Mitacs Globalink Research Internship 2022

Report

Project Title: Best practices and solutions to explore data from scientific discoveries in computer science

By
Ritika Sarkar

Professor-in-charge: Dr. Fabio Petrillo

Duration: June 6 - August 26

At Université du Québec à Chicoutimi

I. Introduction

The project automates Snowballing, a method of conducting Systematic Literature Reviews. The framework is developed to aid in conducting individual research studies and also aggregate previously done work and studies on a specific research topic.
It implements the two types of Snowballing: Backward Snowballing, which uses the outgoing references from a paper to identify new potential papers; and Forward Snowballing, which identifies new papers using incoming references or citations to a paper. Here, the Reference string containing the Author names, Title, Publisher name, and Year is extracted along with the BibTeX of the papers containing the Abstracts. The Machine Learning model will be proposed to identify which papers are relevant to the study.

Need for the system

Performing systematic literature reviews and systematic mappings in Software Engineering research as well as in other fields is essential for formulating and answering research questions from the findings obtained from scientific papers. This process is manually performed and researchers spend months carrying out different rounds of Snowballing to identify new studies and evaluate them. The project creates an interface for the user i.e. researchers to automatically search for new papers using Backward and Forward Snowballing without having to manually look for them. A machine learning classifier has been made to classify the newly identified papers based on their relevance to the study.

II. Softwares used

The language used is Python.
The third party Python libraries used are:

1. Playwright - For web scraping
2. BeautifulSoup - For scraping dynamic webpages in a lighter way
3. Selenium - For scraping dynamic webpages
4. Semantic Scholar API - For getting the metadata and references and citations of the papers
5. Bibtexparser - For parsing the bibtex files into a uniform format and store them in one file.
6. Pandas - For creating and manipulating dataframes

7. Urllib - For making requests to URLs and receiving responses in desired format
8. Requests and Requests-html - For scraping HTML pages
9. Pyperclip - For accessing the contents stored in the clipboard
10. Sklearn - For extracting features using the vectorizers and building Machine Learning classifiers and evaluating them
11. Nltk - For preprocessing text forthe NLP task
12. Spacy - For preprocessing text for the NLP task

III. Implementation

The Snowballing automation is preceded by inputs from the user in the form of the start or seed set of papers which is required to proceed with either backward or forward snowballing, or both, as deemed by the user. The user also has to specify the number of iterations they wish to perform.

The bibliographic metadata in the form of BibTeX is extracted for the seed set of papers first. The user must select the most relevant papers for the research and place their DOI or Digital Object Identifier as URLs in new lines inside the file links.txt.

The function ***seed_iter***(links) takes the DOI URLs of the papers from the links.txt as a list in the variable *links* and searches for the paper in Semantic Scholar using its open source API in the ***search_sscholar***(paper) function. The reference string of the paper is obtained using the ***get_refs***(paper) function which takes the paper object returned as a dictionary by Semantic Scholar API. The references are saved in the References.csv file. The indices of the References in the csv file are saved as a list in the variable *all_indices*. The ***extract_bib_abs***(p1,p2,p3) function extracts the bibtex using the DOIs by scraping them for papers belonging to the IEEE and Springer digital libraries and by making a request to the doi receiving the bibtex as a response for the other papers. The function returns the bibtex of all the dois passed. P1 refers to the dois of the seed set, p2 refers to the directory you want to save the updated csv file in, and p3 passes the indices to be updated in the csv, which is the *all_indices* variable here. The ***extract_bib_also***(q1,q2,q3,q4,q5) function extracts the abstract of papers which are not published by IEEE and Springer, using ResearchGate. The user needs institutional login credentials for this, otherwise random results are returned by ResearchGate. The Username and Password can be kept confidential by the user by simply writing them as constants in another file and mentioning its name in gitignore. Q1 refers to the dois of the seed set, q2 to the bibs returned by the ***extract_bib_abs***() function, q3 refers to the iteration number which is 0 for the seed set, q4 to the *all_indices* variable, and q5 to demarcate the files by the type of snowballing or whether it is the seed set; here it is mentioned as ‘seed’.

Once the BibTeX extraction for the seed set is over, either type of snowballing or both as chosen by the user executes. The backward snowballing implementation will be explained first.

