INTENT PREDICTION IN CODE

With Applied Machine Learning Techniques

Justin Hugh

Data Science Student | BrainStation

Problem Statement

Software and code are becoming unavoidable in our daily lives both personal and professional, but only a fraction of us are literate in code. Even among developers, there exists a wide range of languages and frameworks so no one is familiar with it all. A model which could predict the intent of code would be hugely impactful for:

- Education. Making code more accessible and interpretable.
- Security. Identifying code with malicious intent.
- Development. Providing contextual tooltips, suggestions, resources.

The goal of this project is to develop an ML model employing NLP tools to interpret what a piece of code is trying to accomplish.

Background

Currently, there are no commercialized or productized options on the market that do a good job of interpreting code and generating an English description. A programmer who is looking to understand code whether in an academic, professional, or hobbyistic settings is left either to search Stack Overflow, or seek out time and help from a colleague.

I conducted preliminary research in order to find alternatives or studies on the topic, and while some others have begun study on this problem, it seems that state of the art applications in the field can perform well in translating python code to pseudocode, however at the moment there is no high-performing model, or model that can convert the code into an English description. For this reason, I'm excited to tackle this problem.

The Data

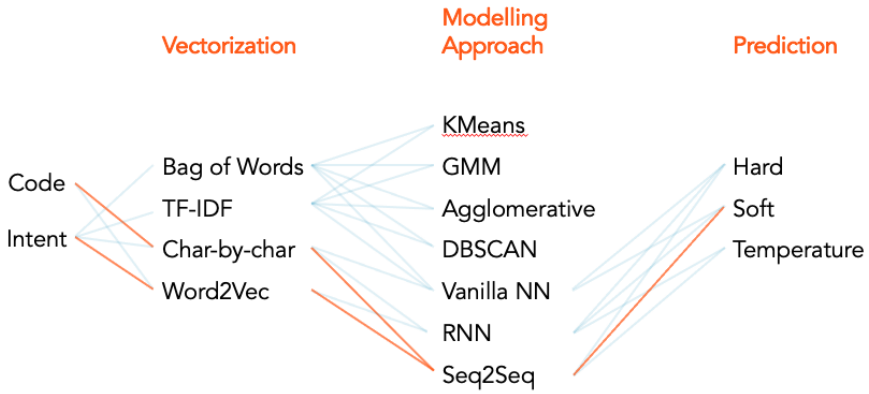
I conducted research, seeking sets of annotated code snippets and found a number of resources available, but with varying characteristics. I was able to find some well-structured and cleaned data but the data set was small.I found large quantities of code, but these were poorly formatted and of lower quality for my problem.

The best balance I found was the CoNaLa Competition Dataset. This data is freely available. The data set is well annotated, cleaned, and accessible. Both the Training and sets are comprised of user submitted queries to Stack Overflow paired with code responses. They have been rewritten such that the intent of the query is very clear, and with correct terminology. The training data is comprised of 2,379 records, while the test data is comprised of 500.

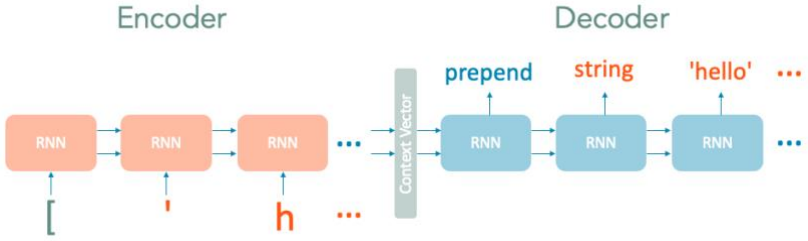
To process this data, I conducted EDA, gaining an understanding of its contents, and cleaning. I was able to remove duplicates from the set, drop null values, and visually confirm the quality of the provided code/intent pairings, which I was very satisfied with.

I also tokenized the data sets, breaking the code snippets into lists of characters, while tokenizing intent fields based on words, but also special characters. These were vectorized using 1-hot encoding, and represented in a list of numpy arrays. The intent of this tokenization was to capture the tokens which compose each of the text fields, as shown in the figure below.

I applied a wide variety of modelling approaches, including four clustering methods (KMeans, Mixture, Agglomerative, and DBSCAN) in order to find clusters in the code and intent fields. These generally did not produce valuable results in terms of silhouette score, inertia, or cluster distribution, and so I looked to deep learning methods.



The most promising deep learning model was a sequence to sequence model in which I was able to employ an encoder and decoder to interpret the context of code, and generate a sequence of words describing it.



With 20 epochs of training, my model achieved minimum loss of 0.779, and accuracy of 0.8474. However, these are not optimal methods of assessing its performance. In fact, I believe qualitative assessment is most appropriate (included below).

I was able to create a model an prediction function which, given snippets of code can create predictions for the intent of that code, in English. An example is visualized below.

```
Input sentence: print('hello world')
Decoded sentence:  get the possible dataframe   ` , contains a module the list path of variable the  tuple column string
-
Input sentence: for i in range(1,10)
Decoded sentence:  int the  of list  ` l `  _ index `  ` single
```

Summary and Conclusions

This model I created has some of weaknesses:

- Grammar – The outputs created are generally not structured in proper english. I suspect that expanding the amount of training data would help with this issue.
- Stoppage - The outputs often end late or prematurely. There are many examples of the predicted text including multiple ideas before it predicts an ending.
- Limited Vocabulary - There are only ~2500 target tokens. Some tokens like """, and "of" appear very commonly and frequently in the output string.

This model I created has notable strengths:

- Structural Patterns – The model is performing well in recognizing token-token relationships and appears to be predicting them in close proximity.
- Token Variety – There is variation in the outputs generated. The model is identifying input differences well. It is generating effective context vectors.

Next Steps

- Expand the Data Set – I intend to continue finding other sources of annotated code. If the training data set were significantly expanded, I would expect to see improvements in the model's token vocabulary, loss, accuracy, and prediction quality (subjectively speaking).
- Continue Tuning Seq2Seq Model – The model in this report can continue to be tuned, with investigations in how adjusting optimizers, dropout, and additional layers affect the performance of the model.
- Iterate on Business Case – I would like to continue investigating the usefulness of this tool with other developers. In order to accomplish this, I would like to create a web application implementation of the model and get commentary on it.