Retrieve Relevant Code Components from a Repository

Kavide Balraju, Btech Student, CSE (Data Science), Vaagdevi College of Engineering

Parimala Vyshali, Btech Student, CSE (Data Science), Vaagdevi College of Engineering

Jannu Meghana, Btech Student, CSE (Data Science), Vaagdevi College of Engineering

Kandikonda Koushik, Btech Student, CSE (Data Science), Vaagdevi College of Engineering

Mrs.R.Divija, Assistant Professor, CSE (Data Science), Vaagdevi College of Engineering

ABSTRACT

The cognitive agent system helps to retrieve most relevant code component by introducing latest techniques. In this paper the authors used latest approach of code embedding which undergoes code2vec tokenization model by tokenizing and converting the code components present in the dataset into a numeric representation to create a input for neural network environment and also implemented cosine similarity matching technique to acquire the relevancy and perform retrieval of code component

1. INTRODUCTION

The growth of code component reusability had increased with most of the developers or end users majorly browse for the required code components in the internet, as it is providing many open source software code components. The user generally enters query in natural language and get plenty of results among which the relevancy of required code component is less, as it contains huge amount of noisy data than relevant data. To overcome this problem the authors introduced a cognitive agent system to retrieve most relevant code component from the repository. As per the work done to implement the concept the authors made use of a latest approach called Code2Vec. This is a neural embedding process of converting the code components into numerical representation called vectors. According to Piyush Arora etal[1], The conversion of code component to vectors can be done in three ways - one is general code as vectors in which the spaces or new lines and stop words are eliminated using tokenizer, another

one is tokenization in which it provides the lexical scanner for the code components to convert into vectors, and last one is AST(abstract structure tree) in which the code components are separated depending on their relationship and represented in the form of a tree. In this work natural language processing is used to perform the retrieval. The code2vec is mainly used to predict the method names. With an idea of fetching the method or code snippet of a particular method, it was decided to implement Code2vec concept as it was proven as effective code embedding for predicting the methods. As per Hong jin kang[2], they proposed token embedding's by code2vec to represent the source code in three downstream tasks as code authorship identification, code clones detection and code comment generation but resulted with a thread of not generalizing to other tasks.

By considering these two papers the authors decided to implement code2vec for retrieval process using tokenization. The cognitive agent system has implemented cosine similarity matching technique to fetch the most accurate code component from the repository - as per the Tim vor der Brick etal[4], they have calculated many similarity matching models and demonstrated cosine similarity as most accurate especially on large dataset. By analyzing piece of writings available, the authors have decided to implement a cognitive system which need to be user friendly, get accurate results and gives reusable code components

The vectorization idea is implemented as it reduces the complexity and increases the quality of results. The authors want to develop a user friendly system hence, implemented using "tkinter" to create a user interface and the query which the user enters can be in any random combination to get accurate results. The user input is a combination of features of code components required.

2. PROBLEM STATEMENT

The logic for almost all computer programs is readily available and stored in repositories, but we lack the ability to quickly and easily obtain the required component logic. Instead, we must rely on Google, which will retrieve the necessary component logic along with a lot of noisy data, making it challenging for programmers to extract the necessary logic from that massive amount of noisy data.

DISADVANTAGES

almost for logic search we need to depend on Google which will fetch required component logic with lots of noisy data. Its bit difficult for the programmer to get the required logic from that huge noisy data.

