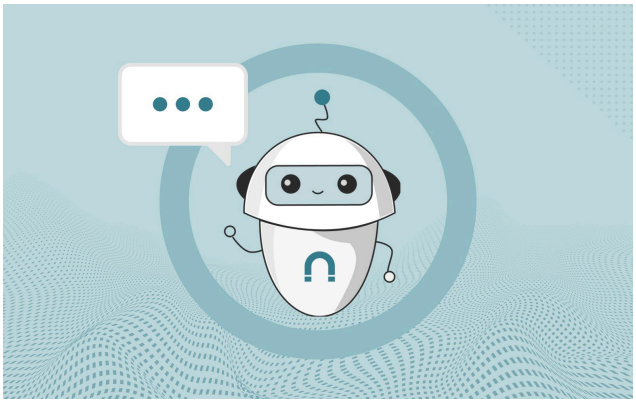
Foundations of Data Science Project

Dr. Khalaj
Fall 2024

Research Assistant

January 2025



Overview

You are meant to develop an AI assistant that can help a user find relevant papers. You will be given a dataset of scholarly articles up to the year 2017. You will try to infer useful information about the articles and use this information to enhance your AI assistant's inference. More data will be scraped from internet resources. The data you have crawled will be used as testing data throughout the course of the project.

You are required to provide your code for this project in the courseware alongside the report. Moreover, both of these files should be uploaded to **GitHub** using Git. You must add us as collaborators to your repository, as it will make monitoring/grading your results much more straightforward. These are the GitHub ids you **must** add:

- AradMNK
- iman1234ahmadi
- MT522

0 Crawling

This section should have been completed from the previous phase until now. Keep this data as it will prove useful later on.

1 EDA

1.1 Basic

- Create *3* bar charts. For each bar chart draw the number of publications in a given year range. Do this for the ranges 1937-1950, 1950-1970, 1970-1990. Compare the three.
- Create a bar chart of the number of references over the years.
- Create a bar chart of the number of authors over the years.
- Find the Pearson correlation coefficient and Spearman Rank correlation coefficient between the *number of authors* and *number of references*.
- Find the Pearson and Spearman correlation coefficient between the *number of authors* and *number of citations*.
- Draw a bar chart of the title length over the years.
- Draw a wordcloud of the abstracts.
- Find a fitting correlation coefficient between the *title length* of each paper with the *title length of the papers it references*.
- Find the top 10 authors with the most publications.
- Find the top 10 authors with the most citations.
- Find the top 10 papers with the most references.
- Find the top 10 papers with the most citations **within** the **dataset**.
- Find a way to see how well the number of publications can predict the number of citations for a given author.

1.2 Network Analysis

1.2.1 Citation Network (Paper-Paper Network)

Nodes: Papers, **Edges:** Citation relationships (from `references` field)

- Plot the clustering coefficient over time to observe how interconnected the citation network becomes.
- Compute the average path length and diameter to understand the network's reachability.
- Identify influential papers using PageRank.

1.2.2 Co-authorship Network (Author-Author Network)

Nodes: Authors, **Edges:** Co-authorship relationships (if two authors have co-authored a paper, they are connected)

- Compute network density per year to analyze how collaborations evolve.
- Identify influential researchers using centrality measures (degree, betweenness, closeness).
- Find communities of authors working in similar fields.

1.2.3 Venue Network (Conference-Journal Network)

Nodes: Conferences/Journals (from `venue` field), **Edges:** If a paper cites another paper from a different venue, an edge is created between venues.

- Analyze interdisciplinary collaborations between venues.
- Find the most influential venues using centrality metrics.
- Identify emerging fields based on newly formed venue connections.

1.2.4. Temporal Evolution of the Citation Network

Construct the citation network for different years and analyze how the structure changes over time.

- Plot the network density per year to observe citation growth.
- Identify bursts of influential papers (papers that suddenly get many citations).
- Examine how new papers integrate into the existing network.

2 Data Extrapolation via Clustering

2.1 Community Detection

1. Find the author-author network. Make sure each author retains the data about the papers it has published.
2. Find the author communities in this network using a fitting clustering algorithm. You are allowed to use any algorithm, as long as the results make sense (Louvain, Walktrap, Hierarchical, Spectral, Newman, etc.). You must train 3 clustering algorithms.
3. Evaluate your clustering algorithm using the clustering coefficient and find the best clustering algorithm from the 3 clustering algorithms.

