Rating Prediction of Clothes Based On Review Text And Size Assessment

Member 1 - A53267693
Member 2
Member 3

Abstract

This report illustrates the difference in performance between a simple lexer of the ac language written in C and another created with Lex, based on their execution time with the same stress example ac program.

1. Dataset

In this project, we used Clothing Fit Data from ModCloth and RentTheRunway. Some basic statistics and properties are as followings:

need to add

In order to make full use of the dataset, we will try two extract different features from the the raw data. And these features can be mainly divided into two parts. One is the assessment of whether the size of the cloths bought by the guests fit them or not and the other is to do some data mining on the review texts provided by the guests to analyze their sentiment to their products. Some detailed information about how we model the prediction task and the methods we use to implement the task will be provided in Section 2 and section 3. We will now describe some interesting findings we found from the raw data, which motivate the design of our model in the following sections.

1.1. Features of Size

1.2. Review Text

In supervised text classification, I tried different stratagies to improve the accuracy. One part of the improvement is achieved by using proper feature extraction methods and the other part of the improvement is achieved by grid searching to find the regularization parameter that leads to the highest accuracy on the development set.

1.3. The Description of The Data And Data Clean-up

There are 192462 samples with ratings provided in the dataset and each of them is made of a piece of review and a corresponding rating, labeling the review with a extent of how they like their clothes. The original reviews are quite noisy. As some samples from the training set show:

Sample 1. "I LOVED the look, BUT: The only size offered was the two, which fit me everywhere but the waist. I pushed through and wore it anyway, and it made eating and drinking impossible and bruised my ribs. That said, the four would have been too large in the bust and shoulders. I ended up having my friend discreetly unzip me a bit and tuck in the sides, which gave me a bit of a plunging back, but allowed me to breathe!"

Sample 2. "PROS:

Beautiful

Lace is gorgeous

Back cut out is fun/flirty without looking unflattering

Perfect for many occasions

CONS:

WAYYY too small. I wore 3 sizes bigger than normal

The dress that fit my stomach was way too big in my shoulders, but I made it work"

There are abbreviations, punctuations, numbers and even some modal particles in the raw data. The words are also case-sensitive. All these factors have a negative impact on the system. Therefore, the data should be cleaned up before feature extraction. To be detail, I remove the digits and punctuations, and make every word lowercase.

2. Predictive Task

3. Model

3.1. Feature Extraction

To improve the accuracy of the supervised text classification, I tried many feature extraction methods. The detailed results of the CountVectorizer method and TfidfVectorizer are shown in Tab 1 and Tab 2. The corresponding analysis are as the following:

Table 1. CountVectorize with different feature extraction methods

Regularization Coefficient / C	Basic	Without stop words	Words (more than 2 characters)	Strip accents	Trigram
C=1	0.77729	0.75764	0.76855	0.77729	0.77947
Best C	0.78165	0.76200	0.77292	0.77947	0.78820
Vocabulary Size	9882	9602	9381	9863	167716

1) **Remove stop words:** Since a basic intuition is that the stop words like "the", "or", "a", etc. are meaningless when classifying a review into positive and negative, it makes much sense to remove these features. And this will decrease the vobabulary size to some extend. However, the result shows that I am too naive. When the stop words are removed, the accuracy decreases, contrary to my expectation. The reason might be stop words play an important role in expressing one's idea. That is to say, some meanful words might be considered as stop words and removed. For example, "down" is a stop word, but in some reviews it can be used as "the restaurant let me down" to express negative motion.

2) **Remove words with no more than 3 characters:** In order to move some modal particles, I only count the words with more than 2 characters as the feature. However, the result proves that the method has poor performance. This is because some modal particles are useful when classifying a review. Besides, the words removed may include some non-modal particles because all the words with no more than 2 characters are removed.

3) **Strip accents:** To deal with the languages besides English, I try to strip the accents. And this also lead to worse peroformance. This is because when the accents are stripped, some languages may be mixed and become unrecognizable, which will decrease the accuracy.

4) **Use Trigram:** Using trigram is to consider the sequence of three word as well as two wordas as features, which means (w_{i-2}, w_{i-1}, w_i) and (w_{i-1}, w_i) are both considered as features. Therefore, as can be seen in Table 1, the feature size is much larger than other mothods. Since the words are more related with the words before and after, the features contain more information about the words' meaning. Therefore, it makes Trigram to achieve better performance.

Table 2. TfidfVectorizer with different feature extraction methods

Regularization Coefficient / C	Basic	sublinear tf	Wo
C=1	0.76637	0.76637	
Best C	0.79475	0.80131	
Vocabulary Size	9882	9882	

5) **Use basic TfidfVectorizer:** TF-idf stands for Term frequency-inverse document frequency. Since the importance of a word increases proportionally to the number of times a word appears in the review but is offset by the frequency of the word in the corpus, the features' importance should be reweighted. The expression of a feature's IDF value is expressed as:

$$IDF(w) = log\frac{N}{N(w)} \tag{1}$$

where N is the number of reviews in the corpus and $N(w)$ is the number of the reviews that contain the word w. When added smoothing method, the expression can be:

$$IDF(w) = log\frac{N+1}{N(w)+1} + 1 \tag{2}$$

$$TFIDF(x) = TF(w) \times IDF(w) \tag{3}$$

where $TF(w)$ is the frequency of word w in a review. Since TFIDF method can decrease the importance of the meaningless words like "to, is, are", the feature will focus more on meaningful words like "nice, tasty, bad, etc." As we can see in Table 2, TFID can improve the accuracy greatly.

6) **TFIDF with sublinear TF:** The difference of sublinear TF with basic TF is that it transfers $TF(w)$ to $log[TF(w)] + 1$. The result shows that sublinear TF has better performance in this project.

3.2. The Hyper-parameters of the Classifer

Description of Logistic Regression: The expression of logistic regression is actually a sigmoid function.

$$g(x) = \frac{1}{1+e^{-wx}}$$

The training of a logistic regression is to find the best weights w that make the loss function minimized.

$$L(w) = \Sigma_{i=1}^{n} - y_i log[1 + e^{-y^{(i)}(wx^{(i)})}]$$

where n is the number of samples in the training set and y_i is the label of $i_t h$ data. When using a trained logistic regression classifer to classify, it computes the value of wx. When $wx > 0$, $g(x) > 0.5$, which means the data belongs to class 1 and when $wx < 0$, $g(x) < 0.5$, classifying into class 0.

Grid Search to Find The best Regularization Parameter:

When using the logistic regression, some parameters are set in Table 3:

Table 3. Parameters of the logistic regression classifier

Patameter	Penalty	Dual Form	fit intercept	class weight	solver
Value	'l2'	False	True	None	lbfgs

To find the best value of the regularization parameter C, I grid seach from 0.1 to 50 with the step 0.1. The results of CountVectorizer and TFIDFVectorizer with the variation of C are shown in Fig 1 and Fig 2.

Conclusion: Considering all the feature extraction methods and hyper-parameter search, the best performance the system can achieve is implemented with the following:

Using TfidfVectorizer with sublinear TF and set the regularization parameter $C = 4.8$, the accuracy of the system can be 80.131.

4. Literature

5. Results

The resulting difference may not seem significant but it is a difference nonetheless. Furthermore, making the lexer with Lex proved to be less time consuming than programming it in C. With all this considered we can safely conclude that the lexer produced with Lex has a better performance than the one programmed in