The function ***backward***(links, num_iter, seed_papers) takes the DOI links present in links.txt as input in the variable *links,* the number of iterations, and a list of dictionaries returned by Semantic Scholar by the ***seed_iter***(links) function. For the first iteration the references of each of the seed papers must be extracted, which is already presented in the list of nested dictionaries named as the variable *seed_papers*. From the second iteration the variable *all_dois*, which collects the DOIs of the references, is utilised to search the Semantic Scholar for the papers' references using the ***search_sscholar***(doi) function as shown in the following code.

```
def search_sscholar(doi):
    sch = SemanticScholar(timeout=2)
    length = 16

    if "http://dx.doi.org" in doi:
        length = 18
    elif "https://dx.doi.org" in doi:
        length = 19
    elif "https://doi.org" in doi:
        length = 16
    elif "http://doi.org" in doi:
        length = 15
    paper = sch.paper(doi[length:]) # to remove the url part
    return paper
```

Next, the reference string is obtained for the papers using the ***get_refs***(paper) function by using the keys ‘authors’, ‘title’, ‘venue’ and ‘year’ returned in the metadata by Semantic Scholar, and stored as a list of strings in the variable *refs*.
The following code uses the ***doi_helper***(refs) function, which helps to find the dois if the doi is not present for a paper returned by Semantic Scholar, by using the CrossRef references-to-DOI search website.

```python
for p in range(len(rs)):
    reference = get_refs(rs[p])

    if rs[p]['doi']:
        dois.append(BASE_URL + rs[p]['doi'])
        refs.append(reference)
    else:
        ref_nodoi.append(reference)

if ref_nodoi:
    new_dois = doi_helper(ref_nodoi) # search for doi if not already there
    refs = refs + ref_nodoi
    dois = dois + new_dois
```

The references are added to the References-back.csv file as shown.

```python
if not os.path.isfile(direc):
    df = pd.DataFrame(refs)
    df.to_csv(direc, index=False, header=["References"])
    ref = pd.read_csv(direc)
    ref['DOIS']=""
    ref['Iteration']=0
    ref['Status'] = ""
    ref.to_csv(direc,index=False)

else:
    ref = pd.read_csv(direc)


    inde = all_indices[len(all_indices)-1]+1
    for k,val in enumerate(refs):
        ref.loc[inde+k,'References'] = val
    print(ref.tail())

    with open(direc,'w', encoding = "utf-8",newline='') as f:
        ref.to_csv(f, index=False)
```

Here if the file does not exist previously, a new one is created and then other columns like DOIS, Iteration, Status are also created. If the file exists, the *all_indices* variable populated from the previous iteration is utilised to obtained the starting index to write the new references after the last written reference in the csv file.

The *all_indices* variable extracts the indices of the newly inserted references in the csv file. The variable *completed* is used to indicate redundancy i.e. whether the paper's BibTeX extraction has been done before, using its DOI. Its shown below:

```
for j, s in enumerate(dois):
    last = ref.loc[ref['References'] == refs[j]].index.values
    cell_index = last[len(last)-1]
    if not (s=="No doi"):
        print(f'\nIndex of doi: {s} = {cell_index}')
        if s in completed:
            # mark done already in reference
            ref.loc[cell_index,'Iteration'] = i+1
            iteration = ref.loc[last[0],'Iteration']
            ref.loc[cell_index,'Status'] = "Done already in " +str(iteration)
        else:
            n_dois.append(s)
            cell_indices.append(cell_index)
    else:
        ref.loc[cell_index,'Iteration'] = i+1
        ref.loc[cell_index,'Status'] = "DOI not found"
```

If the extraction has been done before, then the ‘Status’ is marked as such. Otherwise the extraction is done.
Then the DOIs of the new papers are saved in the csv file, and ***extract_bib_abs***() and ***extract_abs_also***() functions are called for the bibtex extraction. This process is repeated for the specified number of iterations.

Forward snowballing also has almost a similar structure of code in the function ***forward***(links, num_iter, seed_papers), but here instead of the references the citations to the papers are extracted using the ***search_cites***(paper) function.

```
for paper in papers:
    dois = []
    citations, refs = search_cites(paper)
    all_dois = all_dois + citations
```

The ***search_cites***(paper) function is given as follows:

```
papers = paper['citations']
for i in range(len(papers)):
    reference = ''
    aut = papers[i]['authors']
    for j in range(len(aut)):
        reference += aut[j]['name']
        reference += ', '
    reference += '"' + papers[i]['title'] + '", ' + papers[i]['venue'] + ', ' + str(papers[i]['year']) + '.'
    all_refs.append(reference)
```

