Research Paper Recommendation System: SCAN Final Project Report

Sarthak Harne
IMT2020032

Chinthan Chandra
IMT2020109

Nilay Kamat
IMT2021096

Arhant Arora
IMT2020503

I. Introduction

We select the problem of building a research paper recommendation system for this project. The idea behind this project is that a user will be recommended research papers based on the papers they have previously read. For this, the system will be given a list of papers that the user has read and it gives out a list of recommended papers.

The motivation behind selecting this problem is to help a user explore a field of research, keeping them updated with the latest work, while also showing them seminal work that has come before. By using such a system, the user might also be recommended some work that lies at the intersection of the fields of research that the user has previously explored.

This problem is unique as unlike domains like social media and similar recommendation systems where we have a graph of interconnected users that can be used for recommendations, in the case of research paper recommendation systems, we instead have a graph of the items (papers) we want to recommend. This is the citation graph. This property of the data helps us use new ways of recommending items, which fall under Graph Based (GB) recommendation systems.

Unlike Collaborative Filtering (CF) methods where reviews/ratings of similar users are taken into account while recommending items, GB systems do not use any feedback from other users. GB systems are also different from Content-Based (CB) systems as the latter use the properties of the items/content of the items previously consumed by the users to further recommend newer items while GB methods instead use the relations between the items (obtained by the citation graph). As this graph indicates how the creators of these items perceive the items (by the citations and references), this gives us a new and different perspective on the items rather than just relying on other users or the item content.

Saying this, we still explore how the abstracts of these papers can be used for a CB recommendation system as this might help us recommend newer papers that might lie out of the immediate co-citation network of the previously read papers the content might still be semantically similar. This method will also help recommend newer papers and help with the cold start problem for such papers.

Finally, we try out a hybrid system by combining the GB and the CB system. This system combines the best of both worlds – the co-citation network context from the GB system and the semantic content context from the CB system.

For this system, we are assuming that this system will be used by a single user. As we explored later on, our proposed method works in real-time for each inference, due to this the same method can be duplicated for the other users.

II. Datasets

Since generating a new dataset on our own was beyond the scope of this project given the time and computation constraints, we used already available datasets and modified them to suit our needs. For the project, we used the following 2 available datasets:

A. ArXiv

This is a freely available, open pipeline on Kaggle to the machine-readable ArXiv dataset. It is a repository of 1.7 million papers, with a ton of relevant features like Paper ID, Authors, Title, Abstract, Category it belongs to, Update Date, etc.

The advantage of using this was that it is freely available. Thus we were able to access a large number of papers (1.7 million) with ease. It also covers a wide range of domains, which will help in making cross-domain recommendations.

While we do have a large number of papers available, we should note that only papers that are freely available and submitted to arXiv are available. Thus, many impactful papers submitted to other conferences might not be available. Also, as almost anyone can submit a paper on ArXiv, the quality of papers is not consistent.

The dataset can be found here: arXiv Dataset

B. ArXiv Internal Citation Graph

Thanks to Clement et al. who made this public, we are able to readily use the ArXiv Internal Citation Graph. The data is available in the form of a list, where we have a given paper and all the papers it cites:

root_paper − > [*referred_paper_1*, *referred_paper_2*, *referred_paper_3*, . . .]

An example entry in the dataset:
math/0312171 − > ['math/0310020' 'math/0211156' 'math/0301042' 'math/0212278']

Clement et al. used about $100 in AWS credits, more than 1 TB of space, and more than 400 core hours to extract this dataset. Thus the advantage for us is that it is now readily available to us and we do not have to extract it ourselves, which would be beyond our computing capabilities at the moment.

We should however also note that the data available is only till the year 2019 and that it contains only the internal ArXiv citations. This means that it comes with biases of the data, for eg. we would have more AI papers etc.

