame used for training a machine learning model looked as follows:

node1	node2	label	cn	pa	tn

Fig. 4: Structure of Data frame

Next, we chose Random Forest Classifier due to its ability of efficiently handling large datasets, to perform our aimed binary classification. For our first classifier, we only chose the closeness scores computed by Common Neighbors as features, and trained a random forest classifier on maxdepth =10, and nestimators=30 as the hyper parameters. Following this, in order to compare the results with other link prediction algorithms, we trained a similar random forest classifier model again, however this time incorporating the closeness scores of Preferential Attachment and Total Neighbors as well as features for training. The results of this comparison analysis are discussed in the following section.

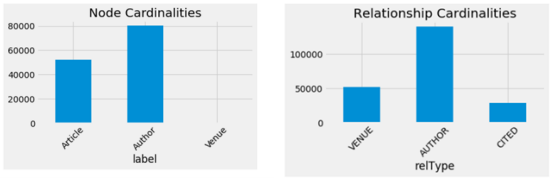
VII. Results and Discussion

A. Exploratory Data Analysis (EDA)

Exploratory data analysis is an approach of analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods. Here, we visualized the overall representation of the dataset and know about the main nodes present in it.

Fig. 5: Nodes in the database

Now, from Figure 5 we see the main nodes and the links in the dataset.The dataset has three nodes, venue, article and authors. Drilling down a bit, We analyzed the number of nodes present of each main label. Besides, we tried to know about the number of different type of relationship the article shares. The following graph have those information.



(a) Distribution of node cardinalities in the database

(b) Relationship types in the data

Fig. 6: Results from EDA

Figure 6a shows the count for number of nodes for venue, article, and author as 4, 51956, and 80299 respectively. Furthermore, the Figure 6b showed that there are three types of relationships in the data; cited, venue and author. The count for each of these are, 28706, 51956, and 140575 respectively.

In the Appendix Figure 13,to explore more about the citation, we queried to get the first 25 authors, articles, venues, and also shows the number of articles that article has cited, as well as the number of articles that it's been cited. Afterwards, to examine the citations, cited by other authors and the number of articles published by authors data more closely, we counted the number of citation, cited, and published articles and then created a distribution of those counts. The following figure have those information.

	citations
count	51956.000
mean	0.553
std	2.418
min	0.000
25%	0.000
50%	0.000
75%	0.000
90%	1.000
99%	8.000
max	211.000

(a) Citations

	cited
count	51956.000
mean	0.553
std	1.301
min	0.000
25%	0.000
50%	0.000
75%	1.000
90%	2.000
99%	6.000
max	51.000

(b) Cited

	published
count	80299.000
mean	1.751
std	2.064
min	1.000
25%	1.000
50%	1.000
75%	2.000
90%	3.000
99%	10.000
max	89.000

(c) Published

Fig. 7: Summary information based on each of the article

Figure 7a talks about the citations each article has and its distribution. This gives us the insight that most articles are cited very few times. 99% of the articles have 8 citations or less and there is a single article which has been cited for more than 200 times. Besides, Figure 11 illustrated the log scale histogram to visualize it. Then we digged in to know about how many articles cite the most other papers, and the Figure 7b showed this information. From this, we can see that most articles does not cite the most other papers.99% of the article cite 6 or less other papers. Similarly, We plotted the graph to visualize it in the Appendix Figure 12. Finally, we have Figure 7c that has the information about the distribution of the number of articles published by authors.

B. Link Prediction Algorithm

The results obtained by the random forest classifier give us information on whether or not a link would exist between any two given authors with label 1 indicating a suggested recommendation, and 0 indicating that there would not exist a link(collaboration) between the authors.
To evaluate the results of the aforementioned recommendation system, we used three measures namely, accuracy, precision, and recall to compare how random forest classifiers performed with different features.

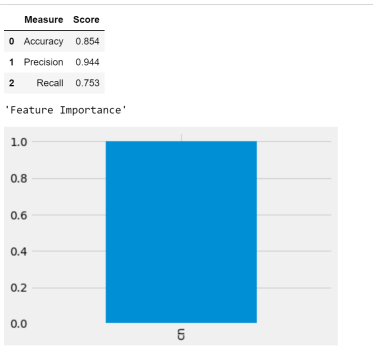


Fig. 8: Random Forest Classifier Algorithm as Recommender for “Common Neighbors”

The Figure 8 above shows the results of the first random classifier used as a recommender. The results showed pretty good performance with common neighbors used as the only feature to train the classifier model. However, we aimed to compare it with link prediction algorithms and as a result, we included the closeness scores from preferential attachment and total neighbors in the feature list to train another random forest classifier. The Feature Importance graph conveys how significant the feature was in training the model.