The ***extract_bib_abs***() as explained before extracts the bibs from the IEEE and Springer websites.

```
res = urllib.request.urlopen(doi)
nurl = res.geturl()
if ("pdf" in nurl):
    b = extract_only_bib(doi,ref,i)

elif ("ieee" in nurl):
    b = runieee(doi,page)

elif ("springer" in nurl):
    b = runspringer(doi,page)

#elif ("elsevier" in nurl):
 #   b = runsciencedirect(doi,page)

else:
    b = "Not standard"
```

It also calls the ***extract_only_bib***(doi, ref, i) function which gets the BibTeX without the Abstract by making a request to the DOI as shown below:

```
# Below code Credits: @ https://scipython.com/blog/doi-to-bibtex/
# fetch bib of each doi
def extract_only_bib(doi,ref,i):

    if not (doi ==  "No doi"):
        url = doi
        req = urllib.request.Request(url)
        req.add_header('Accept', 'application/x-bibtex')
        try:
            with urllib.request.urlopen(req) as f:
                bibtex = f.read().decode()
                return bibtex

        except HTTPError as e:
            if e.code == 404:
                ref.loc[i,'Status'] = "BIB not found"
                return "No bib"
            else:
                print('Service unavailable. 504 error')
                ref.loc[i,'Status'] = "BIB not found"
                return "No bib"
```

The ***extract_abs_also***() function uses ResearchGate to obtain the abstract of the papers. The Bibtexparser library in Python helps to parse the bibs obtained from the ***extract_bib_abs***() function and adds a new key called ‘abstract’ to the bib along with the extracted text as value. The bibs are all stored/appended to the allbibs-(type).bib file.

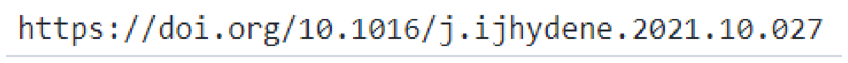
IV. Machine Learning Classifier

The machine learning models are based on the Naive Bayes algorithm, a probabilistic classifier which works well on small datasets and suits the training set in our study. Four different Naive Bayes models have been trained and tested on the data, namely, Multinomial NB, Complement NB, Gaussian NB, and Bernoulli NB from the Scikit learn library for Machine learning.
The ‘Title’ and ‘Abstract’ from the BibTeX files are used as the features for the ML models. Both the Title and Abstract are joined to form a single string for each row under a column named ‘Merged’. The ‘Relevance’ column is the binary target where ‘1’ indicates that the paper is relevant to the study, and ‘0’ indicates it is not relevant. The text is preprocessed by removing stopwords using the nltk library in Python, removing punctuation and urls, performing lemmatization, and finally vectorizing using the traditional Bag of Words ‘CountVectorizer’ and also the TF-IDF Vectorizer for comparing efficiency. The Count Vectorizer gives poor results when the n-gram range is increased by predicting with a greater precision only for the majority class. The TF-IDF Vectorizer has a better distribution of precision and recall for the two classes, and Multinomial NB performs the best in this, achieving a training accuracy of 90.07% and testing accuracy of 82.93%.
The dataset is created by taking the allbibs-(type).bib files and turning them into Training and Testing sets after partitioning them into Training_Excluded.bib, Training_Included.bib, Testing_Excluded.bib, Testing_Included.bib. Training set should include the bibs extracted from the seed set while Testing set should include the potentially relevant papers identified using snowballing.

V. Results

The results of the multiple iterations of the snowballing process are present in the CSV and the BibTeX files at the end of the runs. An example run of Backward snowballing starts with the dois in the links.txt file, which is the start set identified by the user.

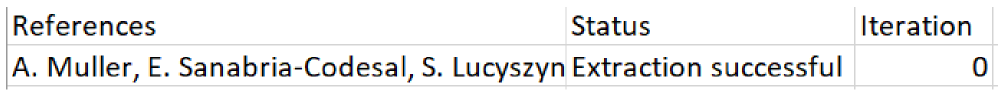
Filename: links.txt



```
https://doi.org/10.1016/j.ijhydene.2021.10.027
```

The References.csv file contains the reference string, the status of the extraction and the iteration number of the seed set.

