International Journal of Advanced Science and Technology
Vol. 29, No. 7s, (2020), pp. 1721-1734

Intelligent Model for Research Peer Recommendation

Vanitha Sivagami S[1], Abirami Sri S[2]
Department of Computer Science and Engineering,
Mepco Schlenk Engineering college, Sivakasi.

Abstract

Social networking provides the ability to connect to other people all over the world. It can be represented by the graph in which nodes represent people and edges are used to show the connection between them. The tendency of people with the similar taste or choices in a social network leads to the formation of communities. Community detection in a social network is becoming a hot topic now-a-days since it is useful for advertising and marketing purposes. None of the existing community retrieval algorithm provides a real time view of the problem .i.e) It does not recommend the optimal peers to the user. The objective of this paper is to develop the efficient community retrieval algorithm that will recommend the optimal research peers to the new user who is looking for collaboration. Initially, research peer information is converted into the form of attributed graph. In order to make searching efficient Core Label(CL) tree is constructed based on the core numbers of vertices of the graph. By using an efficient community retrieval algorithm, the optimal communities are retrieved and recommended to the user. If the retrieved community is not of suitable size then keyword set expansion takes place and searching process is repeated again. This will be continued until the community of suitable size is retrieved. The experimental results show that our proposed model performs well even for large size graph

Keywords: *Medical graph, core, tree, recommendation, ranking, community, search, expansion.*

1. Introduction

The internet has changed the way people interact with each other as well as the culture of work. Social media helps people build better relationships with family and friends, so people spend a lot of their time browsing social sites online. This causes over quintillion bytes of data to be generated every day. In order to insight some useful information from this large amount of data it is represented in the form of a graph .i.e) analysis of the social network can be performed through the use of networks and graph theory. It characterizes networked structures in terms of nodes (individual actors, people, or things within the network) and bridging edges or links (relationships or interactions). The advantage of using graphical model is, it represents the properties of the network more elaborately and systematically. Many numbers of graphical methods are available for identifying the similar groups in a network since it is vital in driving the spread of information. So detecting communities are more useful for advertising and marketing purposes. Similarly, researchers citation details can be modeled into the graph. From the developed graph it is possible to retrieve the researchers who have similar kind of interest. It will be more useful for new researcher who looking for collaboration .i.e) the new researcher can develop new ideas by inferring the techniques and ideas of existing researcher.

In this paper, an intelligent model for research peer recommendation is proposed. Initially, the input dataset is parsed and attributes for the graph is created after doing preprocessing steps like hyphen removal, stop words removal, lemmatization, digit and dot removal. Using the created attributes the attributed graph is constructed. Then the attributed graph is converted into the form of a Core Label(CL) tree structure in order to make searching efficient. By using an efficient community retrieval algorithm, a set of optimal keyword sets is identified. Then the model will calculate score for each keyword set in optimal keyword sets, in order to find the most relevant one. Finally, output the peers whose keyword set are approximately same as relevant keyword set found before.

ISSN: 2005-4238 IJAST

2. LITERATURE REVIEW

In recent years, several different technologies have been developed for finding the community[1]. One of such method is C-Explorer, it displays the communities for the given query vertex q and allow user to view the result as the interesting graph [2]. It uses an attributed graph whose vertices are associated with labels and keywords and searches for an attributed community (or AC) whose vertices are structured and semantically related by using Community Retrieval (CR) algorithm. It also implements the function for analyzing their effectiveness.

Searching in a spatial graph is a complex task. [3] addresses the problem of performing online community search over large spatial graph. It can yield communities that span large areas with vertical locations.i.e.) It tries to find the Spatial-Aware Community (or SAC) whose vertices is structurally and spatially close as possible while the spatial cohesion focuses on the closeness between their Geo-locations but it doesn't include the temporal information about the graph. [4] incorporates both temporal and attribute information along with the structural properties of the graph. It can handle graphs with heterogeneous type of attributes and structural and attribute information changes are handled perfectly that are essential for applicability to real world networks.

