Indian Statistical Institute, Kolkata

Bibliographic Citation Recommender System

A dissertation submitted in partial fulfillment of the requirements for the award of
Master of Technology
in
Computer Science

Author:
P Omkar Ashrit
Roll No: CS1914

Supervisor:
Mandar Mitra
CVPR Unit, ISI

Certificate

This is to certify that the dissertation entitled **"Bibliographic Citation Recommender System"** submitted by **P Omkar Ashrit** to Indian Statistical Institute, Kolkata, in partial fulfillment for the award of the degree of **Master of Technology in Computer Science** is a bonafide record of work carried out by him under my supervision and guidance. The dissertation has fulfilled all the requirements as per the regulations of this institute and, in my opinion, has reached the standard needed for submission.

Mandar Mitra 09.07.2021

....

Mandar Mitra
CVPR Unit,
ISI, Kolkata

Acknowledgements

I would like to thank my supervisor Prof. Mandar Mitra for suggesting and guiding me to undertake work in the topic. I would also like to thank Mr. Dwaipayan Roy and Mrs. Dipasree Pal for helping me in many ways for understanding the problem and doing this work. I would also like to thank my friends who have supported and motivated me during this time.

P Omkar Ashrit

07/09/2021

Abstract

With increasing volume of published scholarly articles every year, it becomes demanding to include appropriate references in a paper. Recommendation systems for bibliographic citations suggests possible references to authors. We discuss performances of different recommendation approaches in this report. We also have implemented a prototype extension for VS Code which suggests references in real time to the authors. **This** is the link of the GitHub repository for the dissertaion.

Contents

Chapter 1

Introduction

1.1 Citation

A citation is a reference to any secondary source of information used in a research report. Whenever an author directly quotes, paraphrases or summarizes the essential elements of an idea, observation, result etc. conceived or reported by someone else, an in-text citation referring to the original source of information should follow. An in-text citation is a brief notation within the text of a paper or presentation that refers the reader to a fuller description that provides all necessary details about the corresponding source of information. These details are typically provided in the form of a bibliographic record that appears as a footnote, or at the end of the report.

1.2 Recommendation System

A recommender system (RS) is an automated system that selects and presents items or entities that are potentially interesting to a user. These entities may be products, ads, people, films, songs, music, etc. Recommender systems that we encounter on a daily basis include those deployed by Amazon, Netflix and YouTube. For example, when a user watches a film on Netflix, the system later recommends other, different films based on the user's previous viewing history. Similarly, Amazon recommends various products, based on a user's history of purchases and product views. Recommender systems are responsible not only for selecting the items that are shown to a user, but also for ranking them.

There are many approaches to build Recommender Systems. Below, we discuss the standard approaches in the context of Bibliographic Citation Recommendation System.

1.2.1 Collaborative Filtering

Collaborative filtering finds similar users and uses this information to make recommendations. A citation graph is generally used in these methods. This graph is formed using the citations between papers. By following the citation graph for a paper, one can find what papers cite it and what papers are cited by it.

Standard Collaborative Filtering algorithms view dataset as a ratings matrix, whose columns represent ‘items’ and rows represent ‘users’. Each entry of the matrix is a user’s rating for a specific item. Collaborative Filtering algorithms make recommendations by trying to predict what can appear in the blank entries of this matrix.

1.2.2 Content Based Methods

Content Based methods use the content of the input paper draft. They try to find papers with similar content and output the corresponding citation as recommendations. It involves the following steps.

- Content Analysis: This is an important step especially when the data is unstructured data like text. So, the content should be pre-processed to filter out the irrelevant stuff. This step is responsible to translate the unstructured data to a structured form.
- User Profile Learner: This step collects data for a particular user, tries to generalize it and build a user profile for each user. In our system, a scholarly article is considered as a user.
- Recommender: The final step, the recommender uses user profiles built in the previous step and makes recommendations.

