A Recommendation System for Research Papers

Akash Ponduru; Chaitanya Krishna Kommineni; Nagi Revanth Avulapati; Anand Suresh

In response to the excessive abundance of academic publications, we suggest a novel method that mixes graph theory, machine learning, and topic modeling to expedite paper navigation and recommendations. Using metadata from large research corpora, our system creates a knowledge network that reveals detailed links between authors, and topics. Advanced graph theory algorithms extract relevant insights, whereas machine learning approaches create individualized recommendations based on user preferences. Furthermore, topic modeling improves content structure, ensuring quick navigation and finding of relevant publications. This unified approach provides a dynamic and user-centric solution, relieving the constraints provided by the vast volume of academic literature and boosting multidisciplinary collaboration in research communities.

INTRODUCTION

The sheer abundance of information and internet access has made reading not simpler, but more difficult to manage, particularly for newcomers. Think of yourself as a newly admitted college student in computer science, physics, chemistry, or any other domain; How one can navigate the massive amount of literature, academic papers. to find quality materials to read - and to read them in a systematic manner. This results in a huge challenge. We propose a unique strategy that uses graph theory, machine learning, and subject modeling to speed up paper navigation and recommendations.[1].

PROJECT OVERVIEW

There are two basic starting approaches to recommendation systems that exclude any deep learning-based or matrix factorization-based methods to recommend items:

Content Based Filtering

Content-based filtering proposes items to users based on the item descriptions or content as well as the users' specific preferences. Each item is represented by a collection of descriptors. The system then computes the metric similarity between the user item features and recommends things with similar properties. Although content-based filtering is useful for dealing with the cold start problem and delivering transparent recommendations, it may struggle to capture various user interests. To improve suggestion accuracy, hybrid techniques that combine content-based and other filtering techniques are frequently used [2].

Collaborative Filtering

Collaborative filtering only makes recommendations for new items based on previous interactions between users and items. The characteristics of each individual item are not taken into account.

The historical data of the user interacting with the items is captured and saved in collaborative filtering. A matrix called a user-item interaction matrix is typically used to depict this, with rows standing in for users and columns for the items. When recommending a user to the target, similar users are grouped and their interactions are taken into account [2].

Hybrid approach

Apart from the above two we have worked on a Hybrid approach which is an enhanced form of content-based filtering where we use features from content-based filtering and introduce new features from graphs.

DATA OVERVIEW

The dataset is taken from "Citation Network Dataset"[3] and we have used version 10 of this dataset which has more than 3 million papers and more than 25 million citation relationships and it was last updated on (2017-10-27)

Field Name	Field Type	Description	Example
id	string	paper ID	013ea675-bb58-42f8-a423-f5534546b2b1
title	string	paper title	Prediction of consensus binding mode geometries for related chemical series of positive allosteric modulators of adenosine and muscarinic acetylcholine receptors
authors	list of strings	paper authors	["Leon A. Sakkal", "Kyle Z. Rajkowski", "Roger S. Armen"]
venue	string	paper venue	Journal of Computational Chemistry
year	int	published year	2017
n_citation	int	citation number	0
references	list of strings	citing papers' ID	["4f4f200c-0764-4fef-9718-b8bccf303dba", "aa699fbf-fabe-40e4-bd68-46eaf333f7b1"]
abtract	string	abstract	This paper studies ...

Citation information is derived from ACM, MAG (Microsoft Academic Graph), DBLP etc. and other databases. Each paper has an abstract, authors, year, venue, and title.

The original dataset is divided into 4 parts and every part is a JSON file. In each JSON file, each line represents a research paper, which is in JSON schema.

DATA PREPROCESSING

Based on the features in our dataset, we have string-based features which we want to convert into numeric vector format, for that we are going to perform a few steps which will help us convert our documents into numeric vector format.

Let's take an example string to see how these steps will convert an input document to vectors.

Creating Content Column

The brief thematic information from the "Title" column and the thorough synopsis from the "Abstract" column were combined to create the "Content" column.

Converting String to Lowercase

By changing the text to lowercase, we can guarantee that words have the same case, which keeps the machine from making judgment calls based on case sensitivity. This standardization eliminates differences between words based on capitalization or lowercase, making comparisons easier. It would, for example, treat "Ball" and "ball" as the same word.

Removing Stop words

Stop words from the lowercase "Content" column data have been extracted, including common terms like "a," "the," and others. Removing these frequently occurring words from content comparison and analysis, this guarantees a refined dataset that allows for precise comparisons between research publications.