With the change in lifestyle and interests, the social activities of the people have a dynamic changing tendency. The static culture was therefore unable to reflect the real activities. The individual tends to get in touch with the closest friends. [5] uses that property to find out the path from one node to its closest nodes, and the guided network created can easily identify the communities. Instead of considering the global topological structure of the social network alone, it utilizes the Random Walk Sampling (RWS) and Adaptive Random Walk Sampling (ARWS) method to find the closest friends for each node.

Communities present in attributed graphs such as social networks and information bases can be used in evolving applications such as brand marketing and social events set up. [6] investigates the Attributed Community Query(ACQ) which returns an Attributed Community(AC) for an attributed graph. The AC is a subgraph of G that satisfies both the cohesiveness of the structure (i.e., its vertices are closely linked) and the cohesiveness of the keyword (i.e., its vertices share common keywords). It do two steps verification by candidate generation and by using Core Label(CL) tree it identifies the community.

The tracking of communities is computationally expensive and unstable in dynamic networks since it require network community detection in each of the evolutionary step for the entire network. [7] uses the local modularity optimization technique to maintain the community structure in a large dynamic graph. By using a coloring algorithm, [8] evaluates the nodes that certainly need to change their core numbers by inserting or deleting the node or edges in the graph. Core number for vertices in the graph can be found by using the method described in [9]. Finally, it updates the core number of such nodes by a linear algorithm. In addition to that it follows some pruning strategies in order to accelerate the algorithm by exploiting lower and upper bounds of the algorithm.

For detecting the community the existing algorithms do not scale well for large datasets[10],[11]. Since most of the algorithm follows a conventional learning process it takes lots of time for updating the model when a change occurs. Designing of such systems is costly since it need human intervention to set the parameters.

This section gave an overview about traditional methods used to find the stress state of a person. Section III will give a detailed description about the System Architecture and the phases in our proposed work. Section IV will discuss about experimental setup our work. Section V will explain about results. Section VI will explain about conclusion and future enhancements of our work. Section VII will explain about references of our work

ISSN: 2005-4238 IJAST

3. SYSTEM ARCHITECTURE

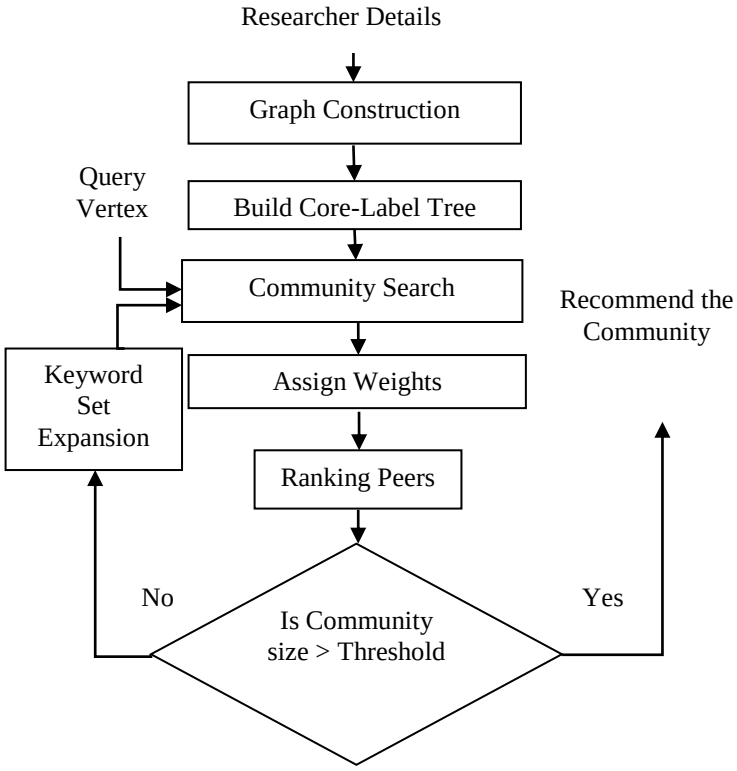

Figure 1: Architecture of the proposed system

3.1 Phases

The following phases are present in the proposed model.