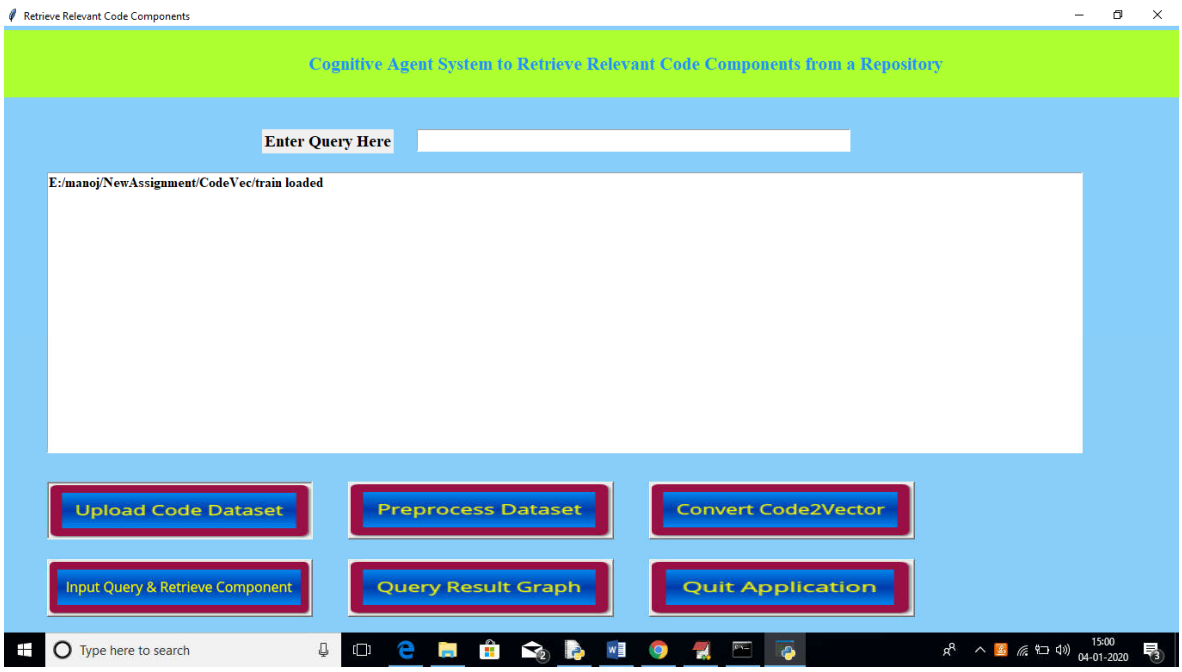
3. PROPOSED SYSTEM

Then Cognitive Agent System is the machine training and literacy process which will be train on being depository law element and also a train model will be generated. Whenever stoner give any query(to hunt law element) also Cognitive Agent System will apply train model on that new query to get applicable law element. Always this fashion will recoup only those factors which are satisfying query and make inventor work easier to get needed element sense.

ADVANTAGES

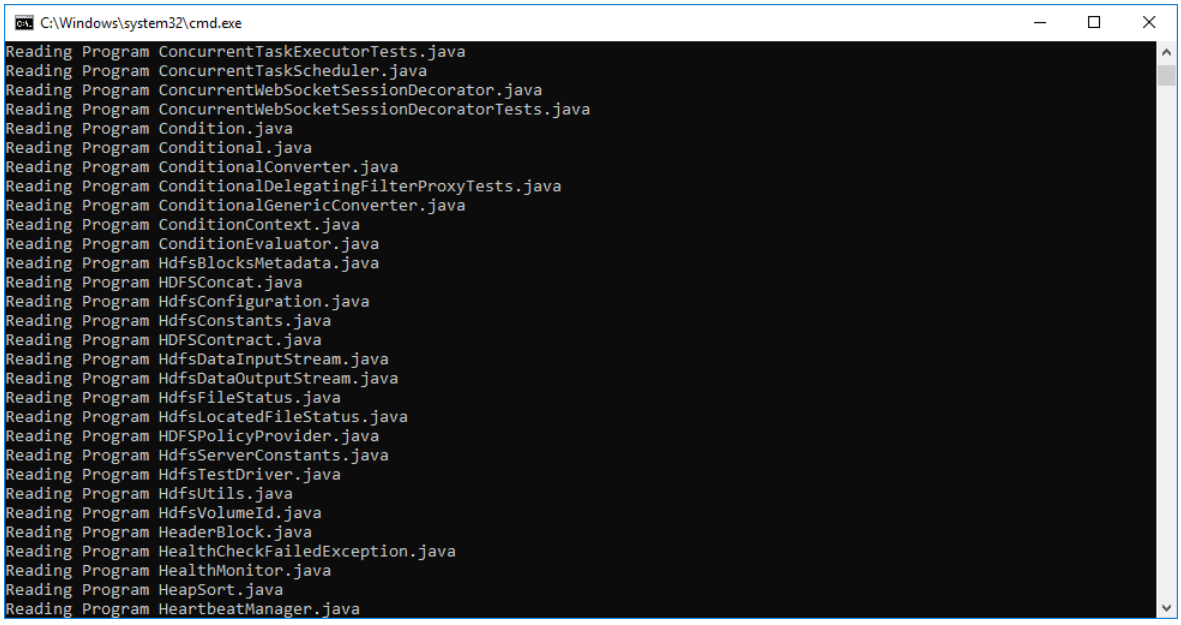
User gives any query then Cognitive Agent System will apply train model on that new query to get relevant code component. Always this technique will retrieve only those components which are satisfying query and make developer work easier to get required component logic.

