# Code Challenge

Michael Jin

October 2024

## 1 Introduction

I ensemble the KNN and SVM classifier model to do a topic classification task on "Yahoo! Answers" dataset.

## 2 Data Preparation

The dataset I choose is the "Yahoo! Answers" dataset, and its link is https://huggingface.co/datasets/community-datasets/yahoo_answers_topics.

The focus question is to identify which topic each question belongs to. It is a text topic classification problem in NLP. Since the original size of the dataset for Yahoo Answers is extremely large (1400000 data points in training set), for saving the time and the gpu compute units, I only randomly sampled out 1/100 data from the original dataset.

## 3 Data Cleaning

Since I am going to do a natural language process, I define a function which will help remove unnecessary part of the sentences like numbers and stop words; it will also do word lemmatization which can improve the generalization ability for the model.

## 4 Convert Words to Vectors Representation

Since the models require numeric input, we have to convert the words into vector representation. Here, I use TF-IDF to vectorize the texts.

TF-IDF, stands for Term Frequency-Inverse Document Frequency, is used in NLP to evaluate the importance of a word in a document relative to a corpus, and can generate vectors representation for them.

The formulas for TF-IDF are listed here:

$$\mathrm{TF}(t,d) = \frac{\#\text{ of } t \text{ appears in } d}{\text{total terms in } d}$$

$$\mathrm{IDF}(t,D) = \log \frac{\text{total number of } d \text{ in corpus}}{\#\text{ of } d \text{ contain } t}$$

where $t$ is the term and $d$ is the document;

Finally, we can calculate TF-IDF by

$$\text{TF-IDF}(t,d,D) = \mathrm{TF}(t,d) \times \mathrm{IDF}(t,D).$$

Explanation:

For TF, it calculates the relevance of a term within the specific document, while IDF measures how important a term is across the entire corpus. The goal for TF x IDF is just to find the word that is frequently appear in document but not in whole corpus. This provides the "feature" words for the model to learn.

Since in one document, it will contain only a small proportion of the entire corpus, the final vectors for a document after TF-IDF calculation will be a high dimensional sparse matrix. This is also a very good input for SVM classifier using linear kernel.

## 5 The Classification Task

I choose to ensemble K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) with linear kernel to do classification task in this code challenge. This is because for text topic classification task, KNN is very interpretable and intuitive. Based on the number of neighbors in vector space, we can predict the class easily. The cosine similarity is also very useful when dealing with the data point in vectorized words space. SVM is also a very useful and common model in topic classification tasks; since we convert word to high-dimensional sparse vectors, SVM can find the hyperplanes to classify the topics. To combine their advantage, I use the method of ensemble learning that combine the predictions by both KNN and SVM to generate a final result.

KNN The core idea of K-Nearest Neighbors (KNN) is to classify a sample based on the classes of its nearest neighbors in the feature space. KNN works by finding the $k$ closest data points (neighbors) to the input sample and assigning the most common class among these neighbors as the predicted class.

For the Yahoo dataset, after converting the text data to IF-IDF vectors, we can apply KNN in the high-dimensional sparse matrix space. Here we can choose cosine similarity as the metric for measuring distance, since those vectors are high-dimensional and sparse.

The mathematical mechanism is described below: For a given query point $\mathbf{x}$, the cosine similarity to a point $\mathbf{x_i}$ in the training set is calculated as

$$\mathrm{d}(\mathbf{x}, \mathbf{x_i}) = \cos < \mathbf{x}, \mathbf{x_i} >= \frac{\mathbf{x} \cdot \mathbf{x_i}}{||\mathbf{x}|| \cdot ||\mathbf{x_i}||}$$

Finally, the probability value for the class to be $c$ is defined as

$$\mathrm{P}(y = c) = \frac{n_c}{k}$$

where $k$ is the number of neighbors and $n_c$ is the number of neighbor points that is belong to class $c$.

SVM The core idea for it is to find a best hyperplane to classify samples in different classes. The hyperplanes are chosen based on maximizing the margin of classifying boarder, which is the minimum distance between hyperplane and the support vector.