1. Graph Construction
2. Building a Core Label(CL) tree
3. Community Search
4. Ranking the recommended peers
5. Keyword set expansion

1. Graph Construction

Initially the researcher details are provided as a JSON document. JSON is a lightweight data interchange format. It consists of a collection of name/value pairs. It is easy for humans to read and write and easy for machines to parse and generate. The input JSON record format is as follows.

```
[
{
id:
title:
authors:[{ name:, id:},{name: , id:},..]
year:
},
{},{},…
```

]

Each JSON record represents a publication's detail which contains a unique id for publication, name of the publication, name and id of author of those publications and published year. The article records in the JSON document are modeled into the attributed graph. It is the graph in which each and every node have some attributes which describe about that node. Each author forms a node in the graph. There exists an edge between two authors if and only if they have co-authored a publication. The attributes for each node in the graph are

Node_id - It represents the unique id of the node.
Author_id - It represents the unique id of the author.
Name - It represents the name of the author.
Keyword Set - It is the set of keywords which describe about the author.

The following figure shows the single node in the graph with attributes description.

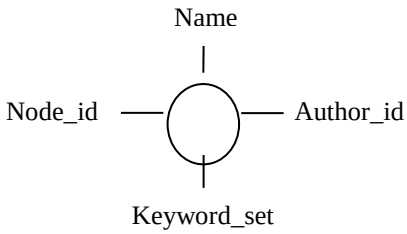

Figure 2: Single node in the attributed graph

The Node_id, Name and Author_id are directly taken from the JSON record, but for making keyword set words appearing in the title of the publication are used after doing some preprocessing step. The following steps carried out in pre-processing.

1.1 Tokenization

Tokenization is the process of dividing text into a set of meaningful pieces. These pieces are called tokens. The piece of text can be split into sentences or even into words.

Example Sentence:
Ontologies in HYDRA-Middleware2 for Better Intelligent Devices.

Tokenized sentence:
['Ontologies' , 'in' , 'HYDRA-Middleware2', 'for', 'Better' ,'Intelligent', 'Devices', '.']

1.2 Hyphen Removal

There is a lot of use of dash (-) to continue the sentences. Hyphen removal is the process of removing such dash between the sentences and break them into two words. The tokenized sentence is fed as input to this process.

Hyphen Removed sentence:
['Ontologies' , 'in' , 'HYDRA' , 'Middleware2', 'for', 'Better' ,'Intelligent', 'Devices', '.']

1.3 Stop words Removal

Filtering useless data is one of the major forms of preprocessing. In the processing of natural language, useless words (data) are called stop words. A stop word is a commonly used word such

ISSN: 2005-4238 IJAST

as 'the','a','an','in' etc., which do not convey any useful meaning so this type of word can be ignored.

Stop words Removed Sentence:
['Ontologies' , 'HYDRA' , 'Middleware2', 'Better' ,'Intelligent', 'Devices', '.']

1.4 Lemmatization

Lemmatization usually refers to doing things properly using a vocabulary and morphological analysis of words, usually aimed solely at removing inflection endings and returning the basic or dictionary form of a word known as the lemma. Lemmatization includes implicit case change so there is no need to do it as a separate process.

Lemmatized sentence
['ontologies', 'hydra', 'middleware2', 'good', 'intelligent', 'devices','.']

1.5 Digit and dot removal

This is the final step in preprocessing, removing digits and dots since it do not infer any meaning.

Final preprocessed sentence
['ontologies', 'hydra', 'middleware', 'ambient', 'intelligent', 'devices']

2. Core Label Tree Construction

In the next step, the attributed graph is converted into the form of a Core Label (CL) tree by using a core number of vertices of the graph in order to make searching efficient. A graph's k-core is the highest subgraph so each vertex has at least k degree. The k-shell is the set of vertices that belong to the k-core, it is contained in the (k+1) - core.

The following Figure shows the partition of the graph based on core numbers.