4. EXPECTED RESULTS



Now click on 'Preprocess Dataset' button to read all programs and to clean all programs by removing stop words and special symbols. Will get below screen after preprocessing

In above screen we can see it process total 101 programs and to see processed program names then see black console. See below screen



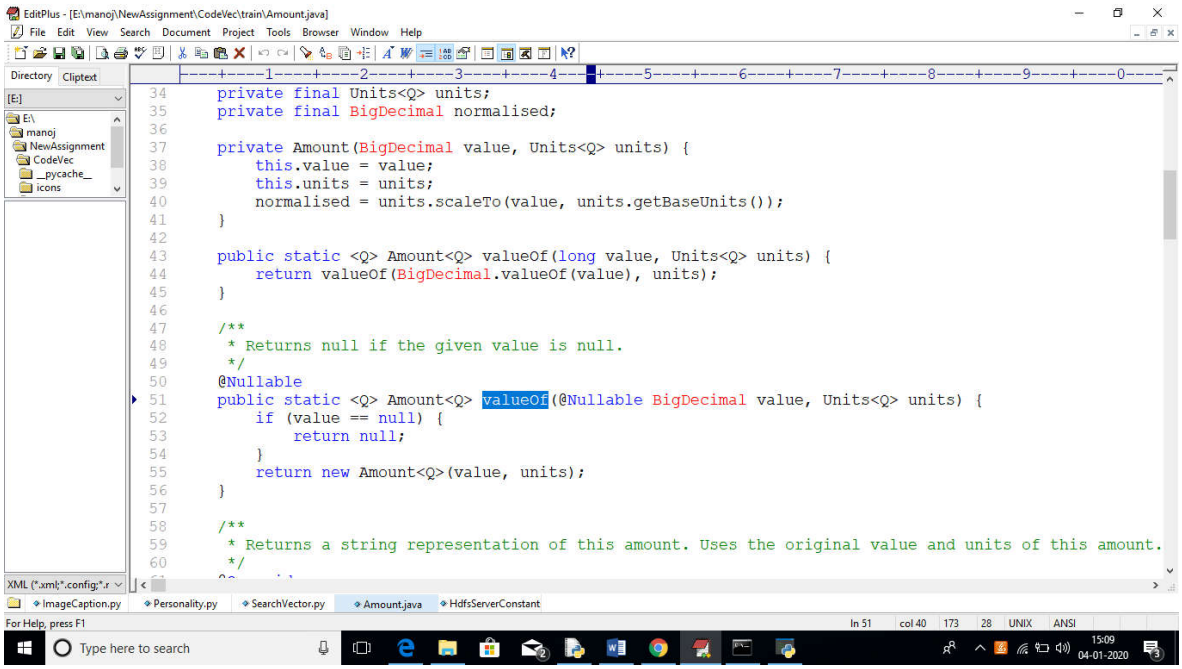
In above screen we can see name of each program which was processed. Now click on ‘Convert Code2Vector’ button to convert all processed program to vector

In above screen we can see program name and then its words and its count. Now enter your query in text field and click on ‘Input Query & Retrieve Component’ button to get all programs which implement that query. Once you build model then you can search any number of queries.

