CS182-FINAL PROJECT: Sentiment Analysis of Bullet Screen

Kehao Wang
ShanghaiTech University
2021533025
wangkh@shanghaitech.edu.cn

Zengxu Yu
ShanghaiTech University
2021533183
yuzx@shanghaitech.edu.cn

Ziyang Wu
ShanghaiTech University
2021533032
wuzy1@shanghaitech.edu.cn

Abstract

Compared with traditional texts that are more "rational" and "restrained", bullet screen, as a short text expression under the popularity of new media, expresses the current situation in a "casual" and "feeling-based" real-time comment format. Video users' immediate thinking and cognition and immediate emotional tendencies have stronger emotional color, timeliness and research value. In this project, we will use different machine learning algorithms for classification, record the time consumption and accuracy of different classification algorithms, and compare the advantages and disadvantages of various algorithms. Finally, we will also put forward some prospects for the future development direction of bullet screen sentiment analysis.

1 Introduction

Massive bullet screen data contains users' subconscious behavioral cognition and rich emotional value. By conducting emotional analysis on bullet screens, on the one hand, it can help content creators check the acceptance of their works, and on the other hand, it can also improve the quality of their works for better results. There are two main types of methods for sentiment analysis of bullet screens, one is based on sentiment dictionary and the other is based on machine learning.

2 Methodology

2.1 k-nearest neighbors (k-NN)

Suppose we have pairs $(X_1, Y_1), (X_2, Y_2), \ldots, (X_n, Y_n)$ taking values in $\mathbb{R}^d \times \{1, 2\}$, where Y is the class label of X, so that $X \mid Y = r \sim P_r$ for $r = 1, 2$ (and probability distributions P_r). Given some norm $\|\cdot\|$ on $\mathbb{R}^d$ and a point $x \in \mathbb{R}^d$, let $\left(X_{(1)}, Y_{(1)}\right), \ldots, \left(X_{(n)}, Y_{(n)}\right)$ be a reordering of the training data such that $\left\|X_{(1)} - x\right\| \leq \cdots \leq \left\|X_{(n)} - x\right\|$.

2.2 Naive Bayes classifier

Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from

some finite set. Naive Bayes classifier assumes that the value of a particular feature is independent of the value of any other feature, given the class variable.

2.3 Support Vector Machine (SVM)

We are given a training dataset of n points of the form $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ where the y_i are either 1 or -1 , each indicating the class to which the point $\mathbf{x}_i$ belongs. Each $\mathbf{x}_i$ is a p-dimensional real vector. Computing the soft-margin SVM classifier amounts to minimizing an expression of the form

$$\left[\frac{1}{n}\sum_{i=1}^{n}\max\left(0, 1 - y_i\left(\mathbf{w}^\top \mathbf{x}_i - b\right)\right)\right] + \lambda\|\mathbf{w}\|^2.$$

Choosing a sufficiently small value for λ yields the hard-margin classifier for linearly classifiable input data. The classical approach, which involves reducing it to a quadratic programming problem.

2.4 Maximum entropy model (MEM)

The maximum-entropy model assumes that the network state probability distribution is given by an exponential function of the network energy $E[S_i] = E[(x_1, \ldots, x_n)]$ such that entropy is maximized while satisfying any statistical constraints. When first and second-order statistics are given, the state probability distribution is given by

$$P(x_1, \ldots, x_n) = \frac{1}{Z}\exp\left(-E\left[(x_1, \ldots, x_n)\right]\right) = \frac{1}{Z}\exp\left(\sum_i b_i x_i + \sum_{i \neq j} J_{ij} x_i x_j\right).$$

2.5 Gated Recurrent Unit (GRU)

The basic idea behind GRU is to use gating mechanisms to selectively update the hidden state of the network at each time step. The reset gate determines how much of the previous hidden state should be forgotten, while the update gate determines how much of the new input should be used to update the hidden state.

