Make a classification of sentimental label for movie reviews

s2302674 **s2346729** **s2440788**

Abstract

We intend to make a classification of the movie reviews which contains 11855 movie reviews collected from Rotten Tomatoes using several machine learning methods, then we will compare their performance and conclude which method performs better, according to the accuracy of the prediction. We use supervised models (such as Naïve Bayes, Neural Networks models) and unsupervised model (K-means) to make classification on the sentimental label of each sentence. Some problems have been found and further studies have been determined.

1 Instructions

1.1 Description of the Task

In this project, we intend to make a classification of the movie reviews which contains 11855 movie reviews collected from Rotten Tomatoes using several machine learning methods, then we will compare their performance and conclude which method performs better. The methods that we are going to adopt are naive bayes classifier, neural network and unsupervised learning.
Inspired from practice and experience that apply Naive Bayes classifier on a set of labeled emails to detect if a new email is spam or not, it is intuitive to reach a proposition that Naive Bayes is one possible way to classify movie comments. It is found that multiple attempts that classify comments using Naive Bayes classifier have been made, which also supports our proposition.
The neural network models have a state-of-the-art performance on various areas such as image recognition, speech processing, also include natural language processing. [2] Therefore, using neural network model to predict the level of each sentence is feasible.
Comparing with supervised learning, unsupervised learning do not need to label the original data when we make prediction or classification, it is a good way to reduce the cost in application. We will explore if unsupervised learning model can classify different sentiment well.

1.2 Relevant Background and Related Previous Work

There has been multiple attempts of applying Naive Bayes classifier on classifying comments. In 2019, riyoko and Yaqin made use of Naive Bayes classifier to detect spam comments on Instagram and proved its capabilities on NLP problem by accuracy up to 80% [5]. At the same year, Muhammad and his colleagues proposed that Naive Bayes classifier can be used with Support Vector Machine to combine their advantages[4]. As early as 2014, Yoshida and his team have applied Naive Bayes classifier to detect flaming message through SNS media[9].
Neural networks are the set of powerful machine learning models including 1D convolutional neural networks, recurrent neural networks, conditioned-generation models, and attention-based models. These models promote the development of different application such as machine translation, syntactic parsing. [3]
K-means clustering[7] is a classical clustering method. To train NLP model, we need to translate nature language into code which the model can train. TF-IDF[8] is method to select the key words of the dataset.

1.3 Significance of the Task

Movie comment is considered to be an important criteria when people decide if a movie deserve watching. However, rating a movie from comments can be a massive and tedious work. Inspired from it, the idea of evaluating a movie based on machine learning approaches is proposed and its capabilities is evaluated. Using different model to make sentimental analysis on the Stanford sentimental treebank is meaningful, since various models could have distinct performance on various aspects, such as training speed, accuracy, generalization. Considering performance on these points could optimize the sentimental analysis further.

2 Exploratory Data Analysis

2.1 Description of the Dataset

The dataset 'stanford Sentiment Treebank' provided by tutors contains 8 files. The following 6 files are important:
- **dataetSentence.txt**: include 11855 film comments and their sentence_index.
- **datasetsplit.txt**: split the whole dataset(11855 film comments) into 3 sub-dataset for training, testing, and validation.
- **dictionary.txt**: include 239232 phrases and their phrase index.
- **sentiment_labels.txt**: include 239232 phrase indexes and their corresponding sentiment values. The sentiment values' range is from 0 to 1. [0, 0.2], (0.2, 0.4], (0.4, 0.6], (0.6, 0.8], (0.8, 1.0] represent very negative, negative, neutral, positive, very positive, respectively.
- **STree.txt**: row indexes in this file are corresponding to sentencen index. Every data record in this file represent a sentence. By these records, every sentences are structure as trees. Every nodes in a tree represent different phrase in dictionary.txt file. We will describe it in detail in next part.
- **SOStr.txt**: row indexes in this file are also corresponding to sentencen index. Records need to be used with 'STree.txt'.