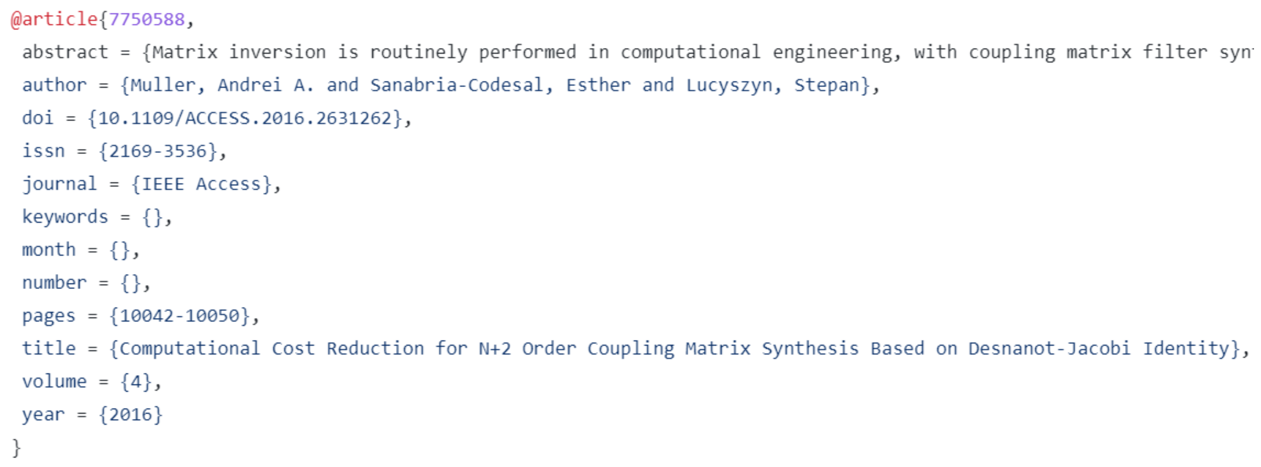
Filename: References.csv



References	Status	Iteration
A. Muller, E. Sanabria-Codesal, S. Lucyszyn	Extraction successful	0

The allbibs-seed.bib file has all the bibtex extracted from the seed papers.

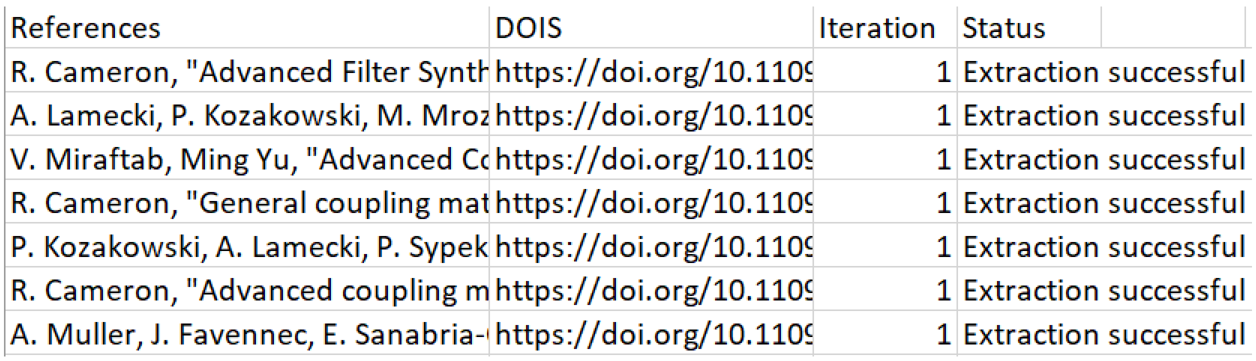
Filename: allbibs-seed.bib



```
@article{7750588,
 abstract = {Matrix inversion is routinely performed in computational engineering, with coupling matrix filter syn
 author = {Muller, Andrei A. and Sanabria-Codesal, Esther and Lucyszyn, Stepan},
 doi = {10.1109/ACCESS.2016.2631262},
 issn = {2169-3536},
 journal = {IEEE Access},
 keywords = {},
 month = {},
 number = {},
 pages = {10042-10050},
 title = {Computational Cost Reduction for N+2 Order Coupling Matrix Synthesis Based on Desnanot-Jacobi Identity},
 volume = {4},
 year = {2016}
}
```