1.3 Problem Statement

Citation recommendation describes the task of recommending citations for a given text. As the published scientific works have increased in recent years, the need to cite the most appropriate publications when writing scientific texts has become very challenging. The BCRS addresses the following problem, with input as a partially or fully written paper that might or might not have placeholders for citations, the system returns a ranked list of possible citations for local or global context as required. $2l$ words (l words before and l words after) around the citation contributes towards the local context. The general topic of the paper including the title, abstract etc can be considered to contribute towards the global context.

1.4 Our Work

The contributions of this dissertation are as follows.

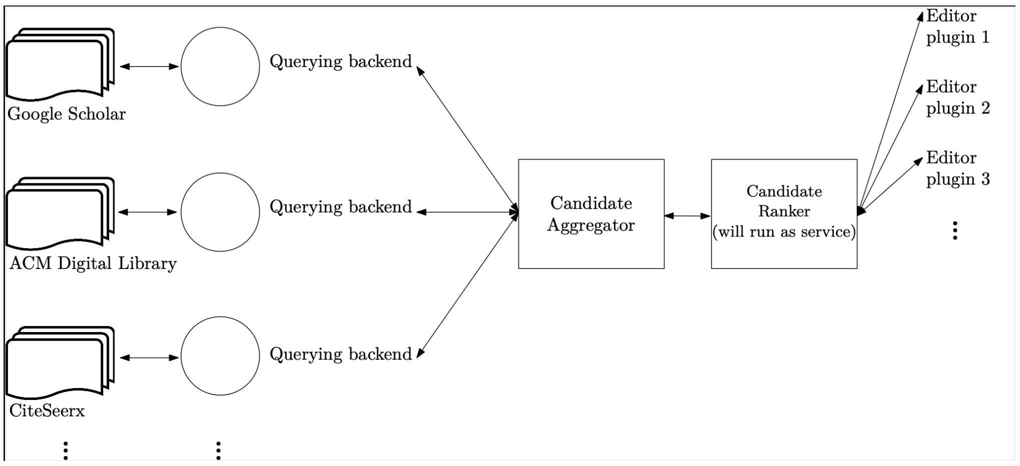
- We have used the proposed recommendation approaches by Roy [3] and reproduced similar results for *Content based methods* and 50% better results using *Reference Directed Indexing.*
- We have also compared the following approaches to the recommendation problem using a common data-set.
  - Sentence embeddings (USE, Word2Vec)
  - TF-IDF vectorizer implemented by scikit-learn
- We have developed an interactive plugin for the VS Code integrated development environment(IDE). The user can get recommendations in real time for the top n papers to be cited while writing a paper using the IDE and selecting the context around the citation placeholder.

Chapter 2

Recommendation Approaches

2.1 Problem Statement

Given a partially or fully written paper with a placeholder for citation as input, return top n papers from the database that could be cited considering different parts of the input paper.



The above is the schematic design diagram of the system. The system has the following components:

- **Querying Backend**: This part of the system runs in frequent intervals and connects to different sources (like Google Scholar, ACM Digital Library, CiteSeerx etc.) and collects an updated list of research papers along with their metadata.
- **Candidate Aggregator**: This part of the system aggregates all the papers in the database and then indexes them appropriately which can be searched by the search engine for the query to retrieve the result efficiently.

- **Candidate Ranker**: This part receives a query from the user and returns the top n papers to be cited. This and the above two contribute to the back-end of the system.

- **Editor Plugin**: Different well known editors will have a plugin installed in their application, which enables the users to interact with the candidate ranker and show the results to them. This is part of the front end of the system.

In this dissertation, we have focused on testing different recommendation approaches which is the part of Candidate Aggregator and Candidate Ranker. In the last part we have built of a plugin for VS Code IDE which interacts with the back-end of the system

