Artificial Intelligence (CS181) Final Project: Twitter Emotion Classification

Yiteng Xu[1,†], Ziyao Zeng[1,†], Jirui Shi[1,†], Shaoxun Wu[1,†], Peiyan Gu[1,†],
[1]ShanghaiTech University
{xuyt1,zengzy,shijr,wushx,gupy}@shanghaitech.edu.cn
[†]Equal contributions

Abstract

Text classification has been an appealing task due to its extensive usage in multiple fields. Specifically, in Twitter, classifying each sentence with one emotion would be a worthy task that could be applied in chatbots, public sentiment monitoring, and suicide prevention. Currently, Naive Bayes-based methods, like Bags-of-Words, and DNN (deep neural network) based methods, like RNN, LSTM, and Transformer, have been widely employed and achieved satisfying performance. In this project, we aim to implement Naive Bayes-based methods and DNN based methods with some adaptations like TF-IDF, and provide throughout ablation studies, visualizations, and error mode analysis. Overall, we obtained comparable results in this task with existing methods.

1. Introduction

Text classification has been a baseline task in Artificial Intelligence and Natural Language Processing for a long time but remains challenging. When it comes to Twitter emotion classification, the challenge gets tougher due to its lack of long-range context information and the ambiguity of keywords in Twitter. Back to the last century, Bags-of-Words models have been quite well-developed and popular in text classification like [5] and [6]. They are first-order probabilistic models under the Naive Bayes assumption, which assumes that all words of the given sentence are independent of each other given the class of the sentence. With the uprising of deep learning, RNN based methods have been comprehensively employed in text processing like LSTM[4] proposed in 1997, and are still being actively used in machine translation and chatbots contemporarily. In 2017, Transformer[10] has been proposed to model massive sequence data with significant improvement compared with previous methods.

In this project, we aim to deal with Twitter emotion classification using SemEval 2018 (Emotion Recognition)[8] sorted in TweetEval dataset[1]. It consists of 3257 train sentences, 374 validation sentences, and 1421 test sentences. Each sentence has 10-20 words and is labeled with one emotion from 4 classes: anger, joy, optimism, sadness. As you could see, a sentence is quite short which means long-range context-dependency could not be leveraged to eliminate the ambiguity of words.

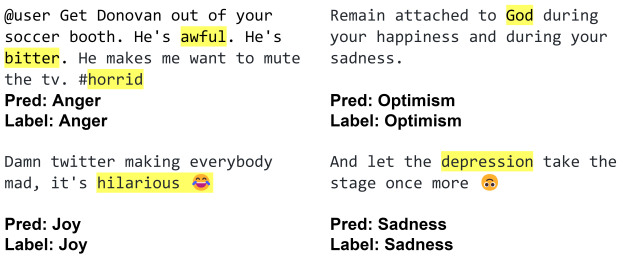

Figure 1. Correct classification, inferred with Naive Bayes + MLE on SemEval[8], highlighted words are the keywords that help the model with correct decisions.

When using probabilistic models to model emotion keywords' frequency, massive ambiguity could occur due to numerous irony and sarcasm existing in Twitter. In this way, we implement TF-IDF(term frequency-inverse document frequency) to exact key words for a better-generalized inference. Besides, to boost its effectiveness, we also implement lots of engineering adaptations like Laplacian smoothing, thresholding infrequent words to be unknown to avoid overfitting, and forming word-groups histogram instead of a single word to model context information. Besides probabilistic models, connectionism models are also effective in this task. We implement DNN methods including vanilla RNN, LSTM, and Transformer. However, due to the short length of Twitter sentences and the small amount of the given dataset, these methods haven't shown their superiority compared with probabilistic models.

We summarize our contributions as below:

- We implement the Naive Bayes method and DNN

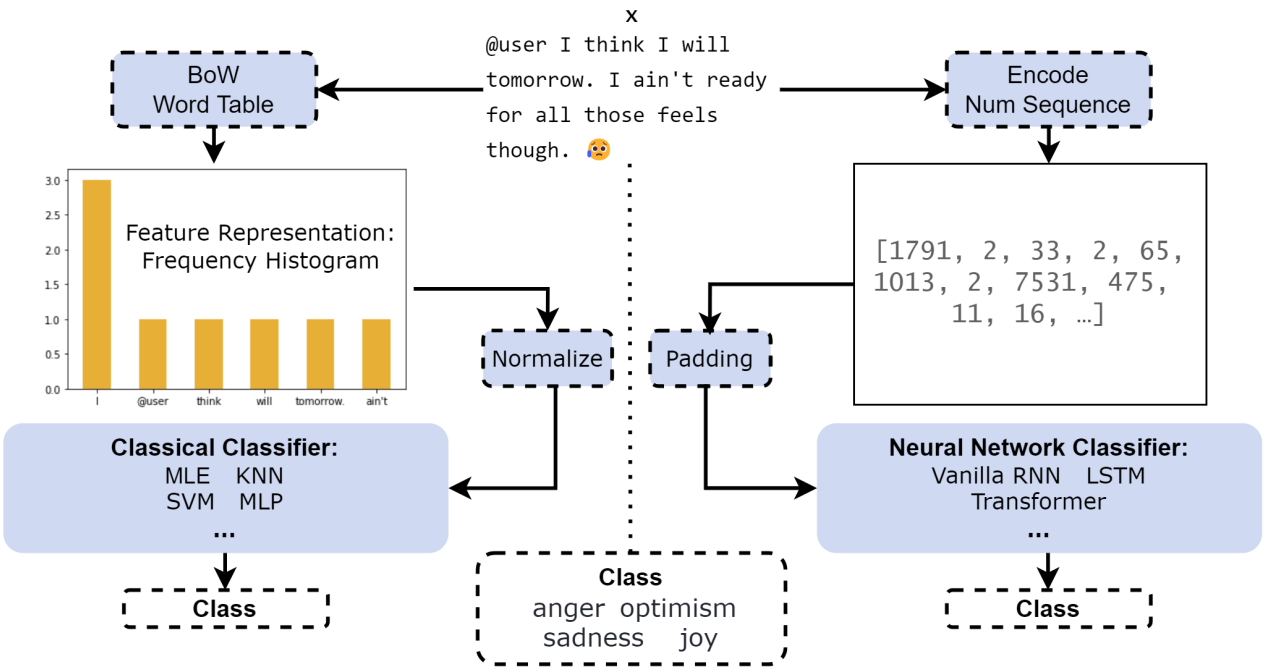

Figure 2. Illustration of our framework which implements Naive Bayes-based methods and DNN based methods with some adaptations like TF-IDF. The left-hand side are the classical methods we used, right-hand side are the neural network-based methods we used.

methods including vanilla RNN, LSTM, and Transformer in the Twitter emotion classification task and show our comparable implement results. Pipeline are shown as Figure 2.

- We implement lots of adaptations in the Naive Bayes method. In this way, we could exceed the Naive Bayes method by a large margin.

- We provide throughout ablation studies, visualizations, and error mode analysis to demonstrate the effectiveness and potential limitations of every component we implemented.

2. Related Work

Naive Bayes Methods. The Naive Bayes assumption assumes all features of the given data are independent of each other given the label. In terms of Natural Language Processing, it assumes that all words are independent conditional on the label of the sentence. By calculating the frequency of words with their corresponding label in training data, a probabilistic model could be built for classification and its effectiveness has been shown in [5] and [6]. Due to its simplicity and explainability, the Naive Bayes method has been nominated as one of the top 10 algorithms in data mining[11]. Though its popularity and effectiveness, directly applying it to Twitter emotion classification suffer for overfitting and lack of generalization ability. Therefore, adaptations have to be made aiming at this task.

Deep Neural Networks. Deep neural networks have shown their superiority in Natural Language Processing like Speech Recognition[7] and Machine Translation[2]. Recurrent Neural Networks(RNN) based methods like LSTM[4] aim at modeling sequence data using iteratively updated hidden states, with great expressiveness at the cost of computation cost and model parameter amount. Recurrently, attention-based methods like Transformer[10] have shown significant improvement compared with previous methods by employing global attention mechanism. Even so, when it comes to processing short sentences, the superiority of deep neural networks are deteriorated resulting in poor trade-offs between performance and computation efficiency. The structure of models we used are shown in Fig3.

3. Method

3.1. Naive Bayes

In Naive Bayes, we follow the attribute conditional independence assumption, which indicates attributes are independent given the label of the input data. $\mathcal{Y} = \{c_1, c_2, \ldots, c_N\}$ denotes all possible labels, $\mathbf{x}$ denotes the input data. We want to infer the posterior probability over input data $P(c_i \mid \boldsymbol{x})$. Thus we have:

$$P(c \mid \boldsymbol{x}) = \frac{P(c)P(\boldsymbol{x} \mid c)}{P(\boldsymbol{x})} = \frac{P(c)}{P(\boldsymbol{x})} \prod_{i=1}^{d} P(x_i \mid c) \quad (1)$$

