COMP30027 | Machine Learning

Reflection

1046432 Jeremy(Fu-Sheng) Huang

The process of completing this project

I started off with the dataset by closely inspecting part of the train and test recipe files. With the help of a frequency graphs, I better understood the dataset on the macroscale. Then, I studied related work to learn what the common approaches were for the project. Multiple ways of representing text have been tried. The nature of text has been considered when determining which model to use.

Each potentially applicable model was fine-tuned and compared. Despite improvement in performance, the LinearSVC is generally costly to train. I avoid overly complicated models to balance training time and performance. Lastly, I built a naïve LinearSVC model using for my result.

Things that you are satisfied with

I am satisfied that my models have done the performance pretty well in the result. This enabled us to compare and analyze the performance difference on various models. As a result, I chose TF-IDF to vectorize texts, classifying instances with LinearSVC, which gave a quite high accuracy (77.6%) on the validation set. By trying out different method to do the model training, the DecisionTree I built that decrease the classifier's performance (73.6%).

Things that can be improved

I completely ignore and skip the analysis of other features as I assumed that most steps would be unique. After conducting peer review, I realized the value of stacking models or utilize more features as model training might be overlooked. For example, others' work showed that the ingredients of the recipe might be related to how many steps this particular cooking dish has done. I might be able to extract other information from ingredients, which can be added as features to my classifiers.

In my report, I evaluate the effectiveness of TF-IDF model. I did not perform much feature selection on generated vectors as the SVCs scale well with high vector spaces. However, I could potentially boost the performance if proper feature selection has been done and I did more preprocessing before training my datasets. These words with low frequency could be ignored as they are likely to be noise.

For the other people do stacking model, I only used a fairly simple approach. Both pre-processing and training were attempted with limited improvement. For example, my LinearSVC model. Then, a more complex classifier may deliver a better performance over my existing approaches. In the DecisionTree approach, only one parameter is used in the model, max_depth. Introducing more parameters and have more comparison with higher max_depth such as a random_state might let the model observe more reasonable connection between text, which has potential to improve performance.