2.2 Collection overview

Our dataset [2] consists of 630,199 scholarly articles from **CiteseerX**. These articles are drawn from various sub disciplines of Computer Science, Communication, Mathematics, Statistics, etc. Each file corresponds to one article and is defined by its DOI. Since similar papers have different DOIs for different versions, we have assigned a single cluster id to all such papers and have saved the mapping from DOI to cluster id. Each file is an XML file having details like title, authors, venue, year, DOI, abstract and list of all the citations in the paper along with cluster id of the particular paper. A context for a particular citation is 200 words before and after the citation. We have 22,23,307 different cluster ids in the dataset. The following is an example of an XML file corresponding to a document.

```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
<paper>
<title>Free-riding and whitewashing in ...</title>
<author>Michal Feldman</author>
<author>Christos H. Papadimitriou</author>
<author>John Chuang</author>
<author>Ion Stoica</author>
<venue>IEEE Journal on Selected Areas in Communications</venue>
<year>2006</year>
<key>journals/jsac/FeldmanPCS06</key>
<doi>10.1.1.1.1493</doi>
<abstract>We devise a simple model to study the ...</abstract>
<citations>
<citation>
<raw>ADAR, E., AND HUBERMAN, B. A. Free Riding on Gnutella...</raw>
<contexts>ants. However, individual rationality results ...</contexts>
```

```
<clusterid>173</clusterid>
</citation>
<citation>
.
.
.
</citation>
</citations>
</paper>
```

The XML tags `title, author, venue, year, key, doi, abstract` correspond to the Title, Authors, Venue of publication, Year of publication, Key of the Paper and unique DOI and abstract of the paper respectively. The `citations` tag contains all the citations present in the document. Each `citation` tag inside the `citations` tag has `raw` , `contexts` and `clusterid` tag. The `raw` tag contains the bibliography text for the cited paper, `contexts` tag has the raw text of the context around the citation and the `clusterid` tag has the cluster id of the cited document. If the cited document is not present in our collection, then this tag will have `None` in it.

The following is an example of the query XML file.

```
<query>
<top>
<paper_num> 1 </paper_num>
<paper_title> A Semi-Supervised ...</paper_title>
<paper_abstract>In this paper we investigate...</paper_abstract>
<query_num> 101 </query_num>
<text>   =-=e National University of Singapore 3 ... </text>
<query_num> 102 </query_num>
<text>   =-=nd Li, 2004; Mihalcea, 2004; Niu et al.,;...</text>
<query_num> 103 </query_num>
<text> based statistical methods ... </text>
</top>
<top>
paper_num> 2 </paper_num>
<paper_title> Dynamic Local Search ... </paper_title>
<paper_abstract> we introduce DLS ... </paper_abstract>
<query_num> 201 </query_num>
<text> applications, for exampl ... </text>
<query_num> 202 </query_num>
<text> ations divided by ... </text>
</top>
</query>
```

The XML tag `top` has all the queries from a single paper. Inside each `top` tag, the `paper_num, paper_title, paper_abstract` tag correspond to the unique number, title and abstract of the paper from where the query is framed respectively. The `query_num, text` correspond to unique number and raw text of the query around the citation place holder respectively.

We have removed digits, single letters, empty spaces, punctuation, unsupported text due to Unicode mishandling as part of our pre processing before doing any kind of indexing on the data.

2.3 Content Based Retrieval

We have used standard IR methods implemented in Lucene for studying content based methods on the training set and list of queries. For a particular paper, the concatenation of all the contexts in it is the context for it. And for a cluster id, we have taken the concatenation of all the contexts of all the papers represented by that cluster id. After removing digits, single letters, empty spaces, punctuation, unsupported text due to Unicode mishandling from the context, we indexed the cluster id using English analyzer of Lucene. This analyzer does the stemming and removal of built in English stop words. For queries we used 2 approaches: One was using the citation context only and the other was using the combination of title, abstract and citation context. The top 100 documents out of 10,000 hits were retrieved for each query. Lucene's implementation of Jelinek-Mercer smoothing for language modelling was used as the ranking model.

2.4 Reference Directed Indexing

In reference directed indexing, the citation context is used for the cited paper instead of the citing paper. Due to limited physical memory of our PC, we couldn't load all the contexts of 22,23,30 cluster ids to the system RAM. Hence, we created separate files for each cluster id and appended all the contexts in the file. Finally we merged all those files to create a single RD file. The cluster ids were then indexed using Lucene with removal of stop words and stemming using Lucene's inbuilt ranking models. All those papers which weren't cited at least once could not be indexed using this method.