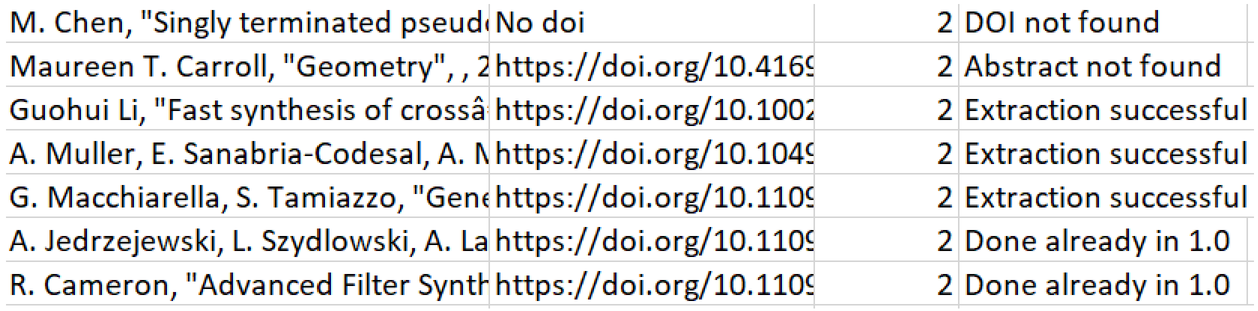
The References-back.csv file contains the reference string, DOIs, the iteration number and the status of extraction for backward snowballing.

Filename: References-back.csv



References	DOIS	Iteration	Status
R. Cameron, "Advanced Filter Synth	https://doi.org/10.110	1	Extraction successful
A. Lamecki, P. Kozakowski, M. Mroz	https://doi.org/10.110	1	Extraction successful
V. Miraftab, Ming Yu, "Advanced Co	https://doi.org/10.110	1	Extraction successful
R. Cameron, "General coupling mat	https://doi.org/10.110	1	Extraction successful
P. Kozakowski, A. Lamecki, P. Sypek	https://doi.org/10.110	1	Extraction successful
R. Cameron, "Advanced coupling m	https://doi.org/10.110	1	Extraction successful
A. Muller, J. Favennec, E. Sanabria-	https://doi.org/10.110	1	Extraction successful

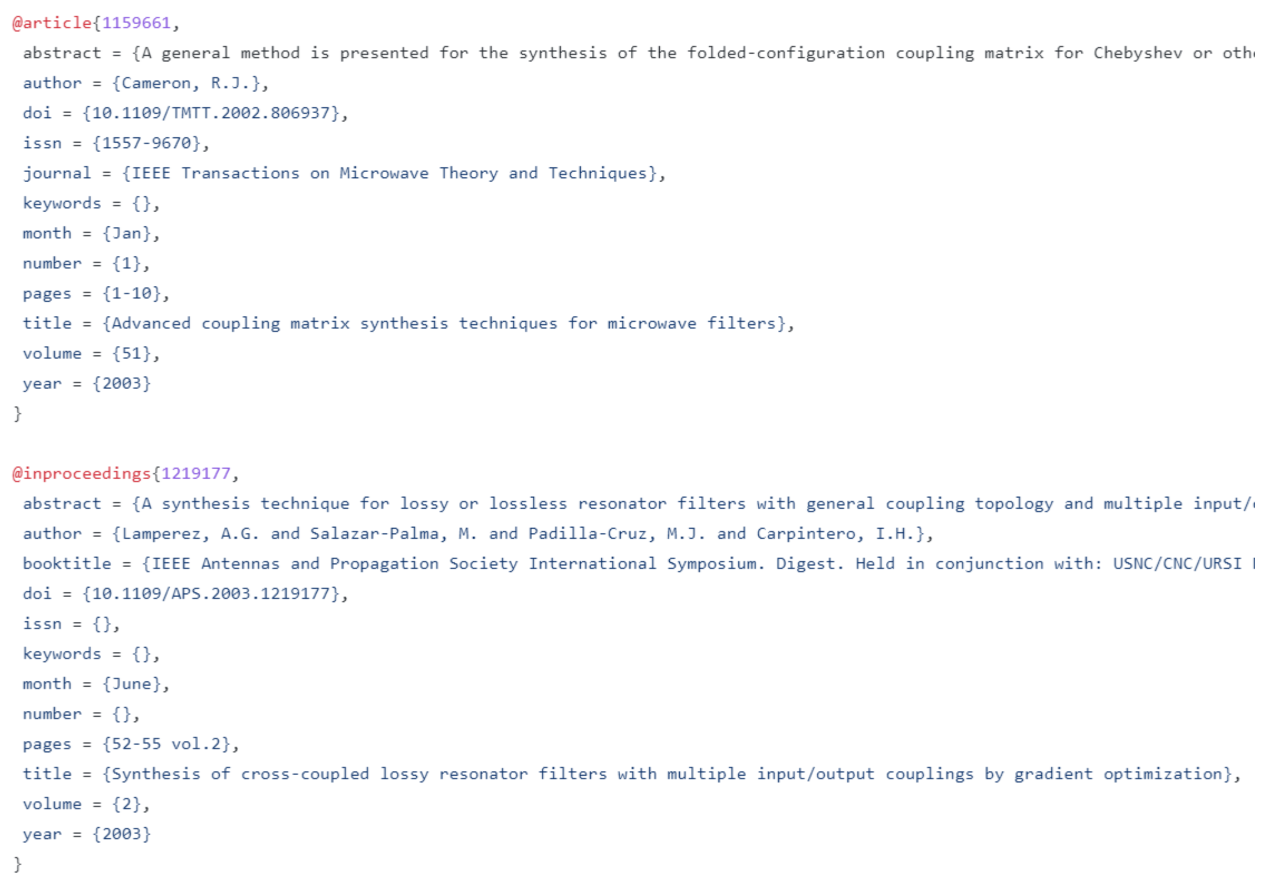
The code also checks the redundant papers using the DOIs and marks them done and in which iteration it was done, as shown in the example below. For the papers without DOIs and abstracts it is marked as such in the status.



