CS410 Final Project Documentation

Apply state-of-the-art text retrieval methods on electronic medical records

Yeyu Ma, Yuanpei Ma, Hanyin Wang

Dec 6, 2022

1. Overview of the function

Electronic Medical Records(EMR) represents extremely rich text data generated from modern healthcare process. In this project, we develop a query system to evaluate the effectiveness of state-of-the-art text retrieval method on EMRs.

So the function query(query_string) inside search.py tries to identify candidate patients with certain diagnosis and treatment for enrolment into future clinical trials. For example, query("pneumonia ceftraxone"), we wish to identify all patients treated with ceftiaxone for pneumonia during the current hospital stay.

We obtain a collection of discharge summaries from MIMIC-III database, since this is a restricted-access database, we cannot share the actual data. But the following source code can be run on any text data collections.

2. Implementation of software

Software is implemented in python, with nltk package. Nltk package helps us to import common stop words and stemmers. The idea is the improved VSM, we try to use assign a vector to each document using tf-idf weight with normalization. Then when we get a query, we also generate a vector for the query. We calculate the cosine similarity between query vector and all document vector, then pick the top N results.

The main function is query() under Query class, which will be described later.

The first important function is read_csv(), in this function, we read the test_text.csv, and parse each line. So a dictionary of row_id to counter(word) is created. Also, the document frequency of each word is calculated.

The second import function is cal_tf_idf() and norm_weight(). Inside cal_tf_idf, we assign a vector to each document by the following rules.

1) calculate tf(w, d) = 1 + log10(count(w, d))
2) Calculate idf(w) = log10(total # documents / df(w))
3) For each document, vector = tf * idf
4) Normalize so that norm(vector) = 1

After we setup the Query class, we can query actual string. For example, if we query("pneumonia ceftraxone"), we will fetch all documents contain these two words, then we can use these documents to calculate tf, idf, and the vector of the query string. We then compare the cosine similarity between this query vector and all document vector so we can pick top N result.

3. How to install the software

First, you need to clone it from GitHub.

git clone https://github.com/y92ma/CS410_FinalProject.git

Then, you need to install anaconda for python environment.

https://www.anaconda.com/products/distribution

We have tested this code under python 3.10, so you can install virtual environment.

```
conda create ——name py10 python=3.10
conda activate py10
cd CS410_FinalProject/code
pip install -r requirements.txt
```

Then you can run the script search.py

```
python search.py
```

4. Contribution

Task	Person in charge	Total hours
• Complete the required CITI research training to get access of MIMIC-IV data (https://physionet.org/content/mimiciv/view-required-trainings/0.4/#1)	Hanyin	5
• Curation, clean and transform of the hospital summarizes data from MIMIC-IV into appropriate format/file.	Hanyin	7
• Apply software pipelines (e.g., MeTA) to build the search engine using different retrieval methods (e.g., BM25).	Yuanpei	10
• Manually annotate a collection of discharge summaries as testing data with domain knowledge	Hanyin	8
• Evaluation performance of the search engine based on testing data	Yeyu	8
• Refine search engine to improve performance. For example, try to fix the abbreviation problem.	Yeyu	8
• Source code and main results documenting and project demo preparation	Yuanpei	6

• Self-evaluation performance again based on testing data. May identify defects and possible solutions for future improvement	Yeyu/Yuanpei	4*2