2.5 Sentence Embeddings

We got vectors for each context in the RD file using average Word2Vec and Universal Sentence Encoder. We used the pre-trained model of word2Vec released by Google trained on the news data. This model gives a vector of 300 dimensions for each word present in its vocabulary. We got an average of the vectors of all the valid words in the citation context and represented the corresponding cluster id with that vector. Universal Sentence Encoder is also a pre trained model released by Google which takes a list of sentences as input and returns a 512 dimensional vector for each of the sentences. For each new query, after getting the vectors for the corresponding query, we used cosine similarity to find similar papers and returned the top 100 papers sorted in descending order of cosine similarity.

2.6 Scikit-learn TF-IDF Vectorizer

Scikit-learn (also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms. We used the TF-IDF Vectorizer which returns the document-term matrix for a given collection. We used this with all the default values and cosine similarity to find top similar papers to the given query.

Chapter 3

Results

The table below presents the results obtained using various approaches. For the content based method, queries were formed using either the context or a combination of title, abstract and context. For all other methods, only the citation context field has been used to retrieve the results.

We have used `Trec_eval` to evaluate our results with the ground truth. `Trec_eval` is an evaluation software which is used to evaluate an IR (Information Retrieval) system. This tool is used by the TREC community for evaluating an ad-hoc retrieval run, given the results file and the ground truth.

From the table we see that the content based approach taking the title, abstract and context of the queries gives the worst result. The most likely reason behind this, is the dilution of focus in the query by terms relevant to whole document, and not only to the query concerned. Further, the inclusion of title in query worsens the result slightly due to much smaller number of terms coming from it. The RDI approach performs the best with Lucene's JMSimilarity. Scikit-learn's TF-IDF vectorizer also performs better than other sophisticated models like USE and Word2Vec. One possible reason for these models not working might be the corpus these models were trained on. They are trained on a general English corpus consisting of news and Wikipedia articles; however the CiteseerX dataset has very specific technical words in them.

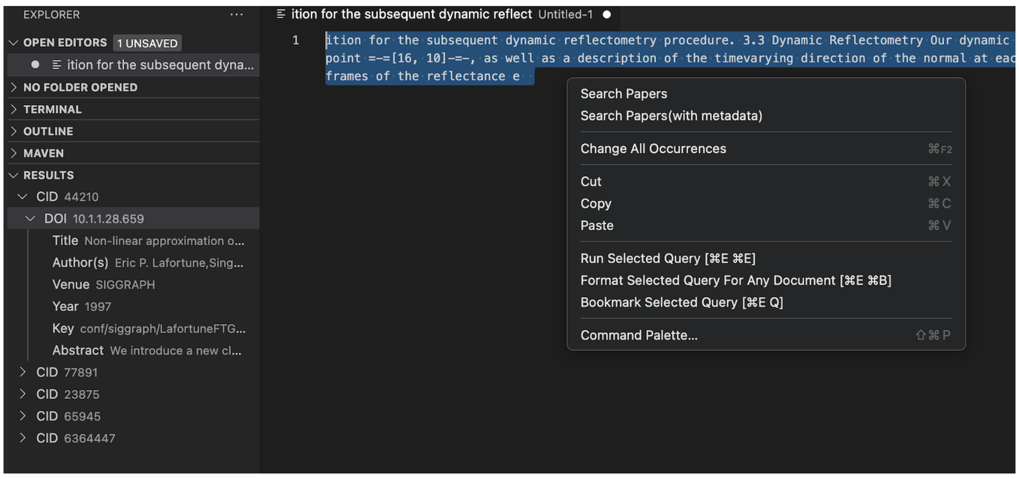
Indexing Method	Query From	Relevant Returned	MAP	Reciprocal Rank
Content Based	Title + Abstract + Context	207	0.0682	0.1758
Content Based	Context	270	0.1551	0.3821
Reference Directed Indexing	Context	**783**	**0.7549**	**0.9012**
Avg Word2Vec	Context	250	0.1246	0.3503
Universal Sentence Encoder	Context	285	0.1606	0.4010
SK Learn TF-IDF	Context	560	0.3739	0.6463

Chapter 4

VS Code Plugin

Visual Studio Code (VS Code) is a cross platform open source IDE made by Microsoft which supports extensions for additional functionality. According to Google trends, VS Code is one of the top 3 most downloaded IDEs worldwide. We have built an extension which recommends top 5 papers to be cited for the given context.

