Python Source Code Search

Andrew Malta
Computer Science, Yale College, New Haven, CT

Abstract

There exists a vast amount of source code publicly available on Github; however, there do not exist many easy ways to search for relevant code to particular tasks on the granular level of functions and classes. My project is focused on the construction of productivity tools for programmers and computer scientists, leveraging source code that can be extracted from Github. The initial stages of the project involve identifying repositories within a particular area of computer science; for example, repositories that deal with natural language processing and related fields. Professor Radev has collected a set of resources on the following topics, which can be found at the All About NLP (AAN) database. Next using static analysis of the code and information retrieval, this project aims to make this python source code indexable and searchable. The end result of this project will be an end-to-end system that given a query returns a number of relevant code end result of this project will be an end-to-end system that given a query returns a number of relevant code.

Data Collection

I collected all of the Github links listed in the resources of the LILY Lab’s “All About NLP” website. Using these links I wrote code to programmatically download archives of the repositories that each link references from AAN. With the archives of each of the repositories, I recursively extracted all of the source files from the unzipped repositories and stored them to avoid naming collisions. Of the 465 repositories extracted from the scrape of AAN’s resources, I was able to create a dataset of 10,138 python source files. This was enough to get interesting search results, yet manageable enough to not have to use distributed computing to make the search fast. Across these python source files I was able to find 84,317 functions and 15,127 classes, many with docstrings that improve the results of the search process.

Static Analysis

Using Jedi, a static analysis tool for python source code, I was able to extract useful structures from the code such as classes and function definitions.

In this phase of the project I extracted the names, starting line numbers, parent scopes, and docstrings of the named entities in the source files. While I had the starting line numbers, I needed to determine where each code fragment ended to enable the search application on top of this data.

Search

I ranked the search results using a variant of the popular tf-idf (term frequency–inverse document frequency) statistic in the information retrieval community.

Lastly, due to the fact that the name of the function, class, or file usually holds increased relevance to the task that it is performing, the scoring function awards higher scores to code with matches in the name of the source object.

Results From Demo

The top result when searching for “beam search” filtering for functions and searching both docstring and code.

Conclusion

I think that with a bit more work this kind of tool can be extremely useful for new and experienced programmers alike. I can see two primary use cases for a tool like this:

• A place to search for example code before heading to stack overflow to ask a question.
• A tool to explore a new topic that you are interested to learn more about.

References


Demo

To demonstrate the working application, I built an web interface in Flask which allows the user to search through the corpus of source code. In this web application the user can enter a query, specify whether they are looking for matches in functions, classes, or entire files, and can choose to search through just the docstrings or all of the lines in each code fragment.