Performing Lemmatization

The words in the "Content" column data have undergone lemmatization, which reduces them to their canonical or base forms. By returning words to their root forms, this process helps to standardize vocabulary and makes textual content analysis and comprehension more efficient.

Tokenize the documents

Tokenization is the process of creating tokens for each word by dissecting the sentence "The cat sat on the mat" into its component parts. As a result, a list of distinct words is created, which serves as the foundation for further processing or analysis.

Creation of a Dictionary for the tokenized documents

The tokenized documents have been converted into a dictionary that associates each distinct token with a unique number. This mapping gives each distinct word token found in the documents a unique index, making it easier to retrieve and manipulate textual data efficiently.

Bag-of-words representation

A document's bag-of-words representation captures word occurrences as tuples, where each tuple is made up of the dictionary word's index and the document word's frequency.

Here is an example showcasing how a document would undergo changes during the above preprocessing steps:

Input Sentence:	*This is an example to demonstrate the different string preprocessing techniques.*
After converting to lowercase:	*this is an example to demonstrate the different string preprocessing techniques.*
After removing stop words:	*example demonstrate different string preprocessing techniques.*
After lemmatization:	*example demonstrate differ string preprocess technique.*
After Tokenization:	*["example", "demonstrate", "differ", "string", "preprocess", "technique"]*
After Dictionary:	*{"example": 0,"demonstrate": 1,"differ":2 , "string": 3 , "preprocess": 4 , "technique": 5}*
After Bag of Words:	*[(0, 1), (1, 1), (2, 1), (3, 1), (4, 1), (5, 1)]*

Here we have a graphical representation of word frequencies in our dataset after lemmatization.

LATENT DIRICHLET ALLOCATION

Using the bag-of words we are now clustering the documents based on the specific topic relevant to each document. This is called topic modeling and can be done by Latent Dirichlet allocation technique. This method generates topics based on word frequency from the set of documents. It's very powerful when you have a mix of topics within documents[4].

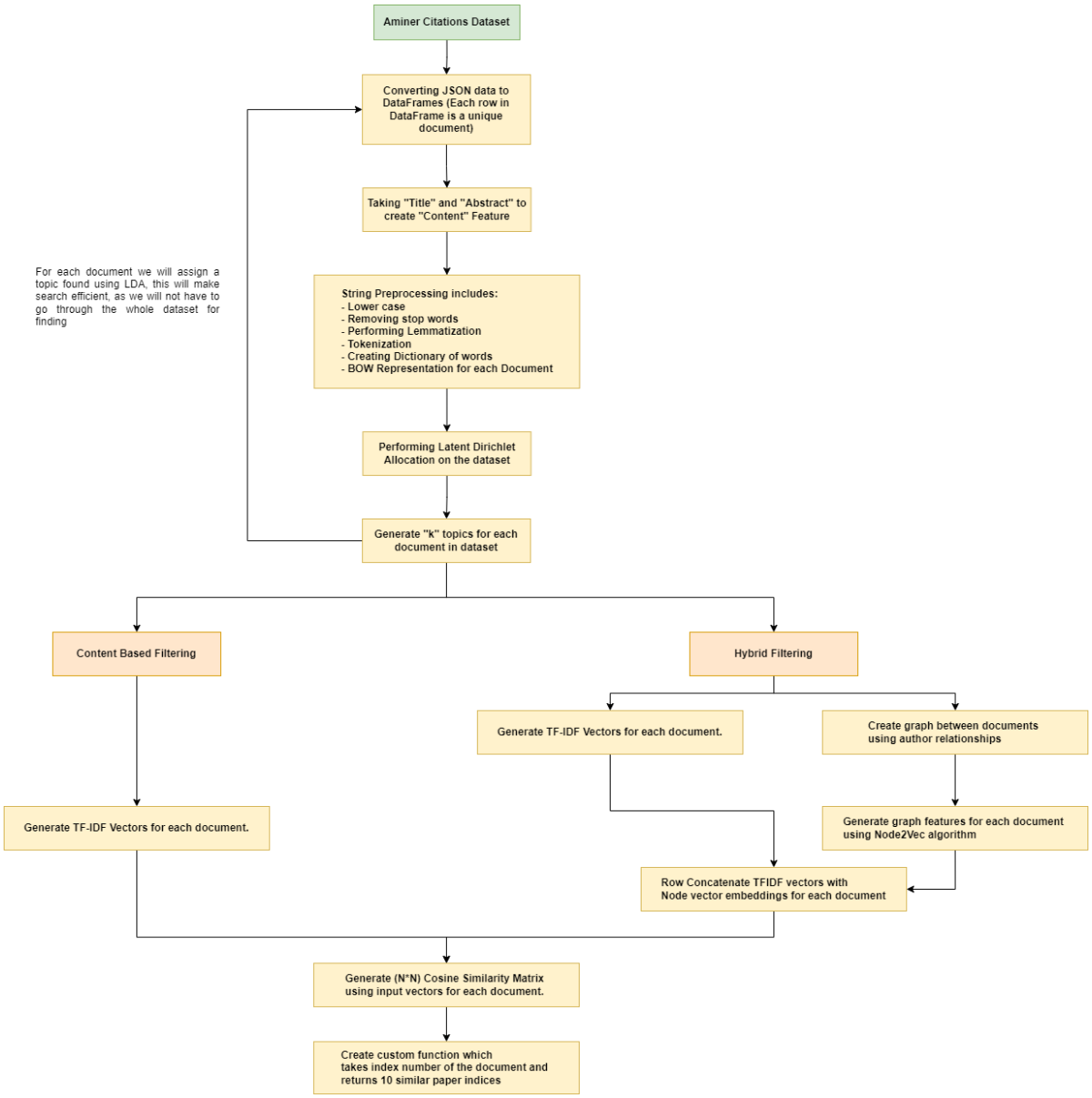
There are three major steps we take to build our LDA model:

1. Calculate the probability of a topic k given a document i.
2. Calculate the probability of a word j given a topic k.
3. Calculate the probability of a word j given topic k and document i by multiplying the probabilities from step 1 and step 2.

```
[(0,
  '0.003*"learning" + 0.003*"model" + 0.003*"1" + 0.003*"data" + 0.003*"using" '
  '+ 0.002*"set" + 0.002*"x" + 0.002*"algorithm" + 0.002*"function" + '
  '0.002*"one"'),
 (1,
  '0.003*"model" + 0.003*"learning" + 0.003*"algorithm" + 0.003*"one" + '
  '0.003*"set" + 0.003*"function" + 0.002*"data" + 0.002*"two" + 0.002*"using" '
  '+ 0.002*"1"'),
 (2,
  '0.004*"model" + 0.003*"learning" + 0.003*"1" + 0.003*"using" + 0.003*"n" + '
  '0.002*"set" + 0.002*"data" + 0.002*"2" + 0.002*"algorithm" + '
  '0.002*"number"'),
 (3,
  '0.003*"data" + 0.003*"learning" + 0.003*"algorithm" + 0.003*"using" + '
  '0.003*"model" + 0.002*"set" + 0.002*"two" + 0.002*"1" + 0.002*"x" + '
  '0.002*"function"')]
```

The above image shows a small LDA model training on a small subset of citations dataset, for every tuple in the list we have the topic index and set of words which describe that topic. LDA helps us understand the segregation of diverse topics in our dataset and will be used to recommend papers based on keywords provided by the user.

METHODOLOGY



CONTENT BASED FILTERING

Once we perform our string preprocessing and Latent Dirichlet Allocation of topics to all the documents in our dataset, we want to recommend academic papers based on the user's interest or through keyword inputs from the user. To achieve this for string-based dataset, we can proceed with the approach of vectorization and finding similarities between data, which would facilitate us to recommend. Vectorization is to create vectors for each document in our dataset. These n dimensional vectors will help us locate the positioning of each document in n dimensional hyperspace. Documents which are closer together in the hyperspace will be considered as similar to each other[2].

Term Frequency -Inverse Document Frequency

There are various ways to create these embeddings or vectors some of which are Count based vectors, Term Frequency – Inverse Document Frequency based vectors (TF-IDF), Word2Vec, etc., For our exercise, we will be using TF-IDF vectors as it measures the importance of a word within a document relative to all of documents. Words within the text corpus of research paper documents are transformed into relevance numbers and stores in the form of vectors [5].