The dataset can be found here: ArXiv Internal Citation Graph

III. Exploratory Data Analysis

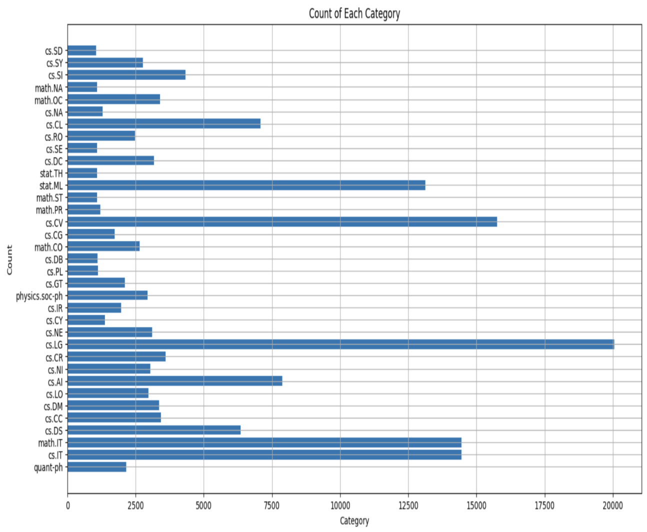


Fig. 1: Categories Plot

The dataset that was selected had barely any numerical or categorical columns. So, we performed EDA wherever possible. One interesting observation we made was that there was inherent bias towards particular categories.

IV. Data Processing

After acquiring and analyzing the data, the next step was to process it. Based on our needs, we performed the following steps:

A. Considering Only CS Papers

From the vast dataset, we took only those papers that belonged to the CS domain. This includes various subdomains like cs.AI, cs.LG, cs.CV etc. We did this for the following 2 main reasons:

- It reduced the data size significantly (from 1.7 Million $->$ 610k), which made it easier for us to handle the data.
- Since we have some understanding of the domain ourselves, we thought we would be able to assess the CS paper recommendations ourselves and evaluate how good the recommendations are.

B. Creating the Citation Graph

We then proceeded to create a citation graph based on the data of CS research papers we now had. We created the following graphs:

- Undirected Graph - We added an edge (Paper A - Paper B) if paper A cited or was cited by paper B.
- Directed Acyclic Graph (DAG):
  - Reference graph - We added an edge (Paper A $->$ Paper B) if A referred B.
  - Citation graph - We added an edge (Paper A $->$ Paper B) if A was cited by B.

Creating these graphs is important as it helps to highlight the *importance* of a node (a research paper) which we might defined based on its degree (in-degree, out-degree or both).

Another advantage with this was that it further reduced the number of nodes to 90268.

The case of Mulmuley and Briand et al.: During the process, we came across an interesting finding: that of cycles in the raw citation graph as seen in the case of Mulmuley and Briand et al.

The case was as follows:

- Mulmuley published a book with arxiv id 0704.0229.
- Briand et al. suggested a correction for the same in a paper with arxiv id 0810.3163 citing Mulmuley's book 0704.0229.
- Mulmuley updated his book, citing Briand et al.'s 0810.3163, creating a cycle in the graph.

The existence of such a cycle is a bane to us, and thus we removed the second type of link across our graph, thereby eliminating these cycles and further reducing the number of nodes to 88281. Fig 2. shows a visualization of a subset of our citation graph.

C. Abstract to Word Embeddings

Among other useful information, our dataset contained the abstract of all papers which we planned to use to good advantage. We thus converted these abstracts to word embeddings using BERT-like models:

- RoBERTa (Liu et al.) - This only recognises words based on general English language data.
- AllenNLP/SciBERT (Beltagy et al.) - Self supervised (BERT like) training done on scientific texts (expanded vocabulary and a better context recognition for scientific literature).

Using these, we received 768 dimension embeddings for each paper.

V. Algorithms Implemented/Used

Before we see the different algorithms we tried to build our recommendation system, let us define some notations:

i) Made for a single user u.
ii) Set of papers: $P = \{p_i \, i \in N\}$.
iii) Set of papers read by user u:
$P_u = \{p_i^u : i \in N\}, P_u \subseteq P$.