2.2 Sentence Structure

By reading the paper link to stanford Sentiment Treebank dataset, we find that, records in STree.txt represent a tree. In computer programming, trees are constructed by nodes.The value in index in ith place represent the parent node for node i.
Here we will use record index 134 as an example to describe (as it is a short sentence). The record 134 in 'STree.txt' is: [12|11|11|9|8|8|10|9|10|13|12|13|0], and the corresponding record in 'SOStr.txt' is: [Dramas|like|this|make|it|human|.]

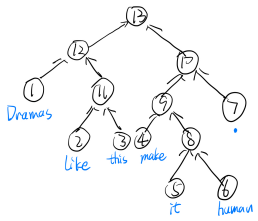

Figure 1: Tree_134

In this example, node1's parent is 12, node2's parent is 11, node3's parent is 11, node4's parent is 9... Therefore, we can make a tree depending on the code. We found that node1 to node 7 is leaves of this tree, every leaves represent a single word correspoding to SOStr.txt file's record, which is [Dramas|like|this|make|it|human|.] in this example. The tree 134 is constructed like Figure 1 below. Every node in this tree, include leaves, root and medium parents, represent the combination of 1 or few words(or punctuation) split by 1 space. And these combinations are those phrases in dictionary.txt file. Each node or phrase have a sentimental value.

2.3 Data Preparation Overview

We did some work to split the dataset 2 aspects. The first is based on datasetsplit.txt file. We can choose different data in train set, test set or validation set. The second is the based on level of tree. We can choose dataset consist single words(leaves), total sentences(roots) or other phrases(parent nodes) and their corresponding label.
For those single word(leaves), we use stop words list to clean the words which will influence the performance of model(like a,that). For phrases which consist more than 1 word, as some word maybe have no useful meaning when they are just word, however have sentiment tendency when they are in phrases, therefore, we do not choose to clean stop-words in phrases.

3 Naive Bayes

3.1 Description of Naive Bayes

Naive Bayes classifier is a kind of "probabilistic classifier" which applied Bayes theorem assuming that there exists strong independence between features.

3.2 Data Preparation

Considering this is a NLP project, general procedure is adopted. As this project aims to label a sentence based on single word it consists of, the first step is to build a dictionary and filter stopword. Then we calculated the frequency of each word in each label. After that, the performance of it is evaluated on test set using adapted posterior probability:

$$p(label = x|sentence) = \frac{p(sentence|label = x)p(label = x)}{\sum_{i=1}^{5} p(sentence|label = i)p(label = i)} \tag{1}$$

The probability of each label of given sentence is calculated and the one with highest probability will be predicted label.

3.3 Performance

Initially for test dataset (sentence_dataset3), 115 is predicted correctly and 986 fail. One possible cause of low accuracy is well-known "the curse of dimensionality". After filtering stopword and repeated word, there is still over 17,000 words, which makes it hard to predict unseen data. Then we observed that word labeled 0.5 is likely to be meaningless word, like people's name. We remove all phrase labeled 0.5 and test it again, then we get 118 is predicted correctly and 983 fail(the accuracy is 0.107). Another possible cause is that Naive Bayes classifier may suitable to deal with binary problem. In this sense, simply dividing sentence into "worth watching" and "not worth watching" may be a possible approach (take 0.5 as boundary). It can also be resulted from rough data preparation procedure. Dimensionality deduction techniques like PCA may help.