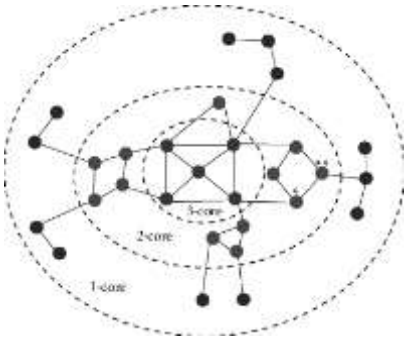

Figure 3: K-cores of the graph

After identifying core number for each node, Core Label (CL) tree can be constructed in which each node has some attributes that defines the nature of that node. Each Node of the Core-Label Tree has the following attributes

✓ coreNum - It is the core number of the k-core
✓ vertexSet - It is the set of vertices that form the node
✓ invertedList - It is a list of all unique keywords that appear as the attributes of the vertices in the

ISSN: 2005-4238 IJAST

vertexSet of that node. Each keyword is associated with a list of vertices that contain that particular keyword.

The following Figure shows the Core Label(CL) tree, in which the node attributes are explained. The node 3 has Core Number 3 and it includes the vertex set as A,B,C and D. The invertedList of node 3 has keyword 'Ontology' which is present in vertices A,B and C, 'Intelligence' keyword is present in B and C and the vertices C and D has 'Device' keyword.

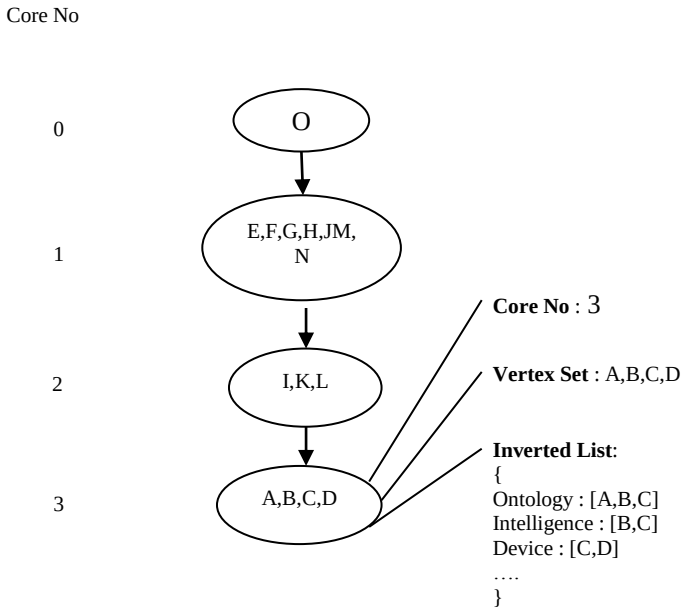

Figure 4: Core Label(CL) tree with attributes

3. Community Search

After attributed graph and CL tree has been built, by using an efficient community search algorithm the model aims to find the community that best match for given Query vertex q. The Community Search algorithm searches through the attributed graph part by part with the help of the CL tree by considering all possible keyword set combinations of the given query vertex's keyword set. At the end it returns the set of optimal keyword sets that satisfies all the predefined criteria. For the returned qualified keyword sets by the community search algorithm, assign each keyword of the keyword sets with some random weights. These weights are based on relevance and proximity measure.

Relevance is measured as the ratio of the number of years the writer worked for on that particular attribute to the total time he / she worked for. i.e.) It is the percentage of the total time the author has been active in a particular research area as per (1),

$$relevance(attr, t) = \frac{(t - first(attr) + 1)}{(t - time() + 1)} \qquad (1)$$

where,

- first(attr) is the year, the author first published for that attribute.
- time() is the year, the author gets his first publication.

- t being the current year.

The numerator part '$t - first(attr) + 1$' gives the period for which the author has been interested in attribute 'attr' and the denominator part 't−()+1' gives the total time for which the author has been publishing.

Proximity is calculated as the inverse of the time between the current year and the last year for which he / she published for that attribute. i.e.) It is the proximity between the latest publication as per (2),

$$proximity(attr, t) = \frac{1}{(t - latest(attr) + 1)} \qquad (2)$$

where,
- latest(attr) is the latest year, the attribute attr was seen in the author's publication.
- t being the current year.

The denominator part '$t - latest(attr) + 1$' gives the time interval after which the author has published for the attribute attr. By using relevance and proximity the weight for the particular attribute can be calculated as follows (3),

$$weight(attr, t) = \alpha * relevance(attr, t) + (1 - \alpha) * proximity(attr, t) \qquad (3)$$

The weight assigned to each entity is decided by α.The value of α would depend on the publication pattern of the author so it is different for each author. The value ranges between 0 to 1.

