WordCloud and Query System for Papers with Code – the latest in Machine Learning

蘇柏庄，郭毓萍

ABSRACT

This paper presents a query system for papers related to *machine learning* and *deep learning* based on text mining techniques. The word system is based on *tf, idf,* and *cosine distance*, which gives a criterion on importance and similarity between documents.

BUILDING THE SYSTEM

The query system is design as finding ten of the most related papers with the user's query. It goes through 4 steps, as shown below.

- *Step 1: Data Collection from the Web*
- *Step 2: Data preprocessing*
- *Step 3: Textual Weighting*
- *Step 4: Papers & Codes Recommendation*

Most steps are conducted with *Python*. The computational is time-consuming with a total of 10630 papers with cods needed to go through the whole processing. While building the system, we split the process into two parts in terms of whether we should update the data in step 1. The data without update is loaded to pickle files, which makes the system less time-consuming for users in most of the time.

✓ **Step 1: Data Collection**

Data can be collected from the website *Papers with Code: the latest in Machine Learning* https://paperswithcode.com/. Papers are categorized to two categories, with codes and without codes. In this system, the data we used are the ones with codes in order to have users reach papers and codes while learning or implementing more ideas on specific topic. The data contains titles, papers pdf urls, and *github* code addresses. We download all 10630 pdf files via papers pdf urls and parse them to text files(*.txt*).

✓ **Step 2: Data Preprocessing**

Through data preprocessing, text files are extracted to multiple tokens. We remove all punctuations and non-English words. The size of tokens has been greatly reduced after we removed all non-English words which takes about 80 percent of all text files since non-English words has much more words than the English words.

Then we go through the process, *Part of Speech tagging*, and extracted nouns and adjectives with the length of words longer than two. Lastly, we intersect the tokens from the text files and the tokens from the titles to form the base of our bag of words.

✓ **Step 3: Textual Weighting**

In the query system, we need high frequencies in single document and rareness in all papers, which makes *TF-IDF weighting* the best choice for textual weighting. Then we have every document went through the textual weighting processing and query through the process with the bag of words of the documents. In terms of performance, we save all tokens and the computational results of *TF-IDF weighting* into pickle files.

✓ **Step 4: Papers & Codes Recommendation**

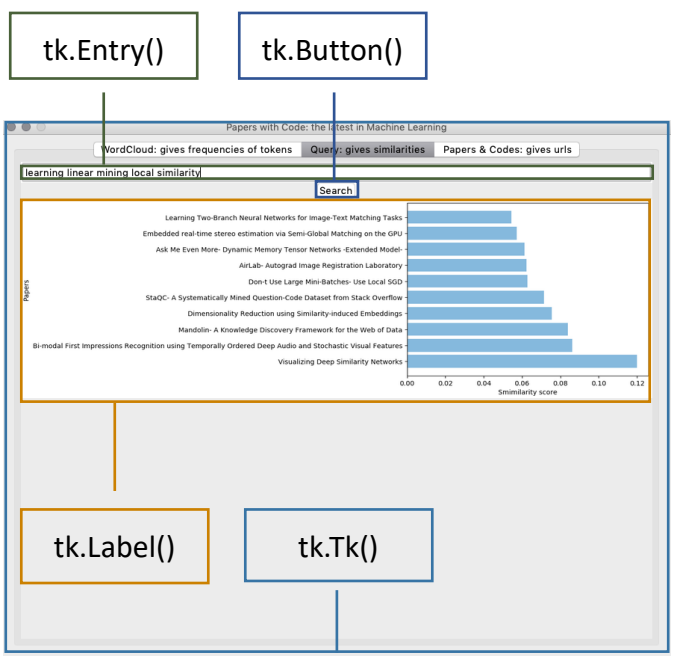
After receiving the user's query, the text will go through the text preprocessing as the documents had. Then, the cosine distance between the *TF-IDF weighting* of all documents and query one by one. The system will then pick ten of the closest distance to query with cosine similarity function, and give the user the result back with a plot with titles and similarity score. Moreover, the system will then give the user urls to access the papers and *github* codes for future learning or improvement.