M. Chen, "Singly terminated pseud	No doi	2	DOI not found
Maureen T. Carroll, "Geometry", , 2	https://doi.org/10.416	2	Abstract not found
Guohui Li, "Fast synthesis of crossâ	https://doi.org/10.100	2	Extraction successful
A. Muller, E. Sanabria-Codesal, A. N	https://doi.org/10.104	2	Extraction successful
G. Macchiarella, S. Tamiazzo, "Gene	https://doi.org/10.110	2	Extraction successful
A. Jedrzejewski, L. Szydlowski, A. La	https://doi.org/10.110	2	Done already in 1.0
R. Cameron, "Advanced Filter Synth	https://doi.org/10.110	2	Done already in 1.0

The allbibs-back.bib file contains the bibtex extracted from all the potential papers identified in backward snowballing and whose reference strings are present in the References column of References-back.csv file.

Filename: allbibs-back.bib



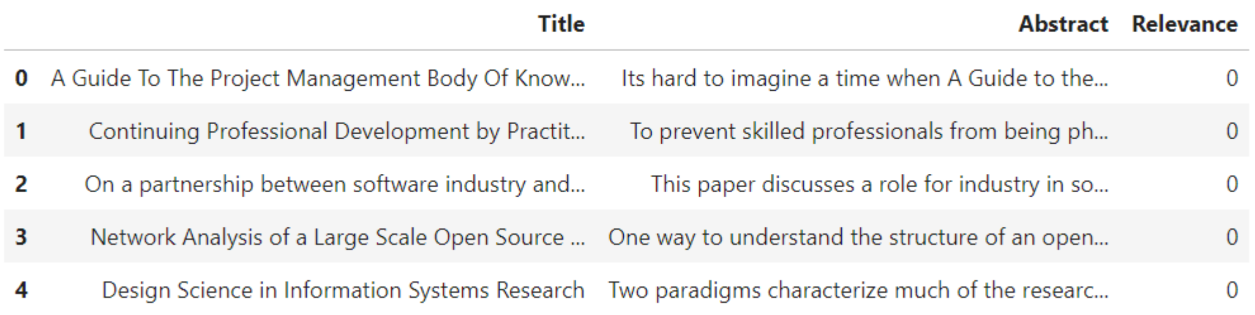
```
@article{1159661,
 abstract = {A general method is presented for the synthesis of the folded-configuration coupling matrix for Chebyshev or oth
 author = {Cameron, R.J.},
 doi = {10.1109/TMTT.2002.806937},
 issn = {1557-9670},
 journal = {IEEE Transactions on Microwave Theory and Techniques},
 keywords = {},
 month = {Jan},
 number = {1},
 pages = {1-10},
 title = {Advanced coupling matrix synthesis techniques for microwave filters},
 volume = {51},
 year = {2003}
}

@inproceedings{1219177,
 abstract = {A synthesis technique for lossy or lossless resonator filters with general coupling topology and multiple input/
 author = {Lamperez, A.G. and Salazar-Palma, M. and Padilla-Cruz, M.J. and Carpintero, I.H.},
 booktitle = {IEEE Antennas and Propagation Society International Symposium. Digest. Held in conjunction with: USNC/CNC/URSI
 doi = {10.1109/APS.2003.1219177},
 issn = {},
 keywords = {},
 month = {June},
 number = {},
 pages = {52-55 vol.2},
 title = {Synthesis of cross-coupled lossy resonator filters with multiple input/output couplings by gradient optimization},
 volume = {2},
 year = {2003}
}
```

Similarly for Forward snowballing the two files References-forw.csv and allbibs-forw.bib are created at the end of the rounds along with the References.csv and allbibs-seed.bib for the seed papers.

