Documentation for CS410 Final Project: Improving ExpertSearch System

Mengyu Zhai (NetID: mzhai)

Fall 2020

1. Overview

1.1 Background

This project is for Fall 2020 CS410 Text Information Systems final project option 2.2 ExpertSearch System. The information quoted below is from CS410 course project instructions and what contained is the foundation of the current project:

"The ExpertSearch system (http://timan102.cs.illinois.edu/expertsearch//) was developed by some previous CS410 students as part of their course project! The system aims to find faculty specializing in the given research areas. The underlying data and ranker currently come from the MP2 submissions of the previous course offering. You can read more about it here (Sections 3.6 and 4: Project are especially relevant). The code is available here."

1.2 Functions of Code

The current project is trying to enhance the utility of the ExpertSearch System by extracting relevant information from faculty bios. More specifically, the code uses techniques for extracting other information, i.e., faculty research interests, than what is already provided in the original system.

Simply put, topic mining is performed on bios and top-keywords are found and shown as the common research areas, in addition to the bio information shown in the search result.

This added function does not influence the ways that users can use to conduct searches in the system. Users can search the expert by whatever search key word they want to use and applying whatever location or university filters they choose. Now if the user is especially interested to know which experts work in a certain research area and what their respective research interests are, they can directly search by that research area they have in mind. What is improved is that in the search result, rather than just showing the matching parts in the bio of an expert (or matching parts in the bios of all the experts in the search results) as before, a set of highly possible research interests (using top-keywords) of the expert will also be

provided. The method is not perfect as we assume top-keywords are common research areas but cannot guarantee that is always the case.

2. Implementation

2.1 What Are Used

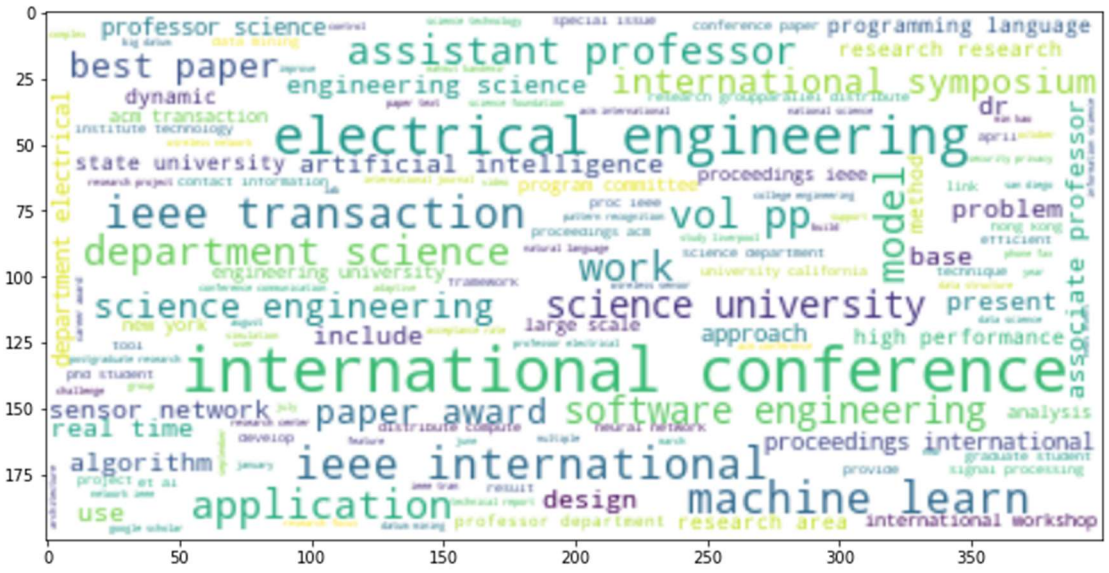
- Python
- LDA topic model: generative statistical model – to detect topics
- gensim API: to lemmatize and add bi-gram tokenization (build dictionary), and train LDA model
- pyLDAvis: to provide interactive visualization of LDA topic results
- word_cloud: to visualize corpus key words
- Other major tools/ packages involved: Jupyter notebook, numpy, pandas