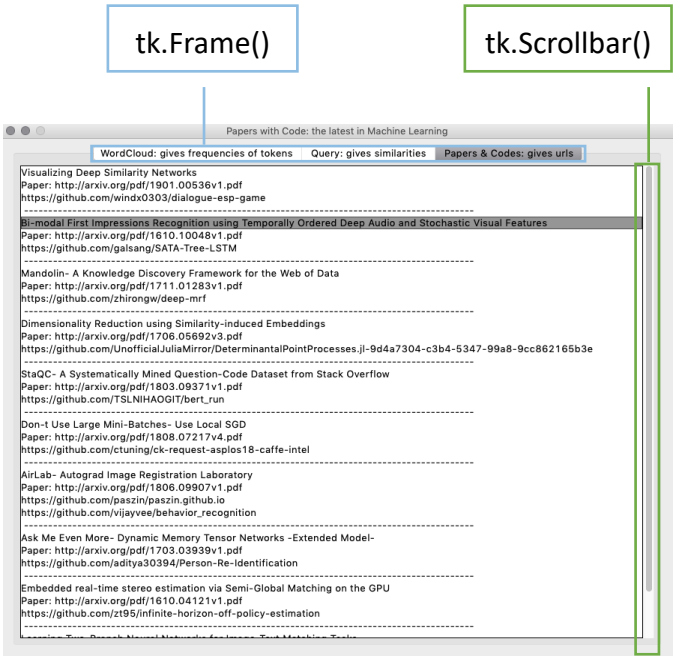
GRAPHICAL USER INTERFACE

In order to offer a better ease of use, we imported 2 python module, tkinter and ttk to create a *Graphical User Interface* (GUI). There are 3 different function partitions, " *WordCloud: gives frequencies of tokens* ", " *Query: gives similarities* ", and " *Papers & Codes: gives urls* ". The interface is illustrated in the figure in the *operation instruction* part.

The main python functions in this project are listed as following: (under ***tkinter*** module)

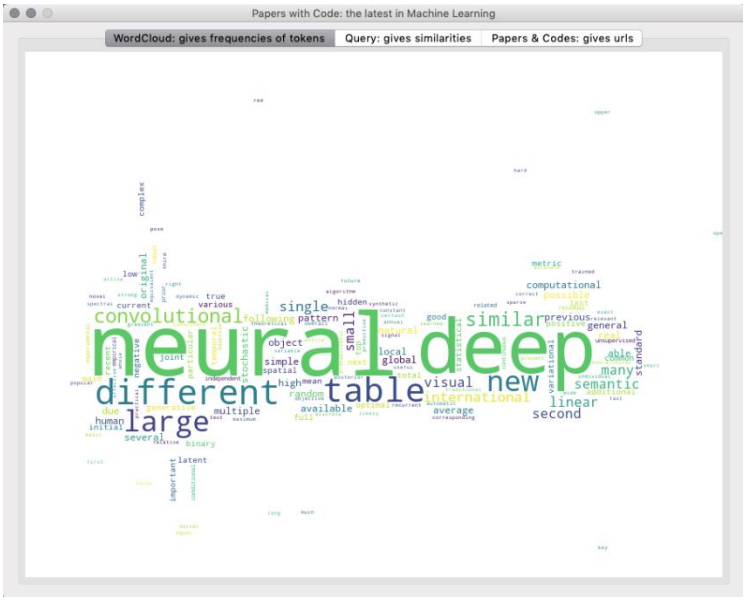
- *Tk()*
- *Label()*
- *Entry()*
- *Button()*
- *Srollbar()*
- *Frame()*