4 Neural Network

4.1 Description of Neural Network

The initial idea is to create a vocabulary to encoder text into indices (encode each word in one sentence) which will be treated as input to train a multiple-layers model with a bidirectional RNN model with LSTM layer and a dense layer as hidden layers. Except input and output layer, before the hidden layers, there are two layers which are vectorize layer (this layer is used to encode each word in one sentence into the specific index according the vocabulary) and embedding layer (it projects the encoded word into a vector which could show the similarity between each word). And use this model to make a classification(there are total 5 classes: 0,1,2,3,4)

4.2 Data Preparation

There are 11855 sentences with 5 classes, select 80% of the data to train the model and 20% of data to validate. First, setup input pipeline: The original data: Every sentence has one label which is the corresponding class that sentence belongs to.

```
Sentence:  b'"Madonna still ca n\'t act a lick ."\n'
Label: 0
Sentence:  b'"The movie is without intent ."\n'
Label: 1
Sentence:  b'"It \'s coherent , well shot , and tartly acted , but it wears you down like
showing off his doctorate ."\n'
Label: 1
Sentence:  b'"It \'s a solid movie about people whose lives are anything but ."\n'
Label: 3
Sentence:  b'"A smart , compelling drama ."\n'
Label: 4
```

Figure 2: data after removing punctuation

Based on total sentence we have, setup a vocabulary with size 1000 (there are 1000 words in the vocabulary), and maximum sequence length 50 (each sentence will be represented with 50 indices).

Second, create model with input layer, vectorize layer, embedding layer, bidirectional RNN model with LSTM layer, a dense layer and output layer.

```
tf.Tensor(
[152   1  13   2 108   8   1   7 147   8 119 638  17  63 144   1   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
   0   0   0   0   0   0   0   0   0   0   0   0   0   0]], shape=(50,), dtype=int64)
```

Figure 3: encode each sentence

4.3 Performance

```
 Layer (type)                Output Shape              Param #
=================================================================
 text_vectorization_8 (TextV  (None, 50)               0
 ectorization)

 embedding_10 (Embedding)    (None, 50, 40)            40000

 bidirectional_8 (Bidirectio  (None, 100)              36400
 nal)

 dense_19 (Dense)            (None, 20)                2020

 dense_20 (Dense)            (None, 1)                 21

=================================================================
Total params: 78,441
Trainable params: 78,441
Non-trainable params: 0
```

Figure 4: model summary

Then train the model using training data and adjust hyperparameters according to the validation set. However the performance of training model is extremely bad, the accuracy is very low and even the accuracy of validation set keeps larger than that of training set after adjust hyperparameters (such as batch size, the number of layers, activation function, dropout rate .etc) for many times. Then we try to increase the number of Epoch, although the losses of training and validation sets could decrease, the accuracies of them unchanged and the loss could not converge to a stable value.

```
Epoch 10/50
10/10 [==============================] - 1s 77ms/step - loss: -13.2583 - accuracy: 0.2620 - val_loss:
-14.1628 - val_accuracy: 0.2763
Epoch 11/50
10/10 [==============================] - 1s 78ms/step - loss: -14.1538 - accuracy: 0.2620 - val_loss:
-15.0463 - val_accuracy: 0.2763
```

Figure 5: the situation that accuracy of training data is lower than that of validation data

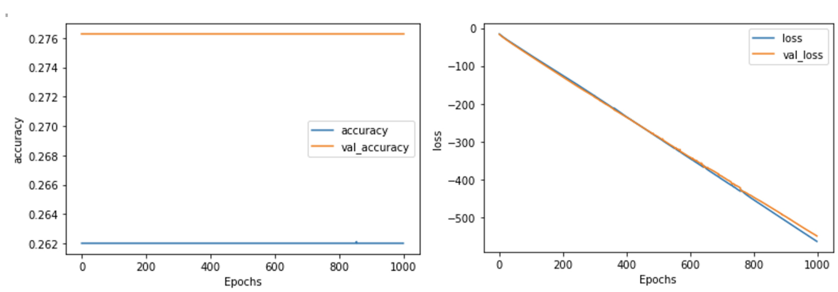

Figure 6: The accuracy and loss of each epoch

Several reasons of guessing: the model is too complex to fit the data; the expression of each word(using indices); the data lost its features when preprocessing the data such that the model could not capture the features of data; there are some problems existing data preprocessing. Then try to decrease the number of layers of neural network. However, the bad situation still keeps. So we select another expression of text is selected, that use each sentimental value of a single word to represent each word for one sentence, like Figure 7.