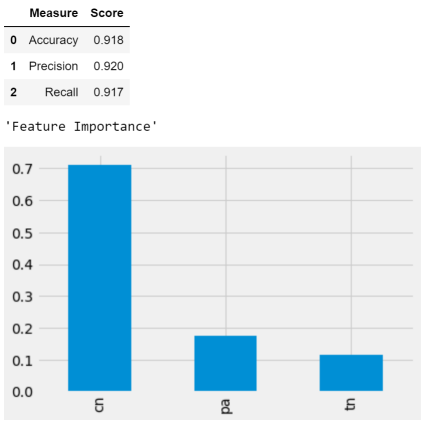


Fig. 9: Random Forest Classifier Algorithm as Recommender for “Common Neighbors”, “Preferential attachment” and “Total Neighbors”

The above Figure 9 shows the results for the random classifier model trained using the closeness scores of all tree algorithms common neighbors, preferential attachment, and total neighbors. It is clear that adding the features from more link prediction algorithms resulted in an improvement of the model. This supports our hypothesis that the results of link prediction algorithms can be used as features to build a recommender system.

VIII. Conclusion

This study on the link prediction for the citations data set helps us to explore the general features of citations, and the implementation of different link prediction algorithms helps to predict the links among nodes in the citation networks. The results showed that the performance of different algorithms varies in predicting links, hence the prediction algorithms for links need to be chosen carefully. Besides, we have found some interesting facts that are worth of extending further. As we came to know more about the relationships between authors associated with the articles, citation received and given of the article and the venues of it, We went further to proceed with recommendation by finding authors who have written the most articles, retrieve the articles they’ve published and how many citations they’ve received for collaborator suggestion.

ACKNOWLEDGMENT

First of all, we would like to express my deep sense of gratitude to Dr. Assefaw Gebremedhin for taking our class and making the learning of this course interesting. Throughout the semester we have had an rigorous and effective training of both the theoretical and applied aspects of network data analysis and interpretation. We are also thankful to the researchers whose works we have cited and learned from.

