DataSearch

By: Mudra Patel, Randy Shao, Emily Zhao
Supervised by Yuan Tian

Introduction

Studies have shown that software engineers spend on average 19% of their development time towards web search where they mostly look for relevant code snippets for their programming tasks, for data science practitioners, this number might be even higher as not all have a programming background. Today’s world has been enabled by data science to recognize pressing problems, discover new findings, and innovate. Though, writing code to perform data science tasks is not simple, especially for newcomers whose forte might lie in analysis but not in technical development. Additionally, with advanced frameworks and libraries being proposed, the learning and utilizing of these third-party tools is becoming a challenge.

Problem

There are numerous general-purpose code search engines that are currently used to support data science practitioners, including the built-in search engines in platforms hosting data science code, such as Kaggle and GitHub. However, these search engines are far from perfect. For example, if a data science expert wants to specify a specific framework or process for a data type, problem domain, or a data analysis pipeline, these search criteria are not considered when designing a common code search engine.

Existing search engines on the other hand, like many others, present the information they retrieve to data science professionals in order of relevance as a ranked list of notebooks. The relevance between a user query and a potential data science code is established through the use of additional filters such as tags, file sizes, file types, etc. to help find words in user queries and source code, which is determined through matching. However, such a simple search engine fails to fulfill common requests from data science practitioners.

Aim/Purpose

Our goal was to propose a set of criteria to measure the effectiveness and usability of a code example search engine, design and implement a new code example search tool leveraging the proposed criteria.

Methods

DataSearch’s front-end is developed using Next.js; a javascript framework which allows for building server-side rendering and static web applications using React; and Tailwind CSS; a utility-first CSS framework.

Firebase Firestore Database is the database used behind DataSearch. Its low-cost and quick set-up are some of the advantages to using this database. In the database, each project is stored with the following attributes: unique id for primary key, name, language, date created, number of views, url, a list of tags associated with the project, and its notebook content. These attributes are sourced directly from the Kaggle website.

The data used to populate our database currently comes from Kaggle, a huge repository of community published data and code. Kaggle offers a customizable Jupyter Notebooks environment, focusing on data science and machine learning.

To deploy our application, we are using Vercel, which is the best platform to host Next.js based web applications.

Results

With our application, we are able to display the list of Kaggle projects from the Firebase database, onto the client with Nextjs. The project has two different functionalities for conducting search functionality. The first method is search filtering, which allows the user to use the search bar to search and filter projects by name, tags, as well as specific code snippets that appear within a project’s content. By searching for project content, the results will display the content in a chunk of lines, as opposed to the entire content body, to help the user pinpoint where the search query appears in the project. Secondly, the user also has the option to filter results by the date that they were created, the language they use, as well as the number of views it has. All projects are hyperlinked, which will direct you to the project directly on Kaggle.

Next Steps

As all the data currently comes from Kaggle, our next steps would include expanding our data sources. Accessing more environments such as GitHub, Stack Overflow, Google open source code, and more sample code websites would bring more value to DataSearch and differentiate us from Kaggle.

Additionally, our database population is currently done manually. Our next steps would include web scraping our data sources so the information can be populated automatically. This would allow for our database to easily grow and have the ability to access more information spanning from a number of different data source websites.