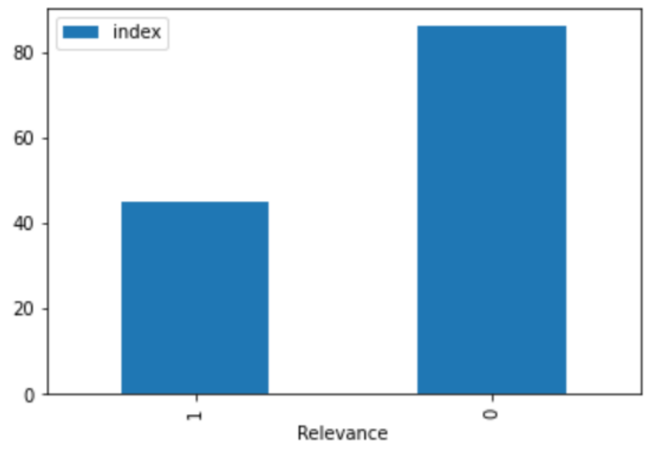
The dataset used for the Naive Bayes classifiers is created from the bib files created while performing snowballing manually. They are separated into the Training-Excluded.bib which contain the seed papers excluded from the study and Training-Included.bib which were included in the study, as the name suggests. Similarly for the testing set, the potential papers identified from snowballing were placed in the files Testing-Excluded.bib and Testing-Included.bib. The Title and the Abstract values were extracted from these bibs to create the Training.csv and Testing.csv with the 'Relevance' column indicating whether the paper should be included using '1', and excluded using '0'.

<u>Filename: Training.csv</u>

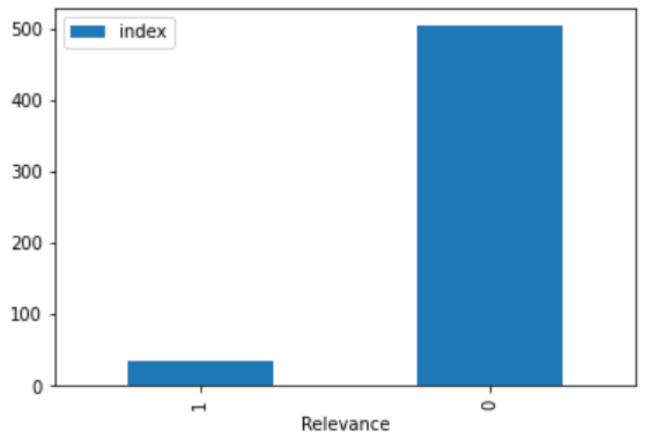


	Title	Abstract	Relevance
0	A Guide To The Project Management Body Of Know...	Its hard to imagine a time when A Guide to the...	0
1	Continuing Professional Development by Practit...	To prevent skilled professionals from being ph...	0
2	On a partnership between software industry and...	This paper discusses a role for industry in so...	0
3	Network Analysis of a Large Scale Open Source ...	One way to understand the structure of an open...	0
4	Design Science in Information Systems Research	Two paradigms characterize much of the researc...	0

The class distribution of the training set taken for the study is in the ratio 1:2 for Class 1 : Class 0.



The class distribution of the testing set taken for the study is highly imbalanced with Class 1 as the minority class.



The Title and Abstract are joined to one string under a new column named as ‘Merged’, and the training set texts after the preprocessing steps and just before the vectorization are as follows:

```
0      startseq startseq collaborative practice syste...
1      startseq startseq software security analysis i...
2      startseq startseq research empirical software ...
3      startseq startseq research academia research i...
4      startseq startseq information system action re...
                                ...
126    startseq startseq time collaboration many issu...
127    startseq startseq conduct empirical study must...
128    startseq startseq analysis large open way open...
129    startseq startseq action research model collab...
130    startseq startseq action science knowledge use...
Name: Merged, Length: 131, dtype: object
```

The Naive Bayes classifiers Multinomial NB, Complement NB, Gaussian NB, and Bernoulli NB from the Scikit learn library were trained and tested on the data, and Multinomial NB gave the best results of all in terms of accuracy on the test set, after applying TF-IDF vectorization as shown below.

```
accuracy_score_mnb = 82.93%
accuracy_score_bnb = 78.85%
accuracy_score_cnb = 64.19%
accuracy_score_gnb = 47.87%
```

The training accuracy of the Multinomial NB model was 90.07% and testing accuracy was 82.93%. The classification report detailing the precision, recall and the F1-scores of the two classes are as follows:

```
              precision    recall  f1-score   support

           0       0.86      0.95      0.90       453
           1       0.42      0.17      0.25        86

    accuracy                           0.83       539
   macro avg       0.64      0.56      0.57       539
weighted avg       0.79      0.83      0.80       539
```

The F1-score is very good for Class 0, as it has a higher support. The precision and recall are relatively better for the Class 1 using TF-IDF than Bag of Words. The performance can be improved by using a more balanced dataset.