All of these sentimental values will be normalized before extending the dimension of each sentence into same dimension. After normalizing, the NA will be replaced by 0.

Then create a model with only two dense hidden layers, and train the model. after adjusting several hyperparameters for several times, the units of each hidden layers are 30 and 50, the accuracy of validation set is 0.32425, although it performs better than the model with bidirectional RNN, the accuracy is still lower than 0.5. Totally, this model performs bad on predicting the label of sentence. Trying to increase the layer of model, it seems play a little role on increasing the validation accuracy. The validation accuracy improves to 0.34343 , but it is still less than 0.5. The accuracy of test data is stable around 0.33.

Finally using only one dense hidden layer, and transfer the classification problem into a regression problem, instead of classifying each sentence into 5 level, the sentimental value of each sentence will

	0	1	2	3	4	5	6	7	8	9	10
0	0.65278	0.44444	0.36111	0.5							
1	0.5	0.48611	0.5	0.54167	0.5	0.51389	0.5	0.5	0.52778	0.5	0.52778

Figure 7: another representation of one sentence

be predicted. According to the sentimental value of each sentence, we could know which level the sentence belongs to. Trying to convert the sentimental value into corresponding label (for instance, when 0<=sentimental value<0.2, then this sentence is predicted to label 0), we find that the predictive labels are prone to neutral when the true labels are positive. Finally, the accuracy of validation set is 0.

Overall, the neural network models we setup could not have a well performance on classification of each sentence label. In the future, the data preprocessing will be further checked or change another representation for input data. More data will be needed, and the hyperparameters of neural network models will be continuous to adjust.

5 Unsupervised Learning: K-means Clustering

5.1 Description

For unsupervised learning, we tried 3 different models including K-means[7] clustering, hierarchical clustering[6] and LDA model. However, LDA model and hierarchical clustering both do not have good performance.[1] We cannot find obvious level in hierarchical clustering. In LDA model, we cannot find obvious link between the sentence's topics, corresponding weight(given by model) and the sentiment value(given by dataset). Therefore, we decided to focus on K-means clustering.

In K-means clustering, we will select k centers randomly. Each time when we add a new element into the model, we will calculate the 'distance' from this element to each centers, then choose the nearest centers with, let the center and element be in 1 cluster, and refresh the cluster's center based on all elements in this cluster.

In our work, we tried many times, using different datasets and different ways to encode nature language, and collected results when k = 5, 10, 15, 20 respectively. The way we use to encode nature language include one-hot, tif-df, and encode by sentiment value, in the end, we add a height weight to each phrase. We will describe it in the next section.

5.2 Data Preparation

- One-Hot: At first, we used one-hot method to encode natural language. We make a new dictionary which contains all single words(leaves), and clean the stop words. Then we translate every sentences into vectors, dimension amounts of those vectors is equal to the number of words in the dictionary we made. If words of a sentence also included in dictionary, the value of the vector 's corresponding value is 1, otherwise is 0. However, the performance of this encoding method is not good.

- TF-IDF: We then encode sentence in td_idf methods. We calculate every words' tf-idf values. $tf\_idf = tf \times idf$, $tf = \frac{n}{N}$, n represents the words' frequency in one sentence, and N represents the words' frequency in whole dataset. And $idf = lg\frac{D}{d}$, D represents the number of sentences and d represents how many sentences have this word. We can select the key words in the dataset.