The TF-IDF vectors are calculated using below formula:

$$Term\ Frequency(Word_i, Document_j) = \frac{Count\ of\ word\ i\ in\ document\ j}{Total\ number\ of\ words}$$

$$Inverse\ Document\ Frequency(Word_i, Document_c) = \log(\frac{N}{n_i});\ n_i \leqslant N$$

$$N = Number\ of\ Documents$$
$$D_c = Set\ of\ all\ Documents$$
$$n_i = Number\ of\ documents\ which\ has\ word\ i\ in\ the\ corpus$$

$$TFIDF = Term\ Frequency(Word_i, Document_j) * Inverse\ Document\ Frequency(Word_i, Document_c)$$

Once TFIDF is done, we get relevance score for each word in the document as vector as below:

	15	96	308	317	601	777	854	920	954	1033	1040
0	0.097773	0.112955	0.089403	0.114547	0.169919	0.064428	0.18738	0.156119	0.131297	0.09987	0.101894

Why TF-IDF:

We used TF-IDF because if we only use Term Frequency, then it creates vectors with probability values of words in the documents. It gives more importance to rare words while penalizing the words which occur frequently, for example stop words. Easy to train with a lot of hyperparameters to control and are computationally stable.

Challenges in TF-IDF:

The downside of using TF-IDF vectors is that vectors do not take account of semantic meaning of words and consider them as different words with different meanings. This was already taken care of by lemmatization in string processing.

Cosine Similarity

Once we have our vectors for all the documents, we can find the similarity between these documents using Cosine similarity using the formula:

$$Cos\Theta = \frac{X_1.X_2}{\|X_1\| \cdot \|X_2\|}$$

The resultant matrix will be of shape (D, D).

- Where D is the number of documents in the subsample dataset.

- Each cell will contain a value between -1 and 1, where -1 means the entities are very dissimilar and 1 means very similar.

The resultant matrix after cosine similarity is shown below:

```
array([[1.        , 0.34036323, 0.47237545, ..., 0.55493331, 0.42708894,
        0.52434814],
       [0.34036323, 1.        , 0.38698448, ..., 0.54442469, 0.48837322,
        0.40985639],
       [0.47237545, 0.38698448, 1.        , ..., 0.42355572, 0.2839023 ,
        0.40474755],
       ...,
       [0.55493331, 0.54442469, 0.42355572, ..., 1.        , 0.27306605,
        0.60067133],
       [0.42708894, 0.48837322, 0.2839023 , ..., 0.27306605, 1.        ,
        0.30251262],
       [0.52434814, 0.40985639, 0.40474755, ..., 0.60067133, 0.30251262,
        1.        ]])
```

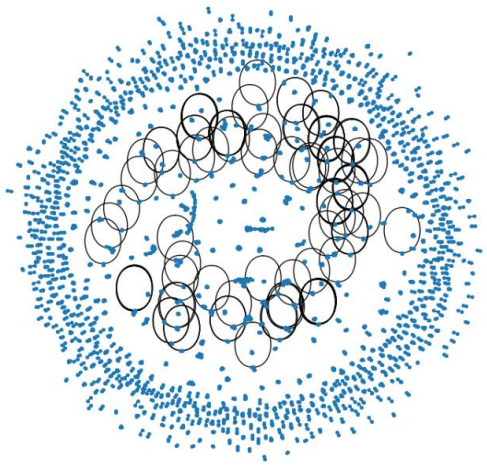
Why Cosine similarity:

The Cosine similarity measures how alike two objects or entities are as opposed to distance which measures how far or how close two entities are. The minimum distance between 2 entities could be 0 and maximum could be infinite. The cosine similarity ranges from -1 to 1, which can be used to find out the documents which are similar to each other. Cosine similarities does not rely on distances between entities in hyperspace but utilizes the change in degrees from origin to understand the positions, which makes it very reliable in high dimensional spaces and very easy to compute[6].

Challenges in Cosine similarity:

Given a document, with the help of a cosine similarity matrix, we can find other documents which are similar to the given document. But if there is no initial document, we cannot find the similarities. This problem is called cold start. To tackle this issue, we take inputs from users, and sort the documents which are similar to the user's inputs using topic index from Latent Dirichlet allocation. Using these documents, we are able to recommend other similar documents with the help of user's inputs.

Graph between Authors and Co-Authors in the sub-set of Aminer DBLP-Citation-Network-V10 dataset



HYBRID APPROACH

The idea behind hybrid approach is to enhance the existing capabilities of content-based filtering by using other features present in the dataset. While looking at the relationship between authors within our dataset, we found that many authors who write research papers in similar domains tend to be connected with each other. We can see the relationship graphically in the below graph.

The graphical representation between authors is generated on a subsample of citations dataset. Different authors in the dataset are represented using blue dots and the connection between them is represented by black lines.

We can see that many authors are forming groups in the small sample of data, we are assuming that this relationship between authors is across the entire dataset. We implemented Node2Vec algorithm on the authors graph to generate a document embedding based on the authors of the research papers.