While writing a paper just selecting a context and searching for recommended papers lists out the top papers with metadata that can be cited for that context. Below is an example of how the interface looks in VS Code.


EXPLORER
OPEN EDITORS 1 UNSAVED
NO FOLDER OPENED
TERMINAL
OUTLINE
MAVEN
RESULTS
Search Papers
Search Papers(with metadata)
Change All Occurrences ⌘F2
Cut ⌘X
Copy ⌘C
Paste ⌘V
Run Selected Query [⌘E ⌘E]
Format Selected Query For Any Document [⌘E ⌘B]
Bookmark Selected Query [⌘E Q]
Command Palette... ⇧⌘P

ition for the subsequent dynamic reflect Untitled-1

```
ition for the subsequent dynamic reflectometry procedure. 3.3 Dynamic Reflectometry Our dynamic
point =-=[16, 10]-=-, as well as a description of the timevarying direction of the normal at eac
frames of the reflectance e
```

CID 44210
DOI 10.1.1.28.659
Title Non-linear approximation o...
Author(s) Eric P. Lafortune,Sing...
Venue SIGGRAPH
Year 1997
Key conf/siggraph/LafortuneFTG...
Abstract We introduce a new cl...
CID 77891
CID 23875
CID 65945
CID 6364447

4.1 How the plugin works

We have a JAVA server running which has access to the files indexed by Lucene. The front end of the extension is written in type script. When the user searches for papers, the front end makes a POST request to the server with the query context in the body and the server respond to that request with the resulting papers along with all the metadata of the papers. The user can configure the number of papers to be retrieved. Once the client gets the response, then it lists out the papers on the side panel in collapsed state.

Chapter 5

Conclusion

We have tried to remove noise from the CiteseerX dataset and tried different methods to generate recommendations for citations given a placeholder in a fully/partially written paper. We have seen that by far the the best performance has been achieved with reference directed indexing. Inspite of its sophistication, models like word2vec and USE didn't yield good results. Content based search was also not effective enough. Using the RDI method, that got the best result, we have build an extension for VS Code which returns results in real time to the user while writing a paper.

However, much work needs to be done. A more careful study of various machine learning based approaches is required where we can fine tune a model to our specific corpus so that the out of vocabulary words will be reduced and more context can be retained while vectorizing. An updated collection of papers with newer version of DOI links will be much helpful in getting the recent metadata for a paper in the extension. These remain the future work which will be undertaken later.

Bibliography

[1] Caragea C. et al. (2014) CiteSeerx: A Scholarly Big Dataset. In: de Rijke M. et al. (eds) Advances in Information Retrieval. ECIR 2014. Lecture Notes in Computer Science, vol 8416. Springer, Cham. https://doi.org/10.1007/978-3-319-06028-6_26

[2] Dwaipayan Roy. An Improved Test Collection and Baselines for Bibliographic Citation Recommendation. The 26th ACM International Conference on Information and Knowledge Management (CIKM-2017), Singapore, November 6-10, 2017.

[3] Dwaipayan Roy, Kunal Ray, Mandar Mitra. From a Scholarly Big Dataset to a Test Collection for Bibliographic Citation Recommendation. The 30th AAAI Conference on Artificial Intelligence (AAAI-16): AAAI Workshop: Scholarly Big Data 2016, pp.705-710, Phoenix, Arizona, USA, February 12-17, 2016.

[4] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M. & Duchesnay, E. Scikit-learn: Machine Learning in Python. *Journal Of Machine Learning Research.* **12** pp. 2825-2830 (2011)

[5] https://github.com/usnistgov/trec_eval

[6] https://tfhub.dev/google/universal-sentence-encoder/4

[7] https://code.google.com/archive/p/word2vec/

[8] https://lucene.apache.org/

[9] https://code.visualstudio.com/api