What Defines a Good Stack Overflow Answer Post,

An Automated Post Rater

1. Abstract

Major Q&A sites for professional and enthusiast programmers, like Stack Overflow, have gained phenomenal popularity during recent years. Programmers are relying on these sites to seek inspiration, find solutions and help others. However, as all of the common open online communities, the level of professionality, correctness, and sophistication varies among different posts and users. In this project, we want to build an automated agent that can mark the “good” answers to a question. A feature extractor is first used to preprocess the data. Then, we apply logistic regression and neural network to train our model. With our first implementation, we can get 65%~70% prediction accuracy when measured against our metric[[1]](#footnote-1).

2. Introduction

As there are many desired qualities that define a good answer. We build our own feature extractor to gauge our data. We consider features.

Other than the domain knowledge related features mentioned above, we also considered the overall style and fluency of an answer. We trained a unigram and a bigram language model using exemplary text that matches the overall writing style of a good answer post. An n-gram sequence model is a function that, given n consecutive words, provides a cost based on the negative log likelihood that the n-th word appears just after the first n-1 words. The cost will always be positive, and lower costs indicate better fluency.

Then, the extracted features are feed into a logistic regression model and a neural network model. The prediction accuracy is then defined as the percentage of correctly rated posts measured by whether the post is marked as “accepted” by the question owner.

3. Related Work

4. Dataset and Features

We scrape all the posts generated by users on Stack Overflow during the period of 2018.04.01~2018.05.01 as our training data, posts generated by users on 2018.03.01 as validation data, and posts generated by users on 2018.06.01 as test data. In total, we have 14504 training data points, 600 validation data points, and 486 testing data points. For the training on neural network, we used *sklearn.preprocessing.StandardScaler* to perform standardization on our data to speed up the training process and improve numerical stability. Here is one of the training data points we used during training:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| unigramCost | bigramCost | parsed\_CommentCount | parsed\_BodyLength | parsed\_UserReputation | parsed\_UserViews | parsed\_UserUpVotes | parsed\_UserDownVotes | InlineCode | BlockCode | BlockCodeLine | Hyperlink | Edit | Label |
| 8.75618004 | 12.1084839 | 0 | 769 | 8366 | 1001 | 1578 | 8 | 7 | 0 | 0 | 1 | 0 | 0 |

Example Training Data Point

5. Methods

Because the automated rater rates each question as “will be accepted” and “will not be accepted”, we need a classification algorithm to train the model.

Logistic Regression

As a foray to improve the performance of our model, we also used neural network as the training algorithm. A neural network is formed by layers of neurons. Each neuron consists a weight vector and an activation function. A neuron takes a vector input, compute the weighted average and transform the result according to its activation function. A layer is formed by stacking multiple neurons. And, lastly, a neural network is formed by stacking multiple layers. Usually, a neuron in layer only talks to the neurons in layer . Forward propagation and backward propagation are used to train a neural network.

Forward Propagation:

Given input , we define . Then for layer , where is the number of layers of the network, we have:

Note: is the activation function.

Backward Propagation:

To develop a general approach for calculating the gradient of loss function with respect to . We need to define:

Then, the three-step “recipe” for computing the gradients with respect to is the following:

1. For output layer , we have:

2. For , we have:

3. We can compute the gradients for layer as:

6. Results and Discussion

Logistic Regression

Neural Network:

There are 6 hyperparameters in my implementation of neural network: hidden\_layer\_size, activation, max\_iteration, regularization, solver, and tolerance. We choose these hyperparameters based on the validation set prediction results. To avoid over-fitting, the general approach we used to choose these hyperparameters is: Choose the simplest thing that works reasonably well. Here is the list of hyperparameters we used:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hidden\_layer\_size | Activation | Max\_iteration | Regularization | Solver | Tolerance |
| (10, 10, 10) | logistic | 1000 | 1e-04 | adam | 1e-06 |

Hyperparameters for Neural Network

The metrics we used to measure the performance of the automated rater are the confusion matrix, precision, recall, accuracy and F1 score.

|  |  |  |
| --- | --- | --- |
| Confusion Matrix | Truth, Good | Truth, Bad |
| Predict, Good | 222 | 99 |
| Predict, Bad | 66 | 99 |

7. Future Work

The way we extract features from the dataset limits the neural network’s capability of discovering new features from the raw data. We need to come up with a new feature extractor that can preserve the semantic information of raw text in our data while still making the data compatible with the neural network.

Contributions

Yanpei Tian: Build the language model to measure “style and fluency” in a post; implement the neural network algorithm.

Reference

[1] *Text Reconstruction*, 8 Nov. 2019, https://stanford.cs221.github.io/autumn2019/assignments/reconstruct/index.html.

[2] “Query Stack Overflow.” *Stack Exchange Data Explorer*, https://data.stackexchange.com/stackoverflow/query/new.

[3] “Learn.” *Scikit*, https://scikit-learn.org/stable/.

[4] “Machine Learning.” *CS229*, http://cs229.stanford.edu/syllabus.html.

1. Our definition of a “good” answer: The answer marked as “accepted” by the person who asked the question. [↑](#footnote-ref-1)