VI. Discussion

The aim of the project was to create an intelligent system to identify relevant studies in Systematic Literature Reviews. Initially, it was attempted to make use of the digital libraries like IEEE, ACM, Springer, Elsevier to extract the references from the websites using the DOIs of the papers by scraping. However for subsequent rounds of snowballing, the Crossref website being used to extract the DOIs from the reference strings couldn’t handle the increasing number of reference strings and threw a 504 error. Hence the system was switched to Semantic Scholar where the DOI is used to query for the metadata, and the Title, Author list, Venue, Year as well as the incoming and outgoing references are returned as DOIs, eliminating the need to search for DOIs for the further rounds.

Some of the Bibtex returned did not have the abstract which plays an important role in deciding the inclusion of a paper. Hence, it was attempted to make use of Open Source websites like ResearchGate and Mendeley to find the abstract using the DOIs. ResearchGate works well only if the user is signed in and has an institutional account, while the idea to use Mendeley was dropped as it returned the results after querying its GraphQL based database as an AJAX response.

For the forward snowballing, it has been proven in literature that Google Scholar returns the most exhaustive results in terms of the number of citations to a paper. It was attempted to scrape the citations returned by Google Scholar using proxies and human mouse actions mimicking, but the application was not robust and behaved unexpectedly some times. The scholarly API based on Scholar was also tried by using proxies in the Google Cloud and DigitalOcean present as Lambda functions, but the scraping was detected and blocked by Google Scholar. The OpenCitations COCI API was also tried for getting the citations, but compared to Semantic Scholar and Google Scholar the number of results were lesser than half, and it was dropped as that would risk losing a significant number of papers not present in the COCI database. Hence the Semantic Scholar API was used as it returned results with less noise and in the format suitable for the application.

For the intelligent classification, a BERT model was built to classify the relevant papers based on the Title and Abstract, but due to the small size of the training set, the model ran out of data to train and overfitted. Hence, classifiers based on Naive Bayes were trained on the data and the results were good.

VII. Limitations and Future work

An important assumption in the study is that papers without DOI are not relevant. The effect of this is yet to be evaluated, but it is expected no major differences will be there. A limitation of the work is that the application depends on third-party websites like Crossref search, ResearchGate, IEEE explore, Springer Link, and the DOI urls. It is recommended to use the system with good internet connection as 504 gateway error has been observed while obtaining the BibTeX as a response for papers not published in IEEE or Springer with a poor connection.

The papers returned by Semantic Scholar are not as exhaustive as Google Scholar, but they are the most relevant to the study with little noise. The limitation is that some papers don't have the DOI or are not present in the Semantic Scholar database, so we risk losing some data. An alternative solution can be proposed in a future study with another API which has more exhaustive results.

The dataset for the ML classifier needs to be labelled only for the seed set. The performance can be improved by training the model for the papers related to a specific study, and other Machine Learning algorithms can be tested on the data as well. Another drawback is the unbalanced training and testing data which make the models perform poorer due to the minority class's low F1-scores. A potential improvement that can be made is by making a model capable of handling multiple inputs as the Title and the Abstract fields, because presently both are being joined into a single string and a single column is fed to the model.

References

1. Web scraping
    - Requests : https://requests.readthedocs.io/projects/requests-html/en/latest/
    - Selenium: https://selenium-python.readthedocs.io/navigating.html, https://data.library.virginia.edu/getting-started-with-web-scraping-in-python/
    - Playwright: https://playwright.dev/python/docs/selectors
2. DOI finder from reference
    - Crossref: https://search.crossref.org/references
3. DOI to bib
    - doi2bib: https://www.doi2bib.org/
4. Scraping Researchgate: https://serpapi.com/blog/scrape-researchgate-all-authors-researchers-in-python/
5. Metadata Extraction API
    - Semantic Scholar: https://api.semanticscholar.org/api-docs/graph
    - Usage of Semantic Scholar: https://github.com/danielnsilva/semanticscholar

6. Machine Learning
    - Naive Bayes: https://scikit-learn.org/stable/modules/naive_bayes.html