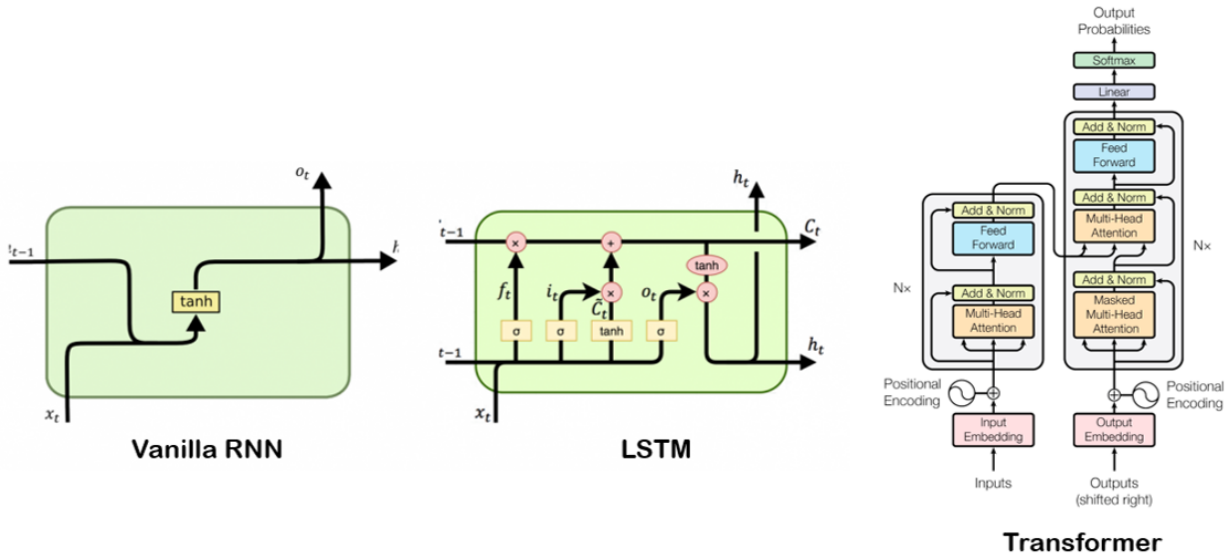

Figure 3. The neural networks models architectures.

where d is the number of feature dimension. x_i is the value of ith dimension of input data.

The training goal is to obtain a mapping $h : \mathcal{X} \mapsto \mathcal{Y}$:

$$h(\boldsymbol{x}) = \arg\max_{c \in \mathcal{Y}} P(c) \prod_{i=1}^{d} P\left(x_i \mid c\right) \tag{2}$$

As for implementation, as shown in Figure 2, during training, we record word frequency given input label. Then during inference, we calculate the word frequency of the input sentence, then form frequency histogram as feature representation. Then, we classify each sentence using its representation with maximum likelihood estimation (MLE).

3.2. Adaptation of Naive Bayes

To improve the generalization ability and effectiveness of Naive Bayes, we implement some adaptations. Original Naive Bayes could be viewed as extracting word frequency histogram as feature representation, and use MLE to classify each feature. As a substitute, classifier could be replaced by KNN, SVM, or MLP. They could deal with complex feature representation more efficiently and effectively in general.

Moreover, in TF-IDF, we use the "term frequency" and "inverse document frequency" to define the importance of a keyword or phrase within the dataset. Term frequency $\mathrm{tf}(t, d)$ of one term t in the dataset d is:

$$\mathrm{tf}(t, d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{3}$$

where $f_{t,d}$ is the raw count of a term in dataset, $\mathrm{tf}(t, d)$ is the normalized count. Inverse document frequency $\mathrm{idf}(t, D)$ of one term t in the dataset d is:

$$\mathrm{idf}(t, d) = \log \frac{s}{|\{s \in d : t \in s\}|} \tag{4}$$

where s is the sentence within dataset, And the TF-IDF is:

$$\text{tf-idf}(t, d) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, d) \tag{5}$$

If simply apply Naive Bayes directly, the size of the word table would be significantly massive, and it would suffer from overfitting. TF-IDF may helps compress word tables and prevent overfitting, form a smaller word histogram, and thresholding infrequent words to be unknown to avoid overfitting.

Original Naive Bayes considers nothing of context information. As a feature engineering trick, we try to form word-groups histogram instead of a single word to model context information.

3.3. Deep Neural Networks

We implemented vanilla RNN, LSTM[4], and Transformer[10] in this task to serve as state-of-the-art methods for comparison.