4. Ranking the recommended peers

By using the calculated weight for each keyword in the set of optimal keywords returned by community search algorithm, find the normalized score for each keyword set and rank the keyword sets according to the score. The normalized score can be calculated as per (4),

$$score(keywordSet_i) = \frac{\sum_{j=1}^{m}(keywordSet_i^j)}{m} \qquad (4)$$

It is the ratio of the cumulative sum of the weights of all the keywords appearing in the keyword set to the total number of keywords. appearing in that set. Here $keywordSet_i^j$ refers to the weight of jth attribute in the ith keywordSet.

The normalized score is found for all the keyword set returned by the community search algorithm. A highest normalized score indicates that the keyword set is the most relevant one i.e) optimal keyword set. If the number of matches between the identified optimal keyword set and the peer's keyword set is greater than or equal to two then add that peer to the optimal peer list. Finally, recommend the peers in the optimal peer list as an optimal community for the given query vertex q.

5. Keyword Set expansion

When the keyword set of given query vertex q is not sufficient to retrieve a community of suitable size [fixed threshold] the process of expanding the community set will take place. Expansion can be done using Google's pre-trained model known as 'word2vec'. It includes the closely related words that impart similar meaning.

ISSN: 2005-4238 IJAST

Words are interpreted as vectors and in the vector space it puts the words that are semantically identical. This is the concept of word embedding, which is basically the technique of numerical representation of words. The two forms of word2vec model are Continuous Bag Of Words(CBOW) and skipgram.

CBOW: From the context, the target word is predicted. The context is used as an observation as a consequence. This is the Word2Vec model's appropriate flavor for smaller data sets.

Skipgram: The basis of this technique is to infer the context from the target words. That pair of context-targets is viewed as a new observation. The Skip-Gram model is therefore ideal for larger data sets.

The CBOW and skip-gram models are trained to discriminate between noise and real words using logistic regression. CBOW is to train faster than skipgram. In addition, it is known that CBOW provides better accuracy for data where the words often appear, therefore CBOW is mostly used than skipgram. Both of the flavors rely on the probabilities of the context and the target words during their predictions.

For re-training of Google word2vec model the words appearing in the title of publication are used after some preprocessing steps such as tokenization, hyphen removal, stop words removal, lemmatization, digit and dot removal. After training, the model will output the top-N similar words for the given word with its probabilities.

The following Figure 5 shows the Google's word2vec model for predicting the most similar top-N words for the given word.

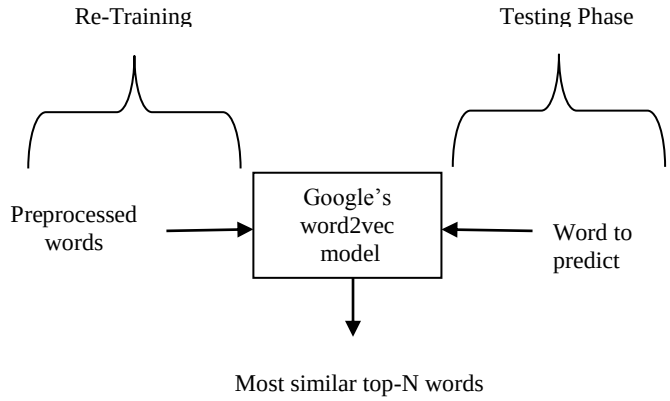

Figure 5: Google's word2vec model

The keyword set of the Query Vertex q is expanded by using the top-N similar words given by word2vec model and community search is repeated again. This process continues until the community of suitable size is reached.

4. EXPERIMENTS AND RESULTS

Citation Network Dataset is used for our experiments. It has several released versions in which DBLP-Citation-network V11 is used. It has records in JSON format. It was released on 05-May-2019 and it has about 4,107,340 papers and 36,624,464 citation relationships and the size of the dataset is about 10 GB. The citation data are extracted from the DBLP (DataBase systems and Logic Programming), ACM (Association for Computing Machinery), MAG (Microsoft Academic Graph), and other sources. We implement all our algorithms in python[12],[13],[14]. Further, we run all our experiments on a system with Intel quad-core i5 processor and 8GB of memory with Windows installed.