Node2Vec

Node2Vec is a method for producing graph node embeddings. It explores small regions using controlled random walks, allowing the algorithm to strike a balance between collecting network structure and semantic information. Users can alter the exploration behavior during random walks by modifying parameters such as p and q. The program then employs Word2Vec to learn vector representations for nodes in the random walks based on their context, arranging nodes with comparable structures in a continuous vector space. The generated embeddings may be applied to a variety of graph-based tasks, including link prediction and node categorization[7].

PROJECT IMPLEMENTATION

(Image 1)

(Image 2)

The above Image 1 and Image 2 show the project outcome which is a web application. The system cannot recommend research papers for users since it does not store any personal and historical information to identify users. Instead, the system asks users for their preferences via keywords and shows an initial list of 10 papers based on the keywords. The initial list is based on the results from the LDA model.

Once a user gets the initial list, they can take any of the document indexes and use that as a starting point. They can search for related papers based on the search.

Image 1 shows results from Cold Search while Image 2 shows results from document search based on the initial document index search.

EVALUATION METHODS

A/B TESTING

A/B testing, is a technique in which two versions (A and B) of a webpage, app, or campaign are compared to determine which version is better at attaining particular goals. Without their knowledge, randomized users are exposed to either the control (original) or variation (altered), and data on important metrics are collected. The important variations in performance are then determined using statistical analysis, driving data-driven decision-making. In marketing and product development, this iterative method is extensively utilized to improve user engagement, conversion rates, and overall effectiveness[8].

RESULTS AND CONCLUSION

We started with the idea to create a system which will help navigate people through different reading materials, for this project we specifically focused on research papers. During the process we learned about different string preprocessing, topic modeling using Latent Dirichlet Allocation model, and how to use graph theory to augment new features using Node2Vec. We were able to identify the different concepts taught in class and how they relate to a real-world use case apart from that we also learned a few new concepts like Cold Start Problem, Vector Embeddings, Natural Language Processing techniques. We were able to successfully create a web application which acts as an interface between user and the system which helps people search for research papers in an effective way based on their preferences. There are a few sets of challenges that we faced throughout the process and how are we planning to work on them in future.

CHALLENGES AND FUTURE PROSPECTS

Our methods' implementation presented a few difficulties along with the steps to make necessary improvements, chief among them being computational limitations, dataset representation, scalability, and the lack of user interaction data.

Computational Power Restrictions

Vector-based algebra, which is essential to our methods, requires a significant amount of processing power. Because of this restriction, we were only able to use a small portion of the dataset and were unable to apply these approaches to the full dataset due to the high computational cost. Rather than relying on standalone systems for our processing pipeline, we could leverage large-scale data analytics engines like Apache Spark for more efficient data handling.

Sampling and Unsupervised Methods

We cannot be certain that our subsampled dataset accurately represents every original topic. Determining whether our sample sufficiently covers the wide range of topics from the original dataset is still a challenge because our methods are unsupervised. This issue would be resolved automatically if we employ Apache Spark or similar engines.

Reproducibility on Large Datasets

Although our findings hold true when applied to datasets of comparable size, scaling up to larger datasets may produce different results. The computational complexity of the methods may cause significant variation in their performance and results when applied to large datasets. We're interested in exploring other mathematical approaches that use features differently. We plan to implement these and assess any improvements in the results.

User Interaction Data and Collaboration Filtering

Our ability to apply collaborative filtering techniques was restricted by the absence of historical user interaction data. This problem might be solved by storing users' past searches and incorporating a feedback system into the application, which would allow user ratings to improve collaboration filtering efficiency in subsequent versions.

These difficulties draw attention to the necessity of representative sampling, scalability concerns, effective computational resources, and user interaction data for more effective and reliable recommendation system implementation on bigger and more varied datasets.

REFERENCES

[1] https://www.buffalo.edu/eln/students/project-portal/host-page.host.html/content/shared/www/eln/project-portal/project-profiles/active-projects/northstar-ai-pathfinding-readers.detail.html
[2] https://www.turing.com/kb/content-based-filtering-in-recommender-systems
[3] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. ArnetMiner: Extraction and Mining of Academic Social Networks. In Proceedings of the Fourteenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'2008). pp.990-998. [PDF] [Slides] [System] [API]
[4] https://youtu.be/VsjYyxd_LaU
[5] https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency
[6] https://en.wikipedia.org/wiki/Cosine_similarity
[7] https://towardsdatascience.com/node2vec-explained-db86a319e9ab
[8] https://www.optimizely.com/optimization-glossary/ab-testing/#:~:text=A%2FB%20testing%20