Abstract

Inspired by neural language models introduced in lecture, we decided to design a neural network that predicts types of algorithm & data structure needed to solve a given coding question.

We collect data from competitive coding websites like leetcode, codeforces, and atcoder. Since the data size is relatively small, techniques like data augmentation and transfer learning will be used to achieve better accuracy.

Ideally, given a coding question's text description, the trained model would predict the most likely algorithm for this question and show the several top-ranking labels ordered by possibility.

The neural network framework and detailed implementation haven't been settled down yet, and we'll analyze multiple models including GloVe[1], CNN[2], RNN[3], LSTM[4] to see which model works best for our case.

Introduction

All three of our team members enjoy solving coding questions and consider it as a way for training logic and consolidating programming skills.

We noticed that finding the correct algorithm is always the first step for forming a solution, while a biased identification will lead to miserable turnabouts till you get on the right track.

Therefore, we decided to design an effective Neural Network that can generate possible algorithms & data structure labels for a given coding question based on its text description - a model specialized text classifier for coding questions.

Another reason why we choose this topic is because there are many antecedent researches and frameworks for the field of text classification, so we can do analysis on different architectures then use pretrained models to reduce resources required for training while having better performance.

Related works

- https://towardsdatascience.com/deep-learning-techniques-for-text-classification-78d9dc40bf7c
- Deep Learning--based Text Classification: A Comprehensive Review
- Deeplearning Model Used in Text Classification

Method / Algorithm

1. **GloVe Embedding**
   We first came up with GloVe - an unsupervised learning method that doesn't need intensive computational resources.
   For this method, we'll first construct a dictionary of words that contains all words that appeared in training data.
   Then, we do PCA on this vocabulary base to construct the word-embedding matrix for the model.
   Afterwards, with the word embeddings, we construct a NN with multiple hidden layers for transforming from word embedding inputs to softmax possibilities, where we can get the final probability rankings of all algorithm labels.
   We'll first train on small size-data and fine tune all hyperparameters including number of principal components from PCA, model structure, learning rate to get the best combination.
   Once we've finished tuning, it's time to train the full-sized model.
2. **RNN**
   In addition to word embedding, we decided to explore the possibility of RNN-based structures, like pure RNN, LSTM, and transformer.
   Here, we'll still use word embedding matrix to vectorize each word, and input the words sequentially into the model.
   For the general structure, it'll be word embedding in the beginning, MLP in the middle, with specified structure and algorithm depending on different structures of RNN, and layers of fully connected nodes with softmax in the end for final output.
   We'll start from simple RNN to more complex architectures, and see if a more sophisticated model will do better.
   Still, not much detail has been settled down and we'll be doing extensive testing and research in following weeks.

Reference

1 Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

[2] Fukushima, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biol. Cybernetics* 36, 193–202 (1980). https://doi.org/10.1007/BF00344251

[3] McCulloch, W.S., Pitts, W. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics* 5, 115–133 (1943). https://doi.org/10.1007/BF02478259

[4] Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.