ISSN: 2005-4238 IJAST

The citation network dataset is in JSON format. Each record represents a single publication which is associated with publication id, title of the publication, list of authors who wrote the publication and the published year. By using attributes (id, title, authors, year) of the JSON record, attributed graph was constructed in which each node has attributes such as node_id, author_id, author_name and keyword set. After creating nodes, the edges are added between them if they have co-authored a publication. For our experiments we consider 10,000 publications which constructs about 22,914 nodes

The attributed graph is then converted into the form of a CL tree in order to make searching efficient in a complex graph. It arranges all vertices of the graph based on its core number. Each node of the CL tree has attributes such as coreNum which is the core number of that node,vertexSet which is the set of vertices forming that node and invertedList is the set of all unique keywords of the vertex in the vertex set of that node, it is a key-value pair where key is the keyword and value is the list of vertices having that key.

After attributed graph and CL tree is built, the search can be performed on the graph using an efficient community retrieval algorithm in order to find the optimal community for the given query vertex q. Searching is performed on the attributed graph using CL tree .i.e) level by level searching in the increasing order of core number takes place since breaking a larger network into smaller pieces gives better insight. The community search algorithm generates all possible keywords of query vertex keyword set and returns the list of optimal keyword sets.

Consider the sample graph described in the following figure 6,

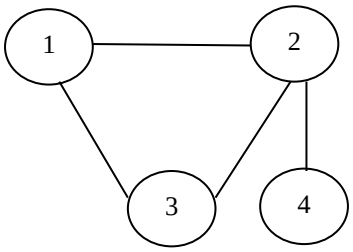

Figure 6: Attributed Graph

The keyword set of nodes 1 and 3 are [‘ontologies’, ‘hydra’, ‘middleware’, ‘ambient’, ‘intelligent’, ‘devices’] and node 4 having keyword set as [‘distributed’, ‘classification’, ’textual’, ‘document’,’grid’] and node 2 has [‘ontologies’, ‘hydra’, ‘middleware’, ‘ambient’, ‘intelligent’, ‘devices’, ‘distributed’, ‘classification’, ’textual’, ‘document’,’grid’]

If the user issues the query vertex q=2 means, the community search algorithm generates all possible keyword sets. From the list of generated keywords by the community search algorithm the model selects the best one by assigning weights to each and every keyword present in each keyword set. The weight assignment is based on relevance and proximity measure which is described above. From the calculated weight, the normalized score is calculated by adding all the weight value of the particular keyword set and divide it by the total number of keywords in the particular keyword set in order to bring it into a single value.

Using the calculated score the set of optimal keywords retrieved from the community search algorithm are ranked. The following Table 1 shows the sample keyword sets with it’s normalized score.

Optimal keywords	Normalized Score	Rank

[‘hydra’,’intelligent’,’middleware’,’ontology’]	0.816	1
[‘ambient’,’devices’,’intelligent’,’middleware’]	0.729	2
[‘ambient’,’hydra’,’middleware’,’ontologies’]	0.547	3
…	…	…
…	…	…

Table 1: Optimal keywords with normalized score

The highest normalized score indicates that the keyword set is more relevant, then the model recommend the peers who is having that keyword set. For the given query vertex q=2, 0.816 is the highest normalized score, the keyword set corresponding to that score are [‘hydra’,’intelligent’,’middleware’,’ontology’]. The peers having this keyword will be recommended to the user .i.e) The peer 1 and 3 will be recommended.

When the given query vertex keyword is not sufficient to retrieve the community of suitable size (threshold) then keyword set expansion takes place by using word2vec model. If the threshold size is fixed as 3, the peers 1 and 3 are not sufficient to recommend because the peer count becomes only 2 but the threshold value is 3 so the keyword set is expanded by imparting closely related words. After expansion, the keyword set of query vertex becomes
S = [‘ontologies’, ‘hydra’, ‘middleware’, ‘ambient’, ‘intelligent’, ‘devices’,‘distributed’, ‘classification’, ’textual’, ‘document’ , ’grid’ , ’division’,’analysis’,‘record’,’enterprise’,’apparatus’]

By using an expanded keyword set,community search , ranking , scoring process will again take place. Now the size of recommendation will get increased to 3 which satisfies the threshold value i.e.) the peers 1,3,4 will be recommended to the user.