IX. APPENDIX

A. GitHub Repository Link

GitHub Link with the Implemented codes

B. Appendix figures

	label	count
2	Venue	4
0	Article	51956
1	Author	80299

(a) Count for node cardinalities

	relType	count
2	CITED	28706
0	VENUE	51956
1	AUTHOR	140575

(b) Count for relationship types in the network

Fig. 10: Node and Link distribution in the dataset

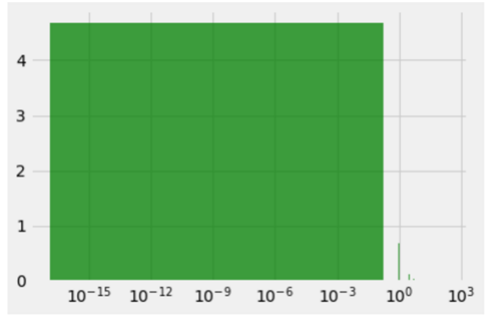


Fig. 11: Log scale of citations information

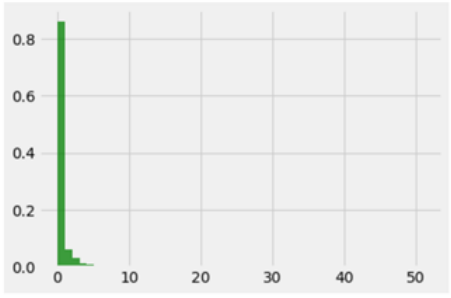


Fig. 12: Log scale of cited information

	article	author	venue	citationsGiven	citationsReceived
0	A New Genetic Operator for the Travelling Salesman Problem	José Ali Moreno	Lecture Notes in Computer Science	0	0
1	A New Genetic Operator for the Travelling Salesman Problem	Néstor Carrasquero	Lecture Notes in Computer Science	0	0
2	Design and implementation of multipurpose single channel bio signal amplifier	R.S. Sandesh	advances in computing and communications	0	0
3	Design and implementation of multipurpose single channel bio signal amplifier	Nithya Venkatesan	advances in computing and communications	0	0
4	U-Create: creative authoring tools for edutainment applications	Sebastian Sauer	Lecture Notes in Computer Science	0	0
5	U-Create: creative authoring tools for edutainment applications	M. Stifter	Lecture Notes in Computer Science	0	0
6	U-Create: creative authoring tools for edutainment applications	Xavier Wielemans	Lecture Notes in Computer Science	0	0
7	U-Create: creative authoring tools for edutainment applications	Kerstin Osswald	Lecture Notes in Computer Science	0	0
8	Characterizing and detecting anti-patterns in the logging code	Boyuan Chen	international conference on software engineering	7	0
9	Characterizing and detecting anti-patterns in the logging code	Zhen Ming Jiang	international conference on software engineering	7	0
10	Towards reusable components with aspects: an empirical study on modularity and obliviousness	Kevin J. Hoffman	international conference on software engineering	3	0
11	Towards reusable components with aspects: an empirical study on modularity and obliviousness	Patrick Eugster	international conference on software engineering	3	0
12	An approach for solving very large scale instances of the design distribution problem for distri...	Juan Frausto-Solís	Lecture Notes in Computer Science	0	0
13	An approach for solving very large scale instances of the design distribution problem for distri...	J H Héctor Fraire	Lecture Notes in Computer Science	0	0
14	An approach for solving very large scale instances of the design distribution problem for distri...	O Joaquín Pérez	Lecture Notes in Computer Science	0	0
15	An approach for solving very large scale instances of the design distribution problem for distri...	S. Gerardo Reyes	Lecture Notes in Computer Science	0	0
16	An approach for solving very large scale instances of the design distribution problem for distri...	A R Rodolfo Pazos	Lecture Notes in Computer Science	0	0
17	An approach for solving very large scale instances of the design distribution problem for distri...	R Laura Cruz	Lecture Notes in Computer Science	0	0
18	An approach for solving very large scale instances of the design distribution problem for distri...	S René Santaolaya	Lecture Notes in Computer Science	0	0
19	Taming heterogeneous agent architectures	Carlos José Pereira de Lucena	Communications of The ACM	2	0
20	Taming heterogeneous agent architectures	Alessandro Garcia	Communications of The ACM	2	0
21	Towards a Formal Specification of Complex Social Structures in Multi-agent Systems	Pere Garcia	Lecture Notes in Computer Science	0	1
22	Towards a Formal Specification of Complex Social Structures in Multi-agent Systems	Francisco Javier de Lucas Martín	Lecture Notes in Computer Science	0	1
23	Towards a Formal Specification of Complex Social Structures in Multi-agent Systems	Juan A. Rodríguez-Aguilar	Lecture Notes in Computer Science	0	1
24	Towards a Formal Specification of Complex Social Structures in Multi-agent Systems	Pablo Noriega	Lecture Notes in Computer Science	0	1

Fig. 13: Citation layout

C. Implemented Codes for EDA

```
Data and Data Preprocessing:
# remove duplicates
display(graph.run("CREATE CONSTRAINT ON
(a:Article)ASSERT a.index IS UNIQUE").stats())
display(graph.run("CREATE CONSTRAINT ON
(a:Author) ASSERT a.name IS UNIQUE").stats())
display(graph.run("CREATE CONSTRAINT ON
(v:Venue) ASSERT v.name IS UNIQUE").stats())

# load data
query = """
CALL apoc.periodic.iterate(
  'UNWIND ["dblp-ref-0.json",
  "dblp-ref-1.json", "dblp-ref-2.json",
  "dblp-ref-3.json"]
  AS file CALL apoc.load.json
  ("https://github.com/neo4j-contrib/
  training-v3/raw/master/modules/
  gds-data-science/supplemental/data/" + file)
   YIELD value WITH value
   return value',
  'MERGE (a:Article {index:value.id})
   SET a += apoc.map.clean(value,
   ["id","authors","references", "venue"],[0])
   WITH a, value.authors as authors,
   value.references AS citations,
   value.venue AS venue
   MERGE (v:Venue {name: venue})
   MERGE (a)-[:VENUE]->(v)
   FOREACH(author in authors |
     MERGE (b:Author{name:author})
     MERGE (a)-[:AUTHOR]->(b))
   FOREACH(citation in citations |
     MERGE (cited:Article {index:citation})
```

```
     MERGE (a)-[:CITED]->(cited))',
   {batchSize: 1000, iterateList: true});
"""
graph.run(query).to_data_frame()

# removing article with no title
query = """
MATCH (a:Article) WHERE not(exists(a.title))
DETACH DELETE a
"""
graph.run(query).stats()

EDA:
#overall represntation
graph.run("CALL db.schema.visualization()").data()

#nodes of each label
result = {"label": [], "count": []}
for label in graph.run("CALL db.labels()").
to_series():
    query = f"MATCH (:`{label}`) RETURN count(*)
    as count"
    count = graph.run(query).to_data_frame().
    iloc[0]['count']
    result["label"].append(label)
    result["count"].append(count)
nodes_df = pd.DataFrame(data=result)
nodes_df.sort_values("count")

# plot
nodes_df.plot(kind='bar', x='label', y='count',
legend=None, title="Node Cardinalities")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

# relation type in graph
result = {"relType": [], "count": []
for relationship_type in graph.run("CALL
db.relationshipTypes()").to_series():
    query = f"MATCH ()-
    [:`{relationship_type}`]->()
    RETURN count(*) as count"
    count = graph.run(query).
    to_data_frame().iloc[0]['count']
    result["relType"].append(relationship_type)
    result["count"].append(count)
rels_df = pd.DataFrame(data=result)
rels_df.sort_values("count")

# plot
rels_df.plot(kind='bar', x='relType', y='count',
legend=None, title="Relationship Cardinalities")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```