- Encoding by Labels: We label the sentiment values into 0,1,2,3,4. And reset the dictionary we used in one-hot method, we can find a certain label for words in dictionary. Vectors translated from sentence have 5 dimensions represents the 5 labels. If there are 5 words in a sentence belong to label label 0 in the dictionary, and 3 words belong to label 1, 4 words belong to label 2, 7 words belong to 3 and 9 words belong to label 4, then the vector is {5,3,4,7,9}, to avoid the influence from length of the sentence, we divide each value in this vector by the sum of words number, which is $5 + 3 + 4 + 7 + 9 = 28$, and the vector is {5/28,3/28,4/28,7/28,9/28}.

- Adding Height Weight: As sentences in the dataset all be constructed to trees, we decide to add a weight to each data(phrase/sentence/word). We assume that a longer of a long phrases' influence to a sentence is higher than single words. So we calculate weights for each nodes based on this function:$weight = \frac{the\ highest\ level\ of\ the\ node\ from\ leaf}{the\ height\ of\ the\ tree}$. (e.g. in the previous example, Tree_134 node11's weight is 2/5).

- Dataset: We used **dataset1**(training data) first when using one-hot and tf-idf, which contains about 8500 sentences. Considering about we are doing unsupervised learning, after few times failures, we start to use the **whole dataset** which contains about 12000 sentences, futher, we choose to every nodes in the whole dataset(including sentence, leaves and other parents node).

5.3 Performance

- One-Hot(using training dataset's leaves): We can see from the table that, in this result, I choose some k-groups which contains more sentences and do not show those group which only contains less than 10 sentences. The mean sentiment value in each groups are all near to 0.5, and most sentences are arrange to 1 or 2 group. Which means we cannot see different sentiment by k-group.

- TF-IDF(using training dataset's leaves): The result using tf-idf is like to the result using tf-idf, every k-lables' mean sentiment value is near 0.5, and most of sentences are arrange to 1 or 2 k-lables,we can not discriminate different sentiment by different group (divide by k-label).

k=5 cluster	mean	var	len	k=10 cluster	mean	var	len	k=15 cluster	mean	var	len	k=20 cluster	mean	var	len
0	0.57	0.0662	871	3	0.47	0.0728	717	7	0.46	0.0698	870	1	0.47	0.0686	789
1	0.5	0.0623	7409	4	0.51	0.0615	5751	8	0.5	0.0615	996	6	0.52	0.0607	4326
2	0.9	0	1	5	0.49	0.0616	1203	9	0.51	0.0612	5255	10	0.5	0.0654	948
3	0.52	0.0656	262	6	0.57	0.0647	756	13	0.56	0.0729	186	11	0.46	0.0702	330
4	0.86	0	1	7	0.5	0.0523	94	14	0.55	0.0558	246	14	0.52	0.062	1583

Figure 8: One-Hot Result

- Encoding by Labels: Most sentences are also arranged to 1 or 2 groups and we still cannot discriminate the sentiment value by k-groups if only use leaves.

The result using every nodes in the whole dataset displayed in Figure 9(green label:sentiment value; yellow label:k-groups' number), we can see that the k-means groups are much more clearly show the sentiment value than before. For example, when $k = 20$, group k 1 most contains sentiment value 0 and 1, group k 19 most contains sentiment value 4. Although there still have some overlapping, but k groups seldom mix the sentimental value more than 1 level.(e.g. sentimental value 0 an 1 may be arranged to 1 group, much less sentimental value 0 and 1 be arranged to 1 group).

- Adding Height Weight(using whole nodes): After adding height weight, we are surprised by the result that no matter we choose k value equal to 5, 10, 15 or 20, we can perfectly arrange different sentimental value's sentences into different k-groups.