Vanilla RNN iteratively updates the hidden state for every single data within a sequence. LSTM[4] update it by adding a highway and several gates to maintain long-range dependency. Transformer[10] leverages global self-attention mechanism to form dependency between every two words by calculating softmax score with the product of key(K) and query(Q), and obtain new feature of linear combination between softmax score and value(V):

$$\mathrm{Attention}(Q, K, V) = \mathrm{Softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{6}$$

Model	M-F1	Accuracy avg.
Naive Bayes + MLE	**70.1**	**66.0**
Naive Bayes + KNN	43.1	33.4
Naive Bayes + SVM	66.2	63.8
Naive Bayes + MLP	64.5	59.6
SVM(SOTA)	63.8	-
RNN	54.7	53.2
LSTM	63.2	62.1
Transformer	67.5	**70.2**
RoB-RT(SOTA)[1]	**77.0**	-

Table 1. Tweeter emotion classification experiments on SemEval[8] among different methods we implement and the SOTA method. With data preprocessing, our SVM model and naive Bayes model get a higher score than sota(SVM).

The training and testing pipeline follows Figure 2. For simplicity, we use the third-party library of Keras[3] to implement DNN methods in Twitter emotion classification.

4. Experiments

4.1. Models, Architectures and Dataset

We use the same models and architectures as mentioned in 3, including the two pipelines as shown in Fig2 and some adaptations like TF-IDF, words-group with frequency. To study the performance of different methods, we train multiple models, from basic Naive Bayes to Transformer, with two feature-exacting methods and adaptative tricks used. Due to the massive amount of methods we implemented, it's too tedious to list all parameters and details of our methods, and we have attached them to our code for reference.

To create our training data, we downloaded the Emotion Recognition part of SemEval 2018 (Emotion Recognition)[8] sorted in TweetEval dataset[1], which consists of 3257 train sentences, 374 validation sentences, and 1421 test sentences. Each sentence has 10-20 words and is labeled with one emotion from 4 classes: anger, joy, optimism, sadness.

4.2. Evaluation Metrics

We use the same evaluation metric from the TweetEval[1], which is macro averaged F1 over all classes, in most cases, which can be seen as:

$$\text{macro } F1 \text{ score }_i = 2\frac{\text{precision }_{ma} \times \text{ recall }_{ma}}{\text{precision }_{ma} + \text{ recall }_{ma}} \quad (7)$$

where recall _{ma} and precision _{ma} both are the average value for each class. For better visualization and realizing the dataset, we also offer the accuracy of each class.

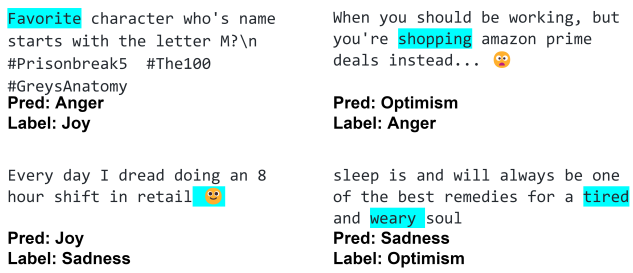

Figure 4. Wrong classification examples, inferred with Naive Bayes + MLE on SemEval[8], highlighted words are the keywords that misguide the model.

4.3. Quantitive Results and Visualization

In most of cases, neural network based methods take averagely better performance than classical methods, but with the cost of computational efficiency. Given the consideration of computation cost, our implementation achieved comparable results with existing method including SOTA method from[1]. With some adaptations like data preprocessing, we get better performance on some models, which is shown in Table 1, and get much less parameters by taking some adaptations like TF-IDF.

Some correct classification examples of SemEval[8] inferred by Naive Bayes + MLE are shown in Figure 1, keywords that largely benefit the decision process have been highlighted. We could notice that sentences containing emotion keywords are significantly likely to be properly classified.

4.4. Error Mode Analysis

Some wrong classification examples on SemEval[8] are provided as well. They are inferred by Naive Bayes + MLE, and are shown in Figure 4. Highlighted keywords are responsible for misleading the model to the wrong prediction. The meanings of selected examples are quite straightforward for people. Even though those highlighted keywords are representing different meaning individually, in the context of the whole sentence, they imply the same emotion as ground truth do.

Those misclassifications are largely due to our assumption of the Naive Bayes model that all features(words) are independent of each other given the label, which neglects the context information that variates words' meaning from individual representation. This implies that constructing the Naive Bayes model would suffer from lacking context modeling and long-range dependency, which would be our possible future direction to solve such problems.

4.5. Ablation Study

To quantify the effectiveness of adaption method like TF-IDF, we conduct ablation studies following the same experiment setting of mentioned above.