For the yahoo dataset, after converting to vectors, we assume they are linear classifiable since all vectors are high dimensional sparse matrix.

The mathematical mechanism is described below:

For a SVM on a binary classification task, assume there is a hyperplane in space:

$$\mathbf{W}\mathbf{x} + \mathbf{b} = 0$$

where $\mathbf{W}$ is the normal vector (which decides the direction of the hyperplane) and $\mathbf{b}$ is the bias. In the specific scenario of this code challenge, $\mathbf{x}$ will be the vectors convert by TF-IDF method of the question. For a given point $\mathbf{x}_i$, its class will be determined by

$$y_i = \mathrm{sign}(\mathbf{W}\mathbf{x}_i + \mathbf{b}),$$

where 1 for positive class and -1 for negative class. This means that the boarder of the classification is just

$$\mathbf{W}\mathbf{x} + \mathbf{b} = \pm 1.$$

SVM is trying to maximize the margin $\frac{2}{||\mathbf{W}||}$ between two classes. Therefore, these things are equivalent to do the optimization problem of

$$\min_{\mathbf{w}} \frac{1}{2}\|\mathbf{w}\|^2$$

$$y_i \cdot (\mathbf{W}\mathbf{x} + \mathbf{b}) \geq 1, \forall i$$

After solving for the best parameters (such as through the basic Lagrange multipliers and SMO algorithm), we than get a binary classification on two classes. For the multi-class classification task here, we can just choose a total number of $\binom{N}{2}$ support vector machines to do binary classification on any two classes.

However, here we will use ensemble learning to combine KNN and SVM, meaning that we have to convert the result from SVM to probability value. This can be done by Platt Scaling.

Specifically, if the output by SVM (the distance to the best hyperplane) on data point $\mathbf{x}$ is $\mathrm{d}(\mathbf{x})$, we can use the mechanism similar to logistic regression to convert to probability value as follows:

$$\mathrm{P}(y=1|x) = \frac{1}{1+\exp(\alpha \cdot \mathrm{d}(x)+\beta)}$$

where $\alpha$ and $\beta$ is the parameter trained from training dataset by MLE algorithm.

Ensemble

I use the ensemble learning method to consider the classification results for both KNN and SVM. Specifically, I use voting classifier to let KNN and SVM vote for the predictions. The Voting Classifier with soft voting uses the probabilities predicted by each base model, rather than the predicted class labels, to make the final decision. In this case, the predicted class is the one with the highest average predicted probability across all classifiers. Soft voting is beneficial when the base models can output well-calibrated probability estimates, as it allows the ensemble to account for the confidence of each model in its prediction, often leading to improved performance compared to hard voting, which simply counts votes without considering the model's confidence.

# 6 Results Evaluation

For this topic classification task, I mainly use two metrics to evaluate the result. The first one is the accuracy score, and the second one is the roc-auc score. The final scores for them is 55.17% for accuracy and 0.87 for roc-auc score. These two evaluation metrics are very suitable for classification task, since they tell you how accurate the model predicts.

In the context of text topic classification, accuracy score represents the proportion of documents that the model correctly assigns to the correct topic category. A high accuracy means that most of the documents are being classified into the correct topics.

For my ensembled model, the final accuracy score is 55.17%.

Compared with the leader board in huggingface, there is still a big gap from this ML model to other deep learning model in NLP. The best accuracy result on Yahoo Answers dataset is 77.62% by BERT. However, I find that the result has already been very close with the Seq2CNN model. And I believe that if we have enough computing units and train the ensemble model in full dataset instead of sample out just 1/100, the final accuracy can be further improved to $\tilde{6}0\%$.

And ROC-AUC evaluates how well the model can distinguish between different classes (regardless of classifying threshold). A higher AUC value indicates that the model has a good ability to discriminate between different topics.

Use this KNN & SVM ensemble model, the final roc-auc score is 0.87. It means that if we randomly choose a data point from positive example and one from negative example, then it has a probability of 87% to rank the probability of data points belong to positive class of positive one larger than the negative one. It shows that our model has very good ability to distinguish between classes.