Reset gate: $r_t = \text{sigmoid}\left(W_r{}^*\left[h_{t-1}, x_t\right]\right)$, Update gate: $z_t = \text{sigmoid}\left(W_z{}^*\left[h_{t-1}, x_t\right]\right)$,

Candidate hidden state: $h_t{}' = \tanh\left(W_h{}^*\left[r_t{}^* h_{t-1}, x_t\right]\right)$, Hidden state: $h_t = (1 - z_t)^* h_{t-1} + z_t{}^* h_t{}'$

where W_r, W_z, and W_h are learnable weight matrices, x_t is the input at time step t, h_{t-1} is the previous hidden state, and h_t is the current hidden state.

3 Experiments

3.1 Dataset and Preprocessing

During the preprocessing process, we used the jieba library to split the bullet screen sentences into words one by one, and removed irrelevant punctuation marks and special symbols.

We first used Python's third-party library SnowNLP to analyze the content of the bullet screens. But the analysis result is not good. Therefore, our team decided not to use SnowNLP for sentiment analysis, but to use it to assist us in labeling the data set to save workload.

We chose to use the three labels "-1, 0, 1" to mark the three types of words: negative, neutral, and positive respectively. After using SnowNLP to mark the first round of data sets, we manually checked the marked data and modified the incorrectly marked word labels. In the end, we obtained more than 17,000 correctly labeled bullet screen data.

3.2 Use Gensim's Word2Vec model to convert word vectors

The essence of the Word2Vec model is to use a shallow neural network to embed words into a low-dimensional vector space. The result is a set of word vectors, where vectors that are close together in vector space have similar meanings depending on the context, while word vectors that are further apart have different meanings.

During the experiment, we used the Word2Vec model for training to prepare for subsequent classification work. Firstly, import the corresponding library and load the labeled data. Secondly, set the model parameters (such as word vector dimensions, window size, minimum word frequency, etc.), and then use the model to train and process labeled data into word vector data. Finally, save the trained model.

3.3 Classification Experiment

We manually completed the implementation of five algorithms such as k-NN, Naive Bayes classifier, SVM, MEM and GRU, and applied them to the sentiment classification of bullet screens.

3.4 Results

Table 1: Classification results of dataset 1 with 1,298 bullet screen data by different algorithms

Method	k-NN	Naive Bayes	SVM	MEM	GRU
Accuracy	63.94%	40.90%	51.42%	49.11%	65.59%

Table 2: Classification results of dataset 2 with 1,611 bullet screen data by different algorithms

Method	k-NN	Naive Bayes	SVM	MEM	GRU
Accuracy	63.63%	48.54%	63.56%	67.04%	68.47%
Runtime(s)	7.6	4.7	5.1	4.4	1.4

3.5 Analysis

Dataset 1 and training set were randomly selected from the same dataset. This dataset was sourced from multiple videos of "ups" such as Muyu Shuixin, Luo Xiang, CCTV News and so on. Dataset 2 is sourced from a video episode of the "up" Lao Fanqie, and there are no comments from Lao Fanqie's video in the training set and dataset 1. On dataset 1, GRU and KNN perform significantly better than other algorithms. Compared with dataset 1, on dataset 2, all algorithms except KNN have shown performance improvement. This shows that Dataset 2 is easier to classify compared to Dataset 1, as Dataset 1 comes from multiple videos and is more complex, while Dataset 2 is relatively simple.

For k-NN The classification performance of KNN is very good, mainly because comments generally have high repeatability. For the data in the test set, if there is similar content in the training set, the accuracy is very high. On dataset 2, which is easier to classify, the performance of KNN did not improve, which to some extent indicates that its generalization ability is insufficient. If you want to use KNN for sentiment classification on a wider range, it requires a lot of data, because many bullet comments contain some "memes" that are specific to their videos and up creators, and KNN performs poorly on these bullet comments.

k-NN algorithm time complexity and space complexity is high. There will be two problems: a). When the samples are unbalanced or repeated, if a sample is input, most of the k neighboring values will be the class with a large sample size, which may lead to classification errors; b). Each sample to be classified must calculate the distance from it to all points. Only by sorting the distance can the k neighboring points be obtained, which requires a large amount of calculation.

