Programming Language Predictor

Andrew Dircks (abd93) - Samuel Kantor (sk2467)

Description:

We plan to build, train, and deploy a machine learning model to predict the programming language of a code snippet, as a string of text. The open-source dataset containing 97 million code snippets labeled with programming language can be found here. This classification problem is relevant for source control platforms like GitHub, which currently uses a two-layered Artificial Neural Network with some manual feature representation to classify text into its respective programming language. Additionally, text editor tools like Carbon and VS Code can utilize this automatic language detection for real-time formatting and syntax-highlighting.

Our goal is to build a model similar to that of GitHub with the training data linked above. This data is classified by 21 language classifications: `Bash, C, C++, CSV, DOTFILE, Go, HTML, JSON, Java, JavaScript, Jupyter, Markdown, PowerShell, Python, Ruby, Rust, Shell, TSV, Text, YAML,` and `UNKNOWN.` With this optimized and tested model, we will demonstrate its capabilities with a fork of Carbon (an open source web-based code image generator) that detects the user's programming language in real time with the output of our model.

Approach:

The primary AI approach we will use is multi-layered neural networks. The final layer of our model will have an output layer with 21 nodes, representing the activation of each programming language. Feature representation and hidden layer architecture will be the most challenging aspect of this project, and we will experiment with different methods found in the natural language processing literature.

We will use Python with the TensorFlow and Keras machine learning libraries to construct, train, and test our model. Due to the large size of the dataset (97,000,000 entries and 60 GB of data), we will work with a small portion of the data on our personal machines for model prototyping and preliminary testing. Full-scale training and diagnostics will be done on virtual machines and compute clusters that we have access to.

Evaluation Plan:

Quantitative evaluation of our system will be the percentage accuracy of the test data after training. We will train our model on ~80% of the dataset and test the accuracy on the remaining ~20%. The performance on the test data will be our primary measurement of progress and success. We will also use K Fold Validation to help us test if our model will generalise well.
Qualitative evaluation of our model will be the user interface we create. Using our model in real time to make updated programming language predictions will give us a sense of the limitations and practicality of our work.

Implementation Timeline:

Week	Goals
3/22	- Download and visualize subset of data on our machines
3/29	- Literature review of NLP methods and previous attempts
4/5	- Feature mapping and pre-processing in Python
4/12	- Familiarize with relevant Keras/Tensorflow models
4/19	- Begin training and testing on local machines
4/26	- Continue previous week, tune as necessary
5/3	- Full dataset training and testing on VMs
5/10	- User interface (Carbon fork with updated language predictor)
5/17	- Final presentation work