k=20	0	1	2	3	4
0	0.00%	9.62%	9.37%	2.41%	0.70%
1	24.77%	11.24%	4.73%	0.64%	0.16%
2	3.11%	3.25%	4.59%	13.05%	8.26%
3	3.97%	9.27%	10.66%	5.01%	0.65%
4	0.26%	0.38%	0.67%	3.83%	20.36%
5	0.79%	0.48%	10.88%	2.28%	3.13%
6	34.11%	16.46%	6.42%	0.80%	0.05%
7	0.07%	0.06%	0.18%	1.48%	11.02%
8	13.38%	3.66%	1.43%	0.71%	0.32%
9	5.43%	3.79%	4.46%	3.15%	1.57%
10	0.00%	4.62%	2.05%	1.38%	0.43%
11	0.73%	15.22%	13.43%	1.51%	0.38%
12	0.00%	1.15%	2.10%	6.94%	4.75%
13	0.20%	1.15%	2.36%	4.02%	8.15%
14	3.25%	2.23%	5.89%	13.66%	6.86%
15	8.68%	1.78%	0.45%	0.06%	0.05%
16	0.60%	4.36%	8.16%	14.79%	5.13%
17	0.40%	1.62%	4.59%	15.20%	7.99%
18	0.00%	9.17%	6.11%	2.19%	0.81%
19	0.26%	0.48%	1.47%	6.88%	19.22%

Figure 9: Using Whole Dataset's Nodes and Label Encoding

6 Conclusions

We have used several models to classify the sentimental labels of the movie review, all of them have difference performance. Based on our models and data selected, it seems the unsupervised methods could give a relative higher accuracy to make the classification of the sentimental labels of movies reviews. There are several reasons causing several models perform not well, for instance the representation of text contents, the hyperparameters set could not improve the performance of models, and the input data lose the original characters after preprocessing. Therefore, further works needed to be done to increase the accuracy of classification, for example, reduce the dimension of data using PCA, keep tuning the hyperparameters if the models need.

From our unsupervised learning training exploration, we deduce that we can not divide different sentiment of the sentence only by phrases or words. A more feasible methods to classify is using a huge dictionary with sentiment values(or levels). Although we need a dictionary with pairs of data, we only need to input phrase for classification. Besides, a tree's structure is also important, if we use all nodes of the tree, the accuracy will be highly improved. However, in our work, it is unreasonable to use those node which contains the whole sentence or half sentence. We tried to not use nodes which have high weight, the results also improved than only use single word. Therefore, in our further work, we can search a good level to use in clustering model.

References

[1] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[2] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[3] Yoav Goldberg. Neural network methods for natural language processing. *Synthesis lectures on human language technologies*, 10(1):1–309, 2017.

[4] Abbi Nizar Muhammad, Saiful Bukhori, and Priza Pandunata. Sentiment analysis of positive and negative of youtube comments using naïve bayes–support vector machine (nbsvm) classifier. In *2019 International Conference on Computer Science, Information Technology, and Electrical Engineering (ICOMITEE)*, pages 199–205. IEEE, 2019.

[5] Beta Priyoko and Ainul Yaqin. Implementation of naive bayes algorithm for spam comments classification on instagram. In *2019 International Conference on Information and Communications Technology (ICOIACT)*, pages 508–513. IEEE, 2019.

[6] Wikipedia. Hierarchical-clustering. `https://en.wikipedia.org/wiki/Hierarchical_clustering`.

[7] Wikipedia. K-means. `https://en.wikipedia.org/wiki/K-means_clustering`.

[8] Wikipedia. Tf-idf. `https://en.wikipedia.org/wiki/Tf-idf`.

[9] Shun Yoshida, Jun Kitazono, Seiichi Ozawa, Takahiro Sugawara, Tatsuya Haga, and Shogo Nakamura. Sentiment analysis for various sns media using naïve bayes classifier and its application to flaming detection. In *2014 IEEE Symposium on Computational Intelligence in Big Data (CIBD)*, pages 1–6. IEEE, 2014.

Responsibilities

We discuss the problem and determine the researching direction together, figuring out the structure of the dataset.

s2440788 takes responsibility of naive bayes part, s2302674 takes responsibility of neural network part, s2346729 takes responsibility of unsupervised learning part and some basic work in data processing before data were used to train models.