For Bayes classifier The emotion of bullet screens is different from other data, and its separation boundaries are relatively fuzzy. As well, in this scenario, each feature is not completely independent, but has a strong correlation, which is inconsistent with the assumption of naive Bayes. This explains the poor classification performance of the Bayesian classifier.

For SVM Simple SVM cannot solve linearly inseparable problems, but emotional classification is linear and indivisible. In the experiment, previous SVM classification accuracy was low. After adding the kernel function, the accuracy has been improved to a certain extent. However, the effect is still not good, possibly because the problem is too complex and the selected kernel function is not suitable enough.

For MEM Through the maximum entropy model, each sample only needs to provide single-digit feature information. It breaks the assumption of homogeneity and introduces characteristic functions, allowing model predictions to consider more complex features instead of only the state of the previous time step.

For GRU GRU is a variant of LSTM, a network structure that can analyze data with time series, thus describing the correlation between preceding and following words. Therefore, LSTM performs the best. Besides, the model is simple, has fewer parameters, so is faster to train.

3.6 Improvements

For k-NN a). Weight k nearset points (points close to each other have a large weight and points far away have a small weight); b). Process known samples and remove duplicate samples that have little effect on classification in advance.

For Bayes classifier For the situation where the probability of a certain class is 0 during calculation, the Laplacian smoothing method can be used to initialize the occurrence number of all words to 1 and initialize the denominator to 2.

For SVM According to different situations, choose the appropriate kernel function for SVM (such as polynomial kernel, Gaussian kernel, etc.).

In general We can perform some data augmentation, specifically adding irrelevant content to a piece of data without changing its category, thereby helping the model learn key content better. Unfortunately, data augmentation is difficult to automatically complete because unlike image data, bullet data is very subtle. Even if a punctuation mark is changed or the word order is slightly changed, the semantics may be completely different. Therefore, we mainly rely on manual data augmentation. The improvement after data augmentation is not significant.

Many bullet comments contain homophones, but our word vectors we use cannot describe this relationship. For example, comments use "birth" instead of "animal". If the model learns that "birth" carries negative emotions, words associated with "birth" will be affected. Therefore, we need a large amount of barrage corpus to train word vector models. Unfortunately, it is difficult to find these data.

In addition, bullet comments have a strong correlation with video content, as well as with the main content of the app. Many of the content may be joking, but if analyzed separately, different results will be obtained. Therefore, we believe that image analysis techniques can be combined with more data to analyze barrage emotions from a broader perspective.

4 Conclusions and Future Directions

4.1 Conclusions

This experiment mainly carried out two key tasks. The first is to use the Word2Vec model for word vector conversion and training on the processed bullet screen sample data. This part takes a long time and requires a lot of work.

The second is to use different classification algorithms to classify bullet screen emotions. In this part, we mainly implement the algorithm, and then analyze and improve the final classification accuracy.

4.2 Future Directions

By classifying bullet screens according to three types of emotions, we can easily get the proportion of various emotions in all bullet screens of a video. So we can explore the following two directions.

Video theme analysis: Does the emotion in the bullet screen remain consistent throughout the video? Or does it change as the video plot develops?

Video recommendation: Incorporate users' personal bullet screen sentiment analysis results into the recommendation algorithm to facilitate the platform to adjust recommended content.

5 References

1. Numpy: Help the implementation
2. GENSIM official tutorial
3. Text processing library SnowNLP introduction and application
4. L. Liu and S. Zhao, "Bullet Screen Short Text Sentiment Analysis Algorithm," in 2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), Shenzhen, China, 2020 pp. 562-568.
5. Sentiment Analysis: An ERNIE-BiLSTM Approach to Bullet Screen Comments