Report for EMNLP HW1: Log-Linear Model for Text Classification

Abstract

In this homework, a feature-based Log-Linear model is implemented for the text classification task without using any existing NLP toolkits and the model outperforms the Naive Bayes model. What's more, models with different feature numbers and feature selection methods are compared.

1 Introduction

I implement a feature-based Log-Linear model for text classification task in HW1 of EMNLP the course. There are 4 steps in total to do the text classification task: preprocessing texts, extracting features, training the model and finally testing the model. Details of the 4 steps are in Section 2. A Naive Bayes model is implemented as a baseline.

2 datasets, *SST-5* and *Yelp*[1], are used to evaluate the model. The Stanford Sentiment Treebank(SST) is a corpus with fully labeled parse trees that allows for a complete analysis of the compositional effects of sentiment in language.(Socher et al., 2013) It contains 8,544 sentences in the training set and 2,210 in the test set. *Yelp* is a dataset of reviews from Yelp. It contains 650,000 training samples and 50,000 testing samples, labelled with review stars from 1 to 5.

2 Methods

In this section, I introduce the preprocessing of raw texts, feature extraction and 2 models(Log-Linear and Naive Bayes).

2.1 Preprocessing

First, characters that are neither English letters nor Arabic numerals are removed. All English letters are turned into their lower case. Sentences are split into terms and each sentence is represented as a set. Then stop words are removed from the sets of sentences. The list of stop words used in my implementation is the same as that of NLTK. Then the words are lemmatized using the lemmatization dictionary from https://github.com/michmech/lemmatization-lists. The final result of preprocessing is a series of sets of sentences, which are saved as files.

[1] https://www.yelp.com/dataset

2.2 Feature Extraction

After preprocessing raw texts, the appearance of term in vocab are taken as features. The function of features is shown as below, where x denotes the sentence, y denotes the class and i denotes that it is the ith feature.

$$f_i(x, y) = \begin{cases} 1, & \text{if } vocab_i \text{ in } x \text{ and } class = y \\ 0, & otherwise \end{cases}$$

Given the number of features N, the top-N terms with highest document frequency are selected to form the vocab. In observation, there are some non-standard expressions in *Yelp* such as 'niceeee' and 'wwhhhhaaattt'. These nonstandard expressions can hardly included into the vocab and this will not harm the performance of the model seriously as they only account for a quite small part.

Of course, there are some other methods of feature extraction. (Yang and Pedersen, 1997) introduce 5 ways to filter features, including document frequency thresholding, information gain, mutual information, $\mathcal{X}^2$ statistic and term strength. Information gain is implemented in my homework as well but it is time consuming and can only be used on small datasets. Let $\{c_i\}_{i=1}^m$ denote the classes.

The information gain of term t is defined to be:

$$\begin{aligned} G(t) = &- \sum_{i=1}^{m} P_r(c_i) \log P_r(c_i) \\ &+ P_r(t) \sum_{i=1}^{m} P_r(c_i|t) \log P_r(c_i|t) \\ &+ P_r(\bar{t}) \sum_{i=1}^{m} P_r(c_i|\bar{t}) \log P_r(c_i|\bar{t}) \end{aligned}$$

More important terms is expected to have a bigger information gain. Top-N terms will be selected as features.

2.3 Models

2.3.1 Log-Linear Model

The feature-based Log-Linear model is a linear classifier. Given a text x and a label y, the score function is

$$score(x, y) = \sum_i \lambda_{f_i(x,y)} f_i(x, y),$$

where $\boldsymbol{\lambda}$ denotes the weights of features. Then the classifier choose a y^* to maximize the score.

$$y^* = \arg\max_y \sum_i \lambda_{f_i(x,y)} f_i(x, y)$$

Given a text x, the conditional probability that its class is y is

$$p(y|x) = \frac{\exp \sum_i \lambda_{f_i(x,y)} f_i(x, y)}{\sum_{y'} \exp \sum_i \lambda_{f_i(x,y')} f_i(x, y')}.$$

Given the training data $x_1, x_1, ..., (x_k, y_k)$, the likelihood of $\boldsymbol{\lambda}$ is $L(\boldsymbol{\lambda})$ and the log-likelihood is $LL(\boldsymbol{\lambda})$. I denote the negative log-likelihood as the loss. The loss function is modified to regularize the $\boldsymbol{\lambda}$.

$$\begin{aligned} L(\boldsymbol{\lambda}) &= \prod_k p(y_k|x_k; \boldsymbol{\lambda}) \\ LL(\boldsymbol{\lambda}) &= \sum_k \log p(y_k|x_k; \boldsymbol{\lambda}) \\ &= \sum_k \boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y_k) - \\ &\quad \sum_k \log \sum_{y'} \exp(\boldsymbol{\lambda} \cdot \boldsymbol{f}(x_k, y')) \\ Loss(\boldsymbol{\lambda}) &= -LL(\boldsymbol{\lambda}) \end{aligned}$$

To maximize the log-likelihood, Batch Gradient Descent (BGD) is used.