To evaluate the performance of the proposed work, time taken to retrieve the community and accuracy of recommendation system are used.

i. **Time taken** is the time to be spent by the model when searching through the Graph using the CL tree to retrieve the optimal community. The following Table 2 describes the retrieval time for the sample of 10 nodes. The retrieval time varies for every query node and the average retrieval time can be calculated by summing up all node’s retrieval time and divide it by total number of nodes considered.

Node_id	Retrieval time(sec)
1	15
2	20
3	10
4	12
5	17
6	16
7	13
8	9
9	22
10	24

ISSN: 2005-4238 IJAST

Table 2: Retrieval time for sample nodes

Average retrieval time for the graph having sample of 10 nodes can be calculated as,

= (15+20+10+12+17+16+13+9+22+24) / 10
= 158 / 10
= 15.8 sec

The following Table 3 shows the comparison of average retrieval time of community from the graph having different number of nodes.

Publications	Total nodes created	Average time for retrieving the community
100	168	19.9 sec
1000	2221	29.7 sec
10000	22914	35.3 sec

Table 3: Retrieval time comparison for the graph having different set of nodes

ii. **Accuracy** is the percentage of number of correct predictions made by the model to the total number of input samples.

Consider the following comparsion Table 4 for actual result and model recommendations for the graph having 39 nodes, in which out of 39 nodes, the model made 33 correct predictions. Therefore the accuracy is,

$$\text{Accuracy} = \frac{33}{39} \times 100$$

$$= 84.6\%$$

Node_id	Actual community	Model recommended community
1	2,3	Null
2	1,3,4,5,6,7,8,9,10,16	1,3,4,5,6,7,8,9,10,16
3	1,2	Null
4	2,5,6	2,5,6
5	2,4,6	2,4,6
6	2,4,5	2,4,5
7	2	2
8	2	2
9	2,10,13,14,15,16,32,33,34,35	2,10,13,14,15,16,32,33,34,35
10	2,9	Null
11	-	-
12	-	-

ISSN: 2005-4238 IJAST

13	9,14,15	9,14,15
14	9,13,15,18,19,20,21	9,13,15,18,19,20,21
15	9,13,14,16,17	9,13,14,16,17
16	2,9,15,17,26,27,28,29,30,31,32	2,9,15,17,26,27,28,29,30,31,32
17	15,16,22,23,24,25	15,16,22,23,24,25
18	14,19,20,21	14,19,20,21
19	14,18,20,21	14,18,20,21
20	14,18,19,21	14,18,19,21
21	14,18,19,20	14,18,19,20
22	17,23,24,25	17,23,24,25
23	17,22,24,25	17,22,24,25
24	17,22,23,25	17,22,23,25
25	17,22,23,24	17,22,23,24
26	16,27,28,29,30,31	16,27,28,29,30,31
27	16,26,28	16,26,28
28	16,26,27	16,26,27
29	16,26,30,31	16,26,30,31
30	16,26,29,31	16,26,29,31
31	16,26,29,30	16,26,29,30
32	9,16,35	9,16,35
33	9,34	Null
34	9,33	Null
35	9,32	Null
36	37,38,39	37,38,39
37	36,38,39	36,38,39
38	36,37,39	36,37,39
39	36,37,38	36,37,38

Table 4: Comparison table for actual community and recommended community for the nodes in the graph

The following table 5 shows the accuracy obtained from the model for the graph having different set of nodes.

Total number of nodes	No. of correct predictions made	Obtained accuracy
39	33	84.6%
65	57	87.69%
127	112	88.19%

Table 5:.Accuracy comparison for the graph having different set of nodes

The table 5 shows that increase in number of nodes will increase the accuracy of the model. The model works well if it have large number of nodes. The following Figure 7 illustrates the comparison of accuracy for the graph having different set of nodes.