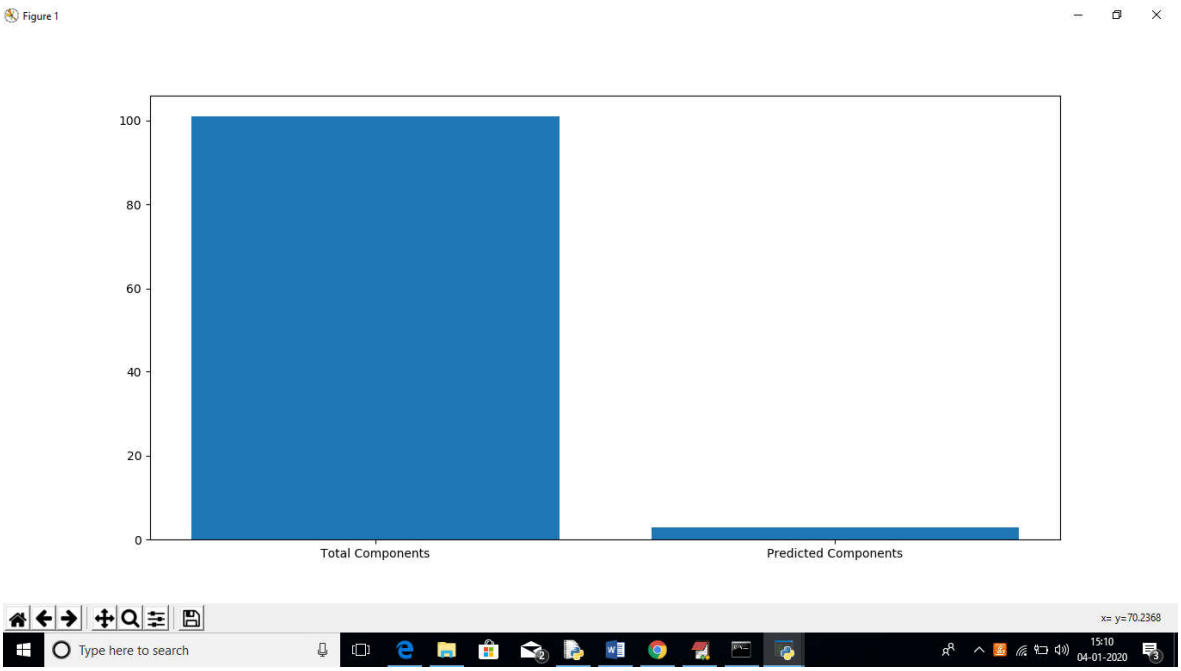
In above screen I entered query as 'valueof' which means I want to get all component which used or contains logic for valueof function. Below are the search results

In above screen we can see total 3 programs found (Amount.java, ConcurrentReferenceHashMapTests.java and HdfsServerConstants.java)which used or contains

logic for ‘valueof’. Now I will open ‘Amount.java’ program from train folder to see whether it really contains that valueof code or not



In above ‘Amount.java’ from train repository we can see it contains ‘valueOf’ function. Similarly you can see any query and get component. Now click on ‘Query Result Graph’ button to get below graph



In above graph x-axis represents total components and predicted components and y-axis represents it count.

6. CONCLUSION

To increase the reuse of software component, software component repositories and to improve the relevancy of the retrieval process - the authors have used Code2Vec concept in which the dataset is embedded by implementing tokenization technique which converts the whole code components from the dataset into vectors. The cognitive system attained success in retrieving the most relevant code snippet by comparing their cosine similarity measure and made it user friendly by abetting in the form of text document.

7. REFERENCES

1. DavidAzcona, Piyush Arora, I-Han Hsiao, Alan Semeaton, “user2code2vec:Embeddings forbProfiling Students Based on Distributional Representations of Source Code”, In The 9th International Learning Analytics & Knowledge Conference (LAK19), Mar(2019), Tempe, AZ, USA.ACM,NewYork,NY, USA,10pages. https://doi.org/10.1145/3303772.3303813

2. Hong Jin Kang, Tegawende F. Bissyande, David Lo, “Assessing the Generalizability of code2vec Token Embeddings”,34th IEEE/ACM International Conference on Automated

Software Engineering (ASE), 11-15 Nov (2019),https://ieeexplore.ieee.org/abstract/document/8952475

3. Bart Theeten, Frederik Vandeputte, TomVan Cutsem, “Import2vec Learning Embeddings for Software Libraries”,Proceedings of the 16th International Conference on Mining Software Repositories, May (2019) , https://www.researchgate.net/publication/33230053 8_Import2vec_-_Learning_Embeddings_for_Software_Libraries

4. TimvorderBr¨uck, Mare pouly, “Text Similarity Estimation Basedon Word Embeddings and Matrix Norms for Targeted Marketing”, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1, june (2019), https://www.aclweb.org/anthology/N19- 1181

5. https://www.machinelearningplus.com/nlp/cosinesimilarity/

6. CH, D. (2021). NARASIMHA CHARY,”. COMPREHENSIVE STUDY ON MULTI-OPERATOR BASE STATIONS CELL BINARY AND MULTI-CLASS MODELS USING AZURE MACHINE LEARNING”," A JOURNAL OF COMPOSITION THEORY, 14(6)