Hint: You can use a subset of the data to train and evaluate your clustering algorithms, find the best algorithm, and then run the best clustering algorithm on the whole data to reduce computation time.

2.2 Naming the Communities

1. Each paper has an *abstract* and a *title*. We want to extract **keywords** from these text fields. Use KeyBERT for extracting keywords from each abstract/title and aggregate them together to form a single list of words as *keywords* for each *paper*.
2. Associate each paper with the corresponding community of authors. A paper might be in two or more communities, so make sure to handle this correctly.
3. Aggregate the keywords of each paper from each community to get the *keywords of the community*. Explain your aggregation method, and report the keywords of each community in your report.
4. Give each community a name, and explain how you came to choose these names.

2.3 Paper-Paper Clustering via Embedding

1. Embed the abstract and title of each paper using a fitting embedding model such BERTopic or SentenceBERT, and aggregate them together.
2. Your embedding should contain the information of *keywords of the community*. An easy way would be to append the keywords as plaintext.

   Note: A paper might be included in several communities! An easy way to overcome this is to use the union set of these keywords.

3. Use any clustering algorithm of your choice to cluster these papers together.
4. Evaluate your clustering algorithm using DBI and Silhouette scores.

   Hint: Silhouette scores are calculated for each point. To evaluate your overall clustering technique, you must somehow aggregate them together, and report the aggregate. For instance, the *average silhouette score*. This means you can choose a small subset of points to calculate their average on, however the silhouette score for each of these points must be calculated over the whole data. We are unsure if current python packages support this, but you are allowed to use these optimizations in case long runtimes become an issue.

5. Report the unique venue values of the dataset.
6. Compute the Jaccard Similarity Index of your clusters with the clustering of **venues**. The venue of each paper is a decent label for clustering as well!

3 Citation Regressor

In this section we want to create a model that can estimate how many **citations** a given paper will have based on its **abstract, title, year,** and *(Bonus)* author-author network.

1. The training data is the abstracts of the entire dataset. Preprocess the training data, and use appropriate embeddings. You are allowed to use LLM embeddings (e.g. BERT models such as BERTopic or SentenceBERT) to boost your model's performance. Do not forget the train/validation split.
2. Find the best regression model and hyperparameters using **AutoML** or **GridSearch**.
3. **This is where your crawled data will come into play!** Your crawled data is the *test set*. Use your crawled data to find the estimated citation count and report important regression metrics (RMSE, MAE, R2, etc.). Give explanations of the reasons your model has overcome/failed these major challenges:
    a. **Time gap** between training/validation set and test set
    b. **New concepts/authors** that may be introduced over time, not being in the training data

Bonus: Author-Author Network

The idea is that scholarly paper authors have relations with each other that may be a great predictor of how many citations a given paper is going to have. To utilize this network, you will have to resort to using a **GNN**. You are free to use any architecture.

4. Product

The end goal is to create a product called **Research Assistant**. The Research Assistant is a LLM enhanced by RAG that can:

1. Find corresponding papers for a given topic;
2. And (hopefully) the LLM can perform analysis on these relevant papers and provide reasonable reports

4.1 RAG

Create a pipeline that takes a string input from the user, and will retrieve relevant papers from the database of papers. The database is a **vector database**, and is indexed based on the embeddings of abstracts/titles of your original dataset + crawled dataset and returns the relevant papers using a similarity metric (cosine similarity, L2, etc.). Use your model as the embedder for this stack. You are allowed to use **any** tech stack for this, such as:

1. **(Bonus) Classic:** Use a database such as Postgres, add the `pgvector` or **(Even more bonus)** `pgai` extension, and
2. **Langchain**
3. **LlamaIndex**

- The user should also be able to retrieve papers based on the *author*. Use NER to your advantage. **Bonus:** You may use the LLM to find the author name, and then somehow search the database by that.
- You should use KeyBERT to extract keywords from the user's input so that the embeddings match the embedding method from previous sections.

4.2 Research Assistant (**Bonus**)

Prompt-engineer an LLM such that it gives reports/summaries of the retrieved papers from 4.1. Your model must be an LLM text generation model. OpenAI API, any HuggingFace model, Llama, or DeepSeek.ai is accepted. If you do not see your model of choice in this list, chances are it is accepted. You can consult the teaching assistant if you have any questions.