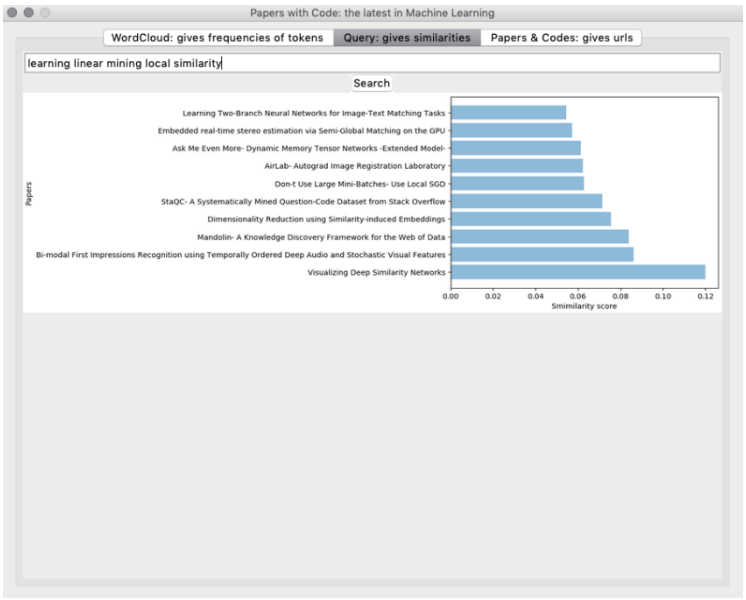


OPERATING INSTRUCTION

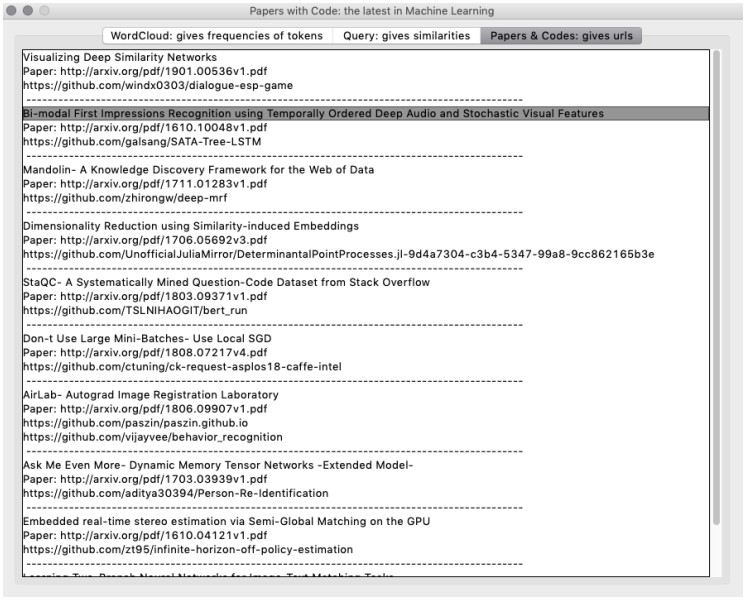
- ✓ Step 1: The default partition is " *WordCloud: gives frequencies of tokens* ", the word cloud is generated based on the frequency of words from our database. These are also inference words to put into the query box.



- ✓ Step 2: In the partition, "*Query: gives similarities*". Put interested words into query box and hit the search button. (And please wait for 30 seconds.) The result will show as a bar chart, showing 10 most related papers according to content similarity.



- ✓ Step 3: In the partition, " *Papers & Codes: gives urls* ". The 10 most related papers urls with codes urls are listed, giving users a direct and convenient access to the query results.



✓ Step 4: if having any other interested words, re-put the words into query box and hit the search button again.

REFERENCES

— *Papers with Codes – the latest in Machine Learning*
https://paperswithcode.com/

— *WordCloud documentation*
https://amueller.github.io/word_cloud/generated/wordcloud.WordCloud.html

— *Tkinter documentation*
https://docs.python.org/3/library/tk.html