ISSN: 2005-4238 IJAST

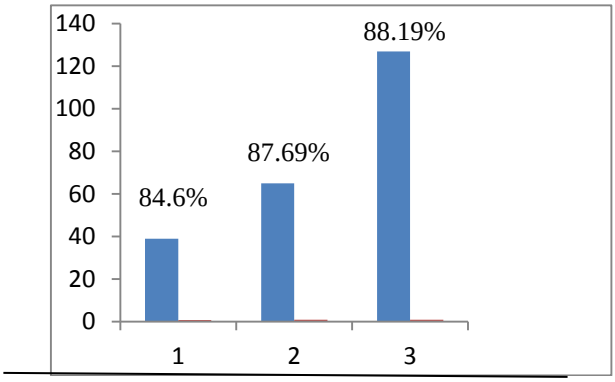

Figure 7: Accuracy comparison for the graph having different set of nodes

5. CONCLUSION AND FUTURE ENHANCEMENT

In this paper, an intelligent model for research peer recommendation is proposed. It introduces a efficient solution for the problem of research peer recommendation to the user. For better representing the researcher details, the model maintains their information in the form of attributed graph. It builds the Core Label(CL) tree efficiently and by using which optimal peers are recommend to the user. By introducing weights to the attributes of the node, it ensures the optimality in the recommendation. The experimental results obtained from the model reflects the real time view of the problem and it outperforms the existing models.

In this work, static graph is considered for community detection. But real world networks are not always static it is keep on evolving over time. So in future this work can be extended for dynamic graph where nodes and edges can be added or removed dynamically.

REFERENCES

1. Vivek Sourabh, C. Ravindranath Chowdary., "Peer recommendation in dynamic attributed graphs" Expert Systems with Applications., Elseveir. 2019, pp. 335–345.
2. Fang, Y. , Cheng, R. , Luo, S. , Hu, J. , & Huang, K.,"C-Explorer: Browsing communities in large graphs". Proceedings of the VLDB Endowment, 10 (12),2017,pp. 1885–1888.
3. Bello, G. A. , Harenberg, S. , Agrawal, A. , & Samatova, N. F., "Community detection in dynamic attributed graphs". In Advanced data mining and applications: 12th international conference, ADMA, Gold Coast, QLD, Australia,2016,pp.329–344
4. Fang, Y. , Cheng, R. , Li, X. , Luo, S. , & Hu, J., "Effective community search over large spatial graphs", Proceedings of the VLDB Endowment,2017,pp.709–720
5. Xin, Y., Xie, Z.-Q., & Yang, J., "An adaptive random walk sampling method on dynamic community detection" proceedings of the Expert Systems with Applications , 2016, pp.10–19.
6. Yixiang Fang Reynold Cheng Yankai Chen, Siqiang Luo Jiafeng Hu., "Effective and Efficient Attributed Community Search", The International Journal on Very Large Data Bases,2017,pp. 803-828.
7. Huang, X. , & Lakshmanan, L. V., "Attribute-driven community search"., Proceedings of the VLDB Endowment,2017,pp. 949–960.
8. Rong-Hua Li, Jeffrey Xu Yu, and Rui Mao.," Efficient Core Maintenance in Large Dynamic Graphs"., ieee transactions on knowledge and data engineering,2014,pp. 2453-2465.
9. Vladimir Batagelj, Matjaz Zaversnik.,"An O(m) Algorithm for Cores Decomposition of Networks"., Advances in Data Analysis and Classification,2011, pp.129-145.

10. Mario Cordeiro, Rui Portocarrero Sarmento, Joao Gama.,"Dynamic community detection in evolving networks using locality modularity optimization".,springer-Social Network analysis and mining,2016,vol.6
11. Rong-Hua Li, Jeffrey Xu Yu, and Rui Mao.," Efficient Core Maintenance in Large Dynamic Graphs"., ieee transactions on knowledge and data engineering,2014,vol.26,p.no.10
12. https://dblp.uni-trier.de/xml/
13. https://www.tutorialspoint.com/python/python_xml_processing
14. https://networkx.github.io/documentation/networkx-1.10/tutorial/tutorial.html

ISSN: 2005-4238 IJAST