2.2 Steps

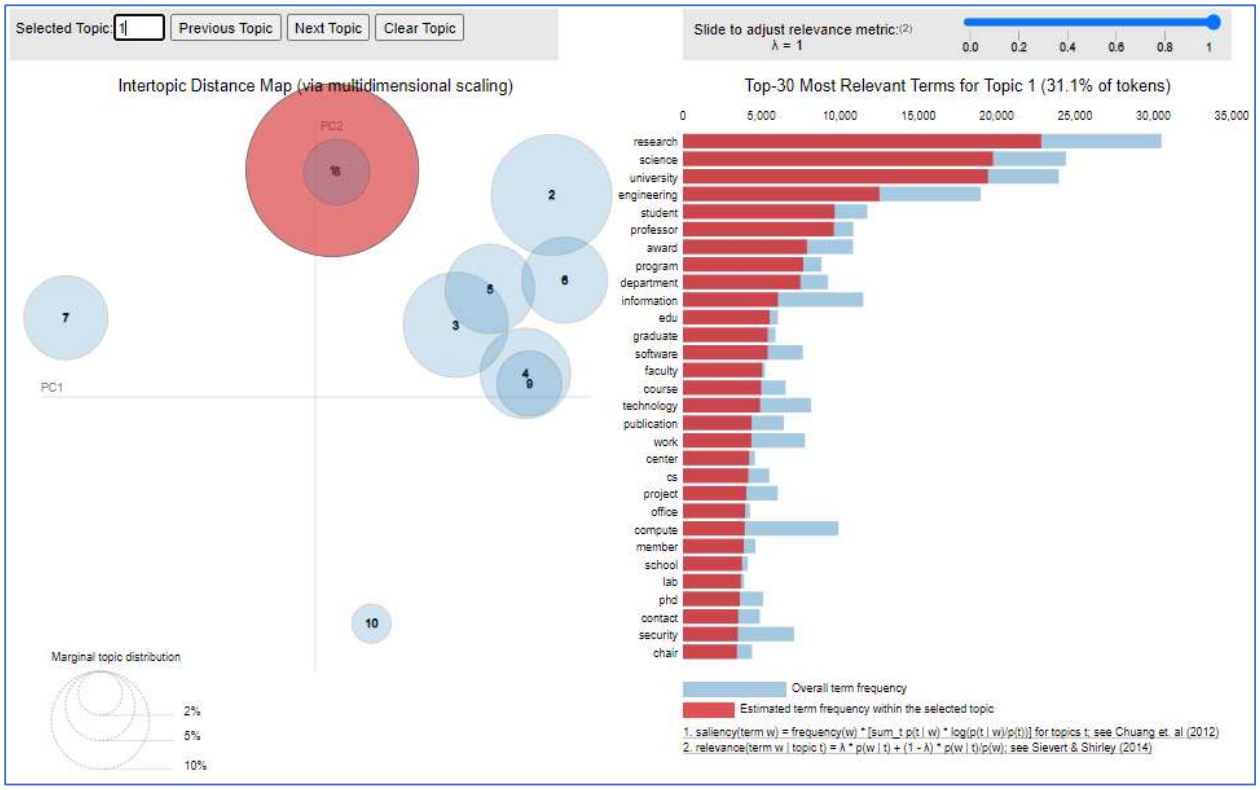
- With jupyter notebook, utilizing gensim API and LDA to build the topics from the combined bios (10 for each bio).
    - Preprocess data
    - Lemmatize and stop word removal
    - Build dictionary and use word_cloud to visualize the corpus
    - LDA modeling
    - Visualize LDA topics with pyLDAvis
- Then also within juyter notebook, results are saved into a researchinterest file under ExpertSearch\data.
- Finally modify the original ExpertSearch web application source code to add and display top topics as research interests.

