Semantic Code Search with Word Embeddings

Erica Assang | ericaassang@gmail.com
INK Lab
Proof School, Class of 2023
USC Viterbi Department of Computer Science, SHINE 2021

Overview

Machine learning as applied to natural language processing is a method that allows computers to gain a semantic understanding of text through data analysis. In other words, machine learning models can "learn" like humans do by identifying abstract patterns across large amounts of data, sample pairs of an input and an output (called a "label"). This method enables machines to complete tasks such as determining whether a movie review is positive or negative, or understanding the relationship between two statements, but it can also be applied to code. During this program, I focused on improving a module of my mentor's project, conducting semantic code search. The module takes in a block of code and a possible description of the code's functionality, and then assesses the likelihood that the code matches the given description.

Prof. Ren's Research at INK Lab

Prof. Ren's research involves work in natural language processing, data mining, converting natural language to computer compatible forms, and developing machine learning algorithms capable of attaining accurate results without a large database of labeled inputs. He approaches these areas with a focus on machine learning algorithms that use imprecise or otherwise non-ideal data to generate labels for initially unlabeled data, which in turn can be used for training a model. This approach is significantly more cost efficient than using labeled data only.

Skills Learned

While working on my project, I have learned how to:

1) Use the Pytorch coding library for machine learning
2) Use Git software and Vim text editor in terminal
3) Export and utilize code from Github

I also gained a deeper understanding of neural networks, tokenization and word embeddings, and other core concepts in machine learning and natural language processing.

Pytorch https://pytorch.org/

Github https://en.wikipedia.org/wiki/File:Octicons-mark-github.svg

Git https://commons.wikimedia.org/wiki/File:Git-logo.svg

Methods

In order to compare the description (or "query") to the code accurately, the program first *tokenizes* both. In this process, the machine identifies punctuation, spaces, and new-line characters and breaks up the strings into words (using a trained dictionary). These tokens are then broken into "subwords," which separate the root of each word from any prefixes or suffixes.

Example text: 'This is a poster about machine_learning'

Possible Tokenization: ['This', 'is', 'a', 'poster', 'about', 'machine_^', 'learn^', 'ing']

We then identify key words or tokens in the query, and search through the tokenized code for similar words. We give the code an understanding of semantically similar words using a deep learning model that has been exposed to textual data. This model uses the data to graph words by representing each with a vector. Words are thus considered similar if their corresponding vectors are close to each other (Fig. 1). In this way, the model can gauge the likelihood that the query describes the code by quantifying the similarities between the individual tokens.

```
vis_dfs[3] # query token = 'version'
```

0	1	2	3	4	5	6	7	8	9	10
0	def	parse^	_	versions	(	s	)	:	parts	=

Figure 1. Word embeddings example. Here, the code looks for words similar to 'version' in the tokenized code, making more similar tokens appear more saturated. Notice that the token 'versions' is bright red, whi the loosely related word 'parts' is lighter in color, and other words have barely any color.

Connections to Prior Work

In working on this project, I utilized my experience coding in Python and was also able to apply skills and concepts from my Java classes. Additionally, my ability to understand others' code efficiently and learn new libraries have improved significantly through working with my mentor.

Further Research

Possible extensions for this module include:

1) Making it compatible with multiple coding languages (the module is currently Python-compatible only)
2) Improving the word embeddings to consider relationships between words and symbols (e.g. 'list' and '[]') and to identify abbreviations more accurately

Acknowledgments

Special thanks to Prof. Xiang Ren for providing me this opportunity, and to my mentor Shushan Arakelyan for guiding me through the project and taking the time to share her knowledge. I would also like to thank Dr. Katie Mills and SHINE team for making this program possible.