```
# citation network
exploratory_query = """
MATCH (author:Author)<-
[:AUTHOR]-(article:Article)-
[:VENUE]->(venue)
RETURN article.title AS article,
author.name AS author, venue.name AS venue,
size((article)-[:CITED]->())
AS citationsGiven,
size((article)<-[:CITED]-())
AS citationsReceived
ORDER BY rand()
LIMIT 25
"""
graph.run(exploratory_query).to_data_frame()

# citation given for article
query = """
MATCH (a:Article)
RETURN size((a)<-[:CITED]-()) AS citations
"""

citation_df=graph.run(query).to_data_frame()
citation_df.describe([.25, .5, .75, .9, .99])

# plot
fig1, ax1 = plt.subplots()
ax1.hist(pd.Series
(citation_df['citations'].dropna()),
1250,density=True,facecolor='g',alpha=0.75)
ax1.set_xscale("log")
plt.tight_layout()
plt.show()


# citation received for article
query = """
MATCH (a:Article)
RETURN size((a)-[:CITED]->()) AS cited
"""

cited_df = graph.run(query).to_data_frame()
cited_df.describe([.25, .5, .75, .9, .99])
#plot
fig1, ax1 = plt.subplots()
ax1.hist(pd.Series(cited_df['cited'].dropna()),
50, density=True, facecolor='g', alpha=0.75)
plt.tight_layout()
plt.show()

# published by authors
query = """
```

```
MATCH (a:Author)
RETURN size((a)<-[:AUTHOR]-()) AS published
"""
published_df = graph.run(query).to_data_frame()
published_df.describe([.25, .5, .75, .9, .99])
```

D. Implemented Codes for Link Prediction Algorithm

The "python codes" are fairly long and we have added a seperte file (as pdf) with the submission for the codes with implementation for this part.

REFERENCES

[1] A. Kumar, S. S. Singh, K. Singh, and B. Biswas, -2 node clustering coecient-based link prediction," Applied Intelligence, vol. 49, no. 7, pp. 27622779, 2019.

[2] M. Farber and A. Jatowt, recommendation: approaches and datasets," International Journal on Digital Libraries, vol. 21, no. 4, pp.375405, 2020.

[3] H. Liu, H. Kou, C. Yan, and L. Qi, prediction in paper citation network to construct paper correlation graph," EURASIP Journal on Wireless Communications and Networking, vol. 2019, no. 1, pp. 112, 2019.

[4] H. Chen, H. Yin, W. Wang, H. Wang, Q. V. H. Nguyen, and X. Li, : projected metric embedding on heterogeneous networks for link prediction,"in Proceedings of the 24th ACM SIGKDD international conference on knowl-edge discovery data mining, 2018, pp. 11771186.

[5] Daud NN, Ab Hamid SH, Saadoon M, Sahran F, Anuar NB. Applications of link prediction in social networks: A review. Journal of Network and Computer Applications. 2020 Sep 15;166:102716.

[6] Dong Y, Tang J, Wu S, Tian J, Chawla NV, Rao J, Cao H. Link prediction and recommendation across heterogeneous social networks. In2012 IEEE 12th International conference on data mining 2012 Dec 10 (pp. 181-190). IEEE.

[7] Kagan D, Elovichi Y, Fire M. Generic anomalous vertices detection utilizing a link prediction algorithm. Social Network Analysis and Mining. 2018 Dec;8(1):1-3.

[8] Mohan A, Venkatesan R, Pramod KV. A scalable method for link prediction in large real world networks. Journal of Parallel and Distributed Computing. 2017 Nov 1;109:89-101.

[9] Lu X, Szymanski B. Predicting viral news events in online media. In2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW) 2017 May 29 (pp. 1447-1456). IEEE.

[10] Martínez V, Berzal F, Cubero JC. A survey of link prediction in complex networks. ACM computing surveys (CSUR). 2016 Dec 20;49(4):1-33.