2.3 Results

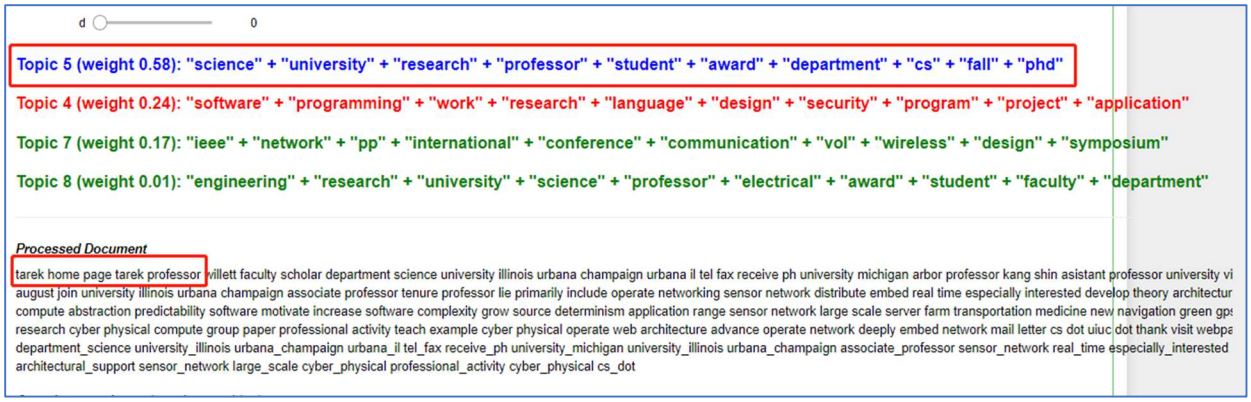
The corpus outlook built using word_cloud:



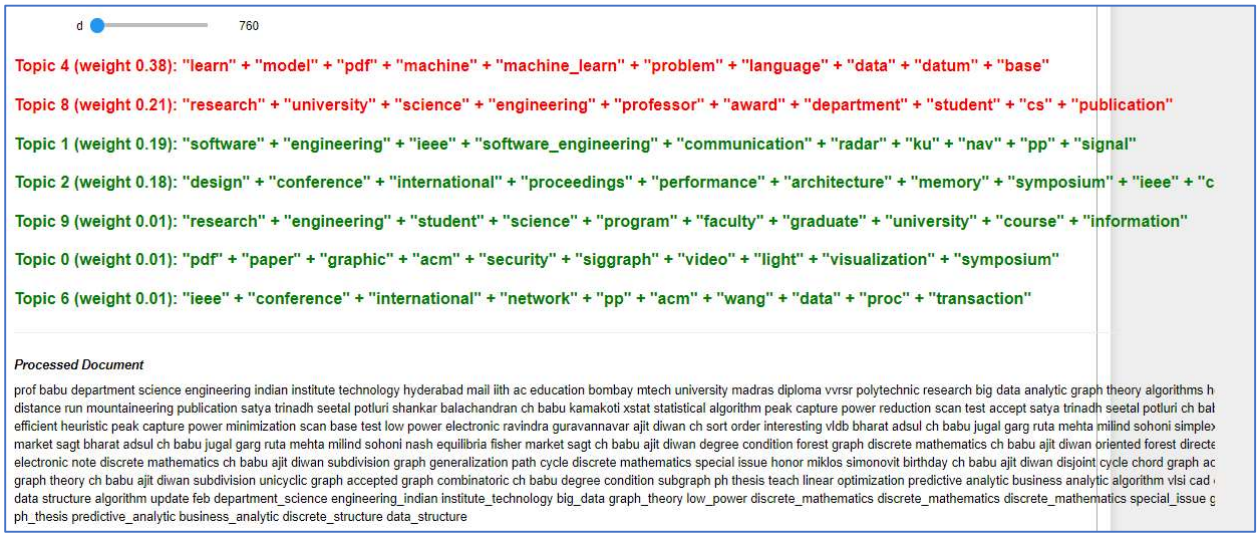
pyLDAvis output example (interactive and topic num = 10):



An example of some selected topics for a bio:

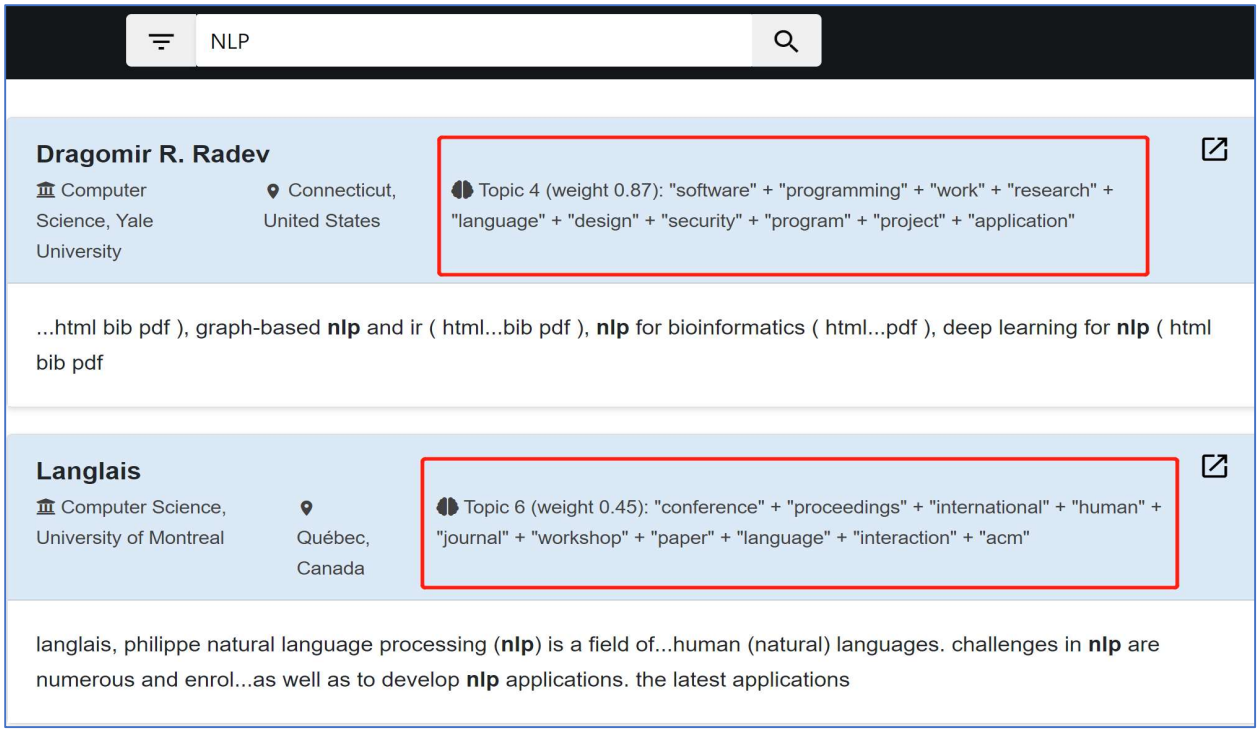


Another example of some selected topics for a different bio:

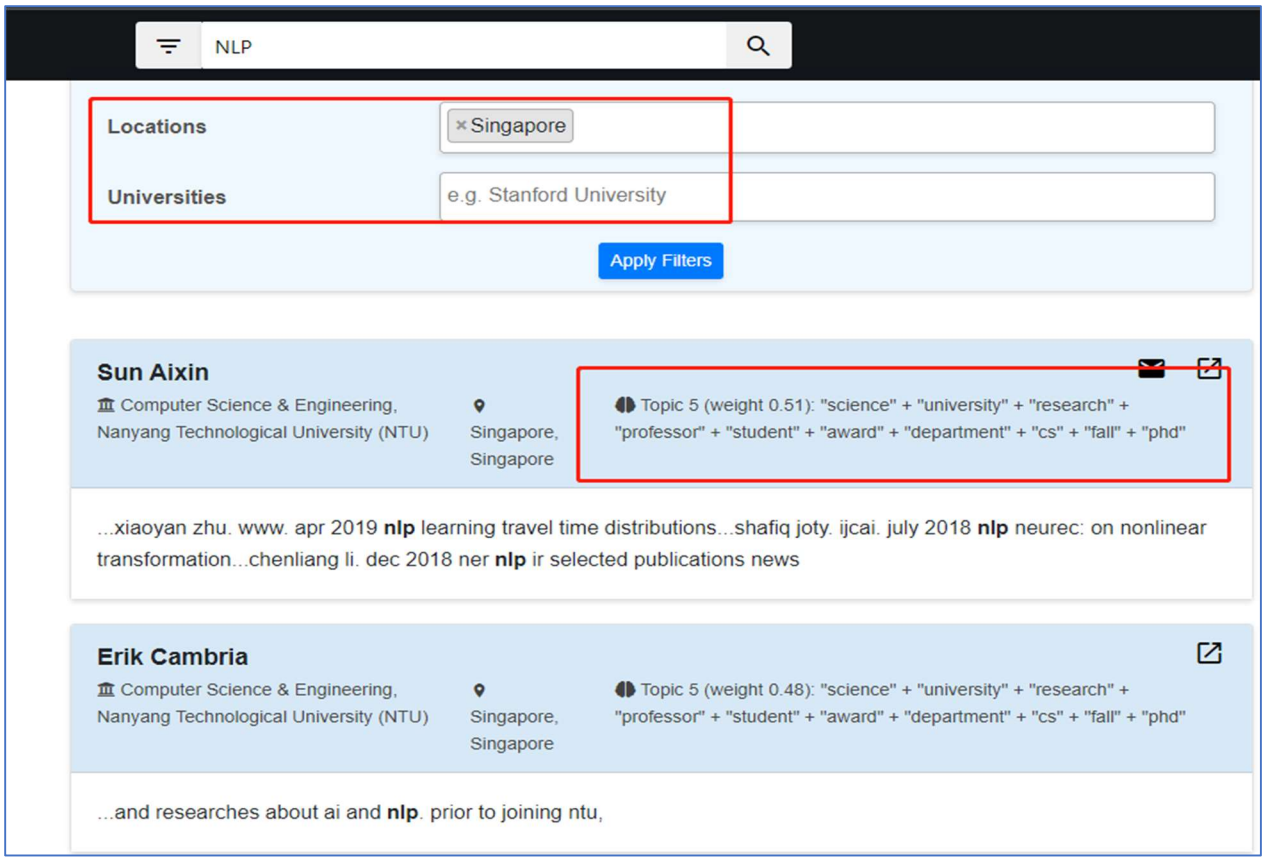


The top topics with the highest weight get saved to researchinterest.txt and will be the one to display in expert search results.

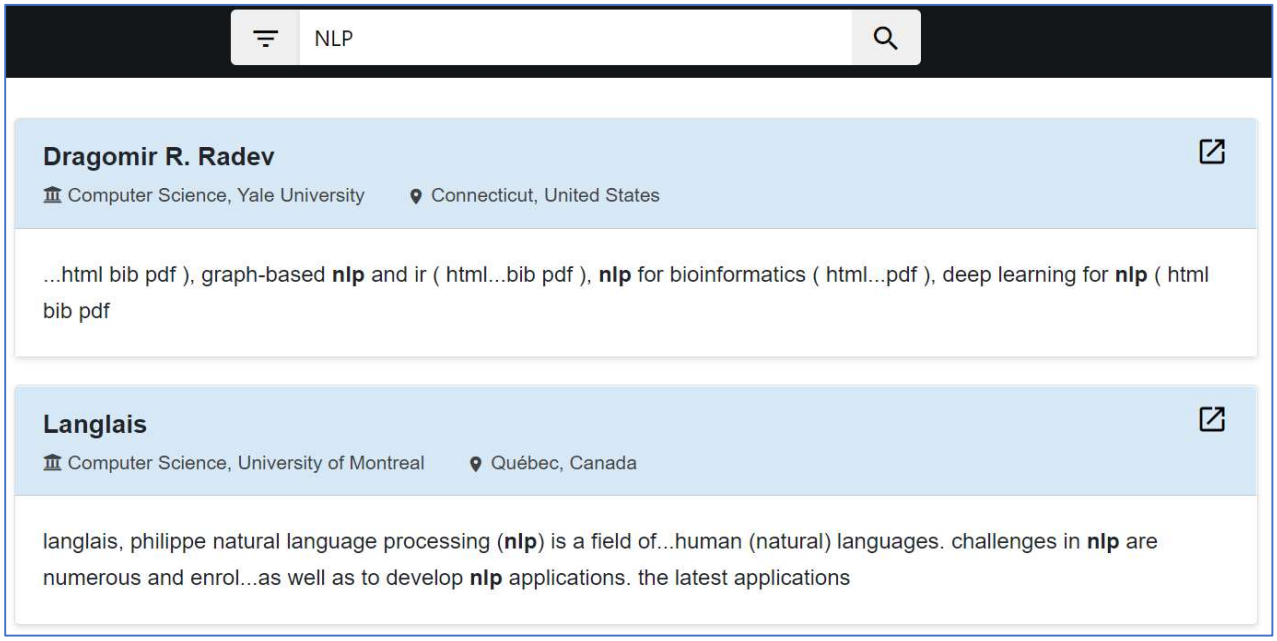
Search by key words, you will be able to see the results similar to image below:



You can also filter by locations and/or universities:



Now compare with the original system's results below, which doesn't provide top topic key-words:



3. Installation and Run

Codes uploaded to github include ExpertSearch_LDA.ipynb and web application codes in ExpertSearch.zip.

Be aware that the web application codes does not work on Windows (mainly because `gunicorn` is not supported on Windows)!

3.1 Simple Instructions (If you are familiar with the project)

To run the web application code, run the following command from ExpertSearch (work with Python2.7 on MacOS and Linux):

```
gunicorn server:app -b 127.0.0.1:8095
```

The site should be available at http://localhost:8095/

3.2 More Detailed Instructions (If you are not familiar with the original project)

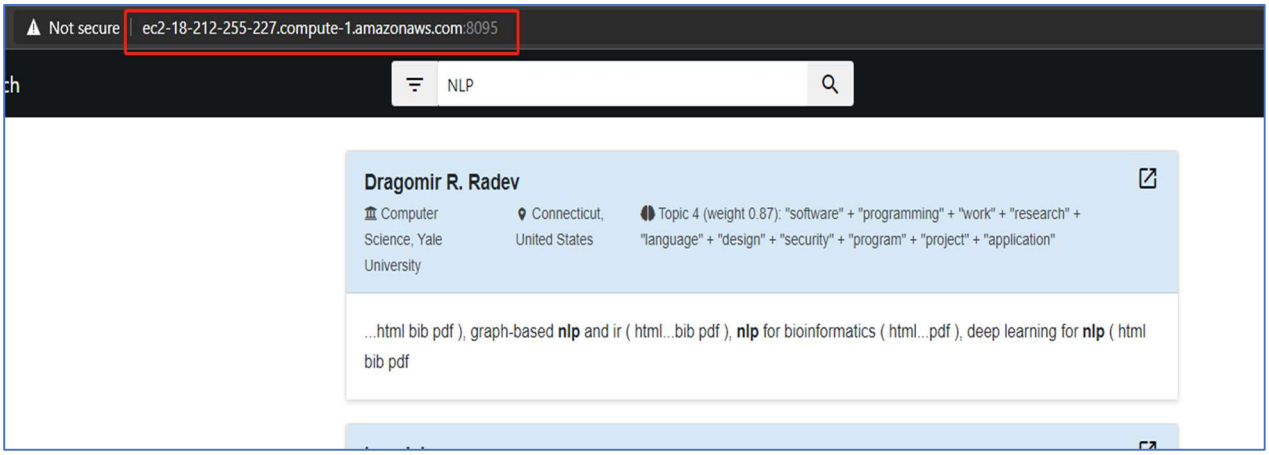
- Download ExpertSearch.zip.
- Unzip it in Python2.7 on MacOS and Linux.
- Go to the folder and run the following command:

```
[ExpertSearch]$ gunicorn server:app -b 127.0.0.1:8095
```

- You should see something similar to below once the website is running:

```
[ExpertSearch]$ [2020-12-11 03:07:57 +0000] [17325] [INFO] Starting gun
icorn 19.10.0
[2020-12-11 03:07:57 +0000] [17325] [INFO] Listening at: http://127.0.
0.1:8095 (17325)
[2020-12-11 03:07:57 +0000] [17325] [INFO] Using worker: sync
[2020-12-11 03:07:57 +0000] [17329] [INFO] Booting worker with pid: 173
29
```

- The site should be available at http://localhost:8095/. Otherwise, you can open the port 8095 so people can access it from http://yourdomain:8095, for example:



References:

1. [Reference code]: https://github.com/CS410Fall2020/ExpertSearch/
2. [Reference code]: https://github.com/TeddyWang0202/BeyondLD
3. [Reference file]: https://bhaavya.github.io/files/SIGCSE2020.pdf