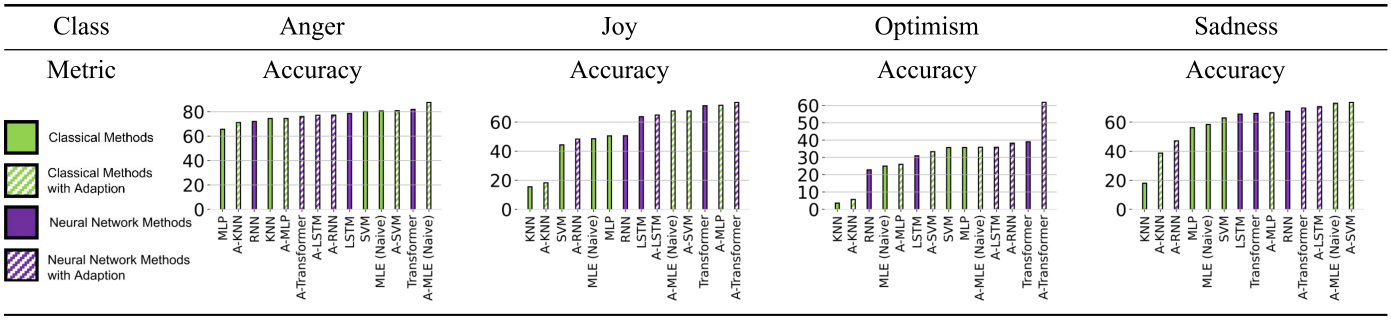

Figure 5. Ablation study results. Models with adaptation get much better results except for the 'anger' emotion. Neural Networks based models mainly get better results than classical models. Every result is evaluated by accuracy in each class.

Model & Adaptation	Dimensions	Accuracy avg.
Origin	12887	53.1
Preprocessing (default)	6635	66.0
TD-IDF w. 3 features	5651	52.1
TD-IDF w. 5 features	6476	54.6
TD-IDF w. 8 features	6627	58.0
TD-IDF w. 10 features	6634	63.4

Table 2. Adaption Study Result. There is a clear dimension reduction after adaptations.

Data Preprocessing: Due to lots of peculiar characters within dataset[1], we prepare some preprocessing techniques like removing those characters from typical words. Specifically, great importance should be attached to emotions, which can not be divided correctly but contributes a lot to emotion recognization. As shown in Fig5, our algorithms get much better performance after data preprocessing. For the 'anger' emotion, we get a lower score, which we think is due to the great overfit before preprocessing.

TF-IDF Adaptation: TF-IDF is a powerful way to reduce the computate consumption. It is a pity that the loss of information from TF-IDF always leads to a lower accuracy, using TF-IDF is still a good trade-off as shown in Table2.

5. Conclusion & Limitations

In summary, we implement the Naive Bayes method with adaptations and DNN methods including vanilla RNN, LSTM, and Transformer in the Twitter emotion classification task and achieved comparable implement results. To demonstrate the effectiveness of our implementation, we have provided throughout ablation studies above.

Even so, those methods we implemented still have some limitations. For example, in the Naive Bayes method, we haven't solved the ambiguity problem yet, which might result in the wrong classification result in the circumstance of the irony of sarcasm. As for the DNN methods, providing better feature engineering could help the model to eliminate ambiguity in light of the small amount of the given training dataset. Both of those are potential future directions that are worth exploring.

6. External Resources

- Keras[3] for DNN implementation.
- Sklearn[9] for popular machine learning algorithm like SVM.

References

[1] Francesco Barbieri, Jose Camacho-Collados, Luis Espinosa-Anke, and Leonardo Neves. TweetEval:Unified Benchmark and Comparative Evaluation for Tweet Classification. In *Proceedings of Findings of EMNLP*, 2020.

[2] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[3] Francois Chollet et al. Keras, 2015.

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[5] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In *European conference on machine learning*, pages 137–142. Springer, 1998.

[6] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.

[7] Yajie Miao, Mohammad Gowayyed, and Florian Metze. Eesen: End-to-end speech recognition using deep rnn models and wfst-based decoding. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, pages 167–174. IEEE, 2015.

[8] Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. Semeval-2018 task 1: Affect in tweets. In *Proceedings of the 12th international workshop on semantic evaluation*, pages 1–17, 2018.

[9] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu

Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[11] Xindong Wu, Vipin Kumar, J Ross Quinlan, Joydeep Ghosh, Qiang Yang, Hiroshi Motoda, Geoffrey J McLachlan, Angus Ng, Bing Liu, S Yu Philip, et al. Top 10 algorithms in data mining. *Knowledge and information systems*, 14(1):1–37, 2008.