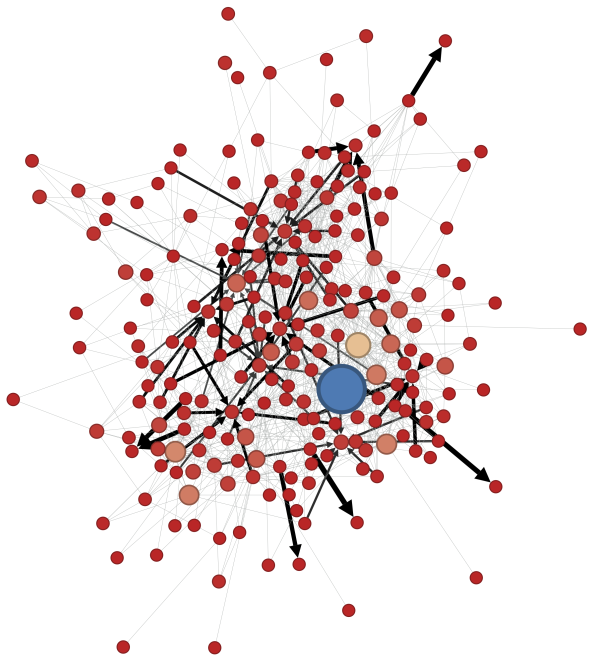


Fig. 2: A subset of our citation graph, plotted using Gephi.

iii) Recommendation System R: $P_u \times P \rightarrow P \setminus P_u$.

The various algorithms/approaches we tried and what we discovered in each of them are explained below:

A. Page Rank [1]

This algorithm uses the fact that if there are more in-links to a particular node, the greater its priority. This can be perfectly used to help us in recommending papers. If a particular paper has more citations, it means that the paper has a significant impact. We get the Markov chain-based transition probabilities and stationary probability over all papers. Page Rank Based Recommendation:

- Select a paper from P_u.
- Take k random walk steps and receive a next paper probability distribution.
- Repeat for all papers from P_u.
- Recommend papers with maximum probability/based on sampling.

So this recommends papers that are before the paper being selected but we also need to recommend papers that are recent in time. The figure 3 shows the same

Initial intuition would be just to reverse the graph. The effect of this would destroy the graph as it would change the principle of Page Rank: more in-links, more priority. There is also the possibility of getting stuck in sinks. Now, we need to find the previous most probable state. Some math to explain it:

A k step random walk starting from paper 4 would be, given the transition matrix is M_t and stationary probability of paper 4 is p_4:

$$\vec{n} = M_t^k \cdot \vec{p_4}.$$

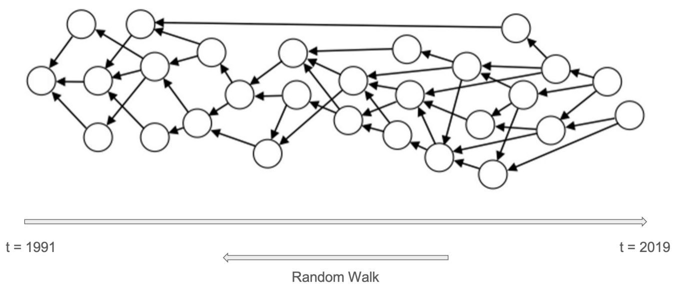


Fig. 3: Page Rank

So, for the probabilities of the previous state, we can calculate them using (Reverse MC traversal):

$$\vec{prev} = (\vec{p_4}^T \cdot M_t) \otimes \pi^T$$

We use this to recommend a set of papers that are a combination of papers before and after the paper that the user has read.

1) Addition of "Beam Search": The random walk and reverse MC traversal end up giving nodes in the vicinity of the starting node. This doesn't allow the implementation to recommend papers that are far away. To overcome this, using the principles of beam search we add a controllable parameter that can be used to move through the graph. A brief example of the same:

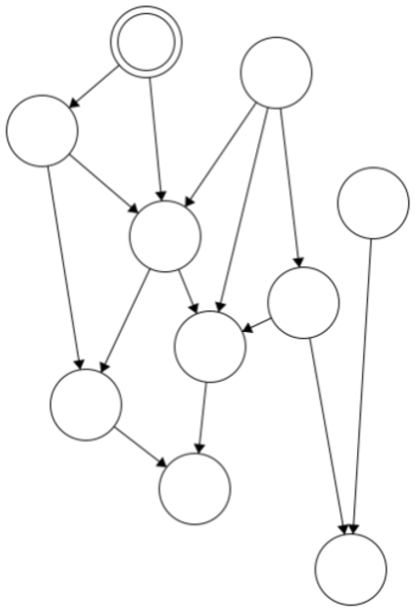
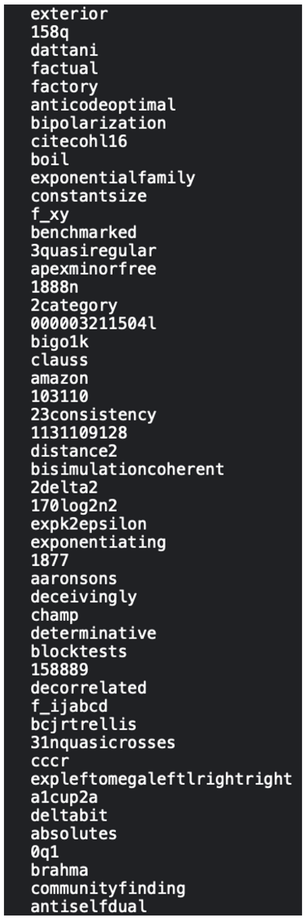


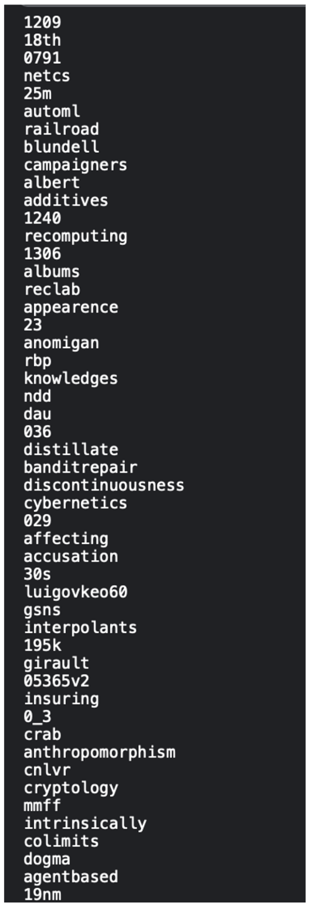
Fig. 4: "Beam Search" principle-based graph exploration.

Suppose we want to recommend 10 papers and the number of beams we want to have is 2. So in the first iteration, we choose the 2 most probable nodes, perform an iteration of random walk on these 2 nodes, choose the 2 most probable nodes from these two nodes, and continue this process until we get the required number of nodes, i.e 10 in this case. 4 shows an example of the exploration method.



```
exterior
158q
dattani
factual
factory
anticodeoptimal
bipolarization
citecohl16
boil
exponentialfamily
constantsize
f_xy
benchmarked
3quasiregular
apexminorfree
1888n
2category
000003211504l
bigo1k
clauss
amazon
103110
23consistency
1131109128
distance2
bisimulationcoherent
2delta2
170log2n2
expk2epsilon
exponentiating
1877
aaronsons
deceivingly
champ
determinative
blocktests
158889
decorrelated
f_ijabcd
bcjrtrellis
31nquasicrosses
cccr
expleftomegaleftlrightright
a1cup2a
deltabit
absolutes
0q1
brahma
communityfinding
antiselfdual
```

(a) Words from the topic with the most number of papers.



```
1209
18th
0791
netcs
25m
automl
railroad
blundell
campaigners
albert
additives
1240
recomputing
1306
albums
reclab
appearence
23
anomigan
rbp
knowledges
ndd
dau
036
distillate
banditrepair
discontinuousness
cybernetics
029
affecting
accusation
30s
luigovkeo60
gsns
interpolants
195k
girault
05365v2
insuring
0_3
crab
anthropomorphism
cnlvr
cryptology
mmff
intrinsically
colimits
dogma
agentbased
19nm
```

(b) Words from the topic with the second most number of papers.

Fig. 5: Words in LDA.

B. Cosine Similarity

In this, we used a simple approach of getting the cosine similarity scores of the papers read by the user with all other papers the user has not read. From the scores obtained, we recommend the top n papers from this, i.e. the papers with the highest cosine similarity scores. We used two methods to obtain embeddings for the abstracts, the results obtained in each method are also given below,

i) Embeddings using RoBERTa: Results were bad as specific context.
ii) Embeddings using SciBERT: Good results.