Dataset	Model	Acc	Macro-F1
SST-5	LL	0.39	0.34
	NB	0.25	0.17
	SOTA	**0.59**	-
Yelp	LL	0.55	0.54
	NB	0.20	0.17
	SOTA	**0.73**	-

Table 1: The results of the 2 models on the 2 dataset and the SOTA results. The evaluation metrics are accuracy and macro-F1.

2.3.2 Naive Bayes Model

The target of Naive Bayes Model is to choose a class s^* that can maximize the probability of text C. The model makes an assumption that terms v_x in C is independent to each other.

$$P(C|s_k) = P(v_x|v_x \in C|s_k) = \prod_{v_x \in C} P(v_x|s_k)$$

$$s^* = \arg\max_{s^*} P(C|s_k)P(s_k)$$

3 Experiment and Analysis

In Log-Linear Model, for both datasets, 20% of the original train data are taken as validation data and the number of features is 2000. For *SST-5*, I set the batch size to 100, learning rate 0.01, and the number of epochs 8. For *Yelp*, I set the batch size to 1000, learning rate 0.001, and the number of epochs 1. The regularization parameter is 0.0001. The learning rate is adjusted in every step with exponential decay.

$$decayed\_lr = lr * rate^{\frac{global\_step}{decay\_steps}}$$

In Naive Bayes model, for *SST-5*, the number of features is 200. For *Yelp*, it is 5000. The results of the 2 models on the 2 dataset are shown in Table 1. The SOTA of *SST-5* is the RoBERTa-large+Self-Explaining model proposed by (Sun et al., 2020). The SOTA of *Yelp* is the HAHNN(CNN) model proposed by (Abreu et al., 2019). Deep learning models are much more powerful than both Log-Linear model and Naive Bayes model. And Log-Linear model performs better than Naive Bayes model. That's because the terms in the vocab are not equally important. Models perform better on *Yelp* because it contains much more data and the average length of texts is longer.

In implementation, I turn features into NumPy arrays and use Numpy operations instead of `for`

N_features	Acc	Macro-F1
500	0.35	0.30
2000	0.39	0.34
5000	**0.40**	**0.35**

Table 2: The results of the Log-Linear model on SST-5 with different number of features.

Model	Method	Acc	Macro-F1
LL	DF	**0.39**	**0.34**
	IG	0.38	0.34
NB	DF	**0.25**	**0.17**
	IG	0.23	0.16

Table 3: The results of the Log-Linear model on SST-5 with different methods for feature selection.

loops as far as possible to improve efficiency. Given the parameters as above, preprocessing raw texts cost 0.64s and 623.2s for *SST-5* and *Yelp* respectively. Feature extraction cost 0.02s and 9.56s. When the number of features is 2000, training of the model costs 6.29 seconds for *SST-5*, and 623.17 seconds for *Yelp*.

I keep other arguments and set the number of features to 500, 2000 and 5000 for the smaller dataset to evaluate its influence on the performance of the Log-Linear Model. The results are shown in Table 2. It can be inferred from the results that a small feature number can harm the performance while a large number can hardly improve the performance in my Log-Linear model.

Besides, I try using information gain as the feature extraction method on the smaller dataset while keeping other arguments the same. The results are shown in Table 3. Surprisingly, usage of information gain decrease the accuracy and macro-F1 score obviously for both models. This may be attribute to the fact that most words are sparse in the data.

4 Conclusion

In HW1, I implement a feature-based Log-Linear model and a Naive Bayes model, detect the influence of feature numbers on the Log-Linear model and compare 2 feature selection methods. It can be inferred from the scores that the number of features is not a decisive factor when exceeding a threshold, and selection by information gain is not suitable for features defined in Section 2.2.

References

Jader Abreu, Luis Fred, David Macêdo, and Cleber Zanchettin. 2019. Hierarchical attentional hybrid neural networks for document classification. *arXiv preprint arXiv:1901.06610*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1631–1642.

Zijun Sun, Chun Fan, Qinghong Han, Xiaofei Sun, Yuxian Meng, Fei Wu, and Jiwei Li. 2020. Self-explaining structures improve nlp models. *arXiv preprint arXiv:2012.01786*.

Yiming Yang and Jan O Pedersen. 1997. A comparative study on feature selection in text categorization. In *ICML*.