C. LDA [2]

In this method, we first obtained the TF-IDF vectors of the abstracts then applied LDA to these abstracts. We kept the number of topics as 20. After this, we chose the most popular topics and then took the average of each element of the TF-IDF vectors of this topic and saw the top-rated words in this. These were gibberish, so we dropped experimenting with this method.

The figure 5, shows the words obtained.

D. Hybrid: Page Rank + Cosine Similarity

After seeing the results, we decided that a hybrid approach of Page Rank and Cosine similarity would be a good one. In this, we take the weighted sum of normalized Page Rank scores and Cosine Similarity Scores. The result is controlled using a parameter τ. The final formula for the same is,

$$S = \tau \cdot PR + (1 - \tau) \cdot CS$$

where PR is Page Rank score and CS is Cosine Similarity score.

The steps to reach this are:

- Obtain recommendations using Page Rank + Beam Search and normalize these probabilities.
- Obtain recommendations using Cosine Similarity and normalize these values.
- Combine the recommendation using these 2 and hyper-parameter τ.
- The Result obtained consists of the top n research papers.

VI. Evaluation

Zhang et al. [3] had surveyed the existing scholarly recommendation systems. In their survey, we observed that there was no concrete evaluation metric and the metric was completely dependent on the dataset and the methodology the authors had used.

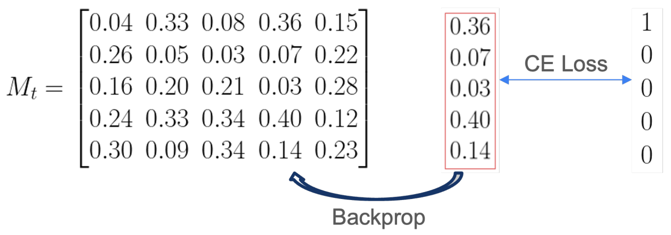
We noticed that many papers had used manual evaluation i.e user studies or had utilized datasets with relevant user data for evaluation.

Given the time and resource constraints, we could not conduct such surveys. Therefore, we manually evaluated the recommendations and assessed their quality based on our subjective judgment.

We checked the results for a few papers that we had come across during our coursework. The recommendations made followed a logical flow of what should be recommended next, and thus we inferred that the recommendations were indeed good.

VII. Future Work

The methods we have used work well as an offline recommendation system with no updates to the weights of the papers recommended, i.e. no personalization.



$$M_t = \begin{bmatrix} 0.04 & 0.33 & 0.08 & 0.36 & 0.15 \\ 0.26 & 0.05 & 0.03 & 0.07 & 0.22 \\ 0.16 & 0.20 & 0.21 & 0.03 & 0.28 \\ 0.24 & 0.33 & 0.34 & 0.40 & 0.12 \\ 0.30 & 0.09 & 0.34 & 0.14 & 0.23 \end{bmatrix} \quad \begin{bmatrix} 0.36 \\ 0.07 \\ 0.03 \\ 0.40 \\ 0.14 \end{bmatrix} \xleftrightarrow{\text{CE Loss}} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Fig. 6: Future Improvements

To overcome this, we could use the probabilities of the transition matrix as an embedding of the current paper, after recommending the papers we can use cross-entropy loss to backpropagate the error can update the embeddings. This is similar to using contextual bandits, here the context is the user's history of papers read and rated. Fig 6, shows the same.

References

[1] L. Page *et al.*, "The PageRank Citation Ranking: Bringing Order to the Web," Proceedings of the 7th International World Wide Web Conference, Brisbane, Australia, 1998.

[2] Blei, David M., Andrew Y. Ng, and Michael I. Jordan. "Latent Dirichlet Allocation." Journal of Machine Learning Research 3.Jan (2003): 993-1022.

[3] Zhang, Zitong & Patra, Braja & Yaseen, Ashraf & Zhu, Jie & Sabharwal, Rachit & Roberts, Kirk & Cao, Tru & Wu, Hulin. (2023). Scholarly recommendation systems: a literature survey. Knowledge and Information Systems. 65. 1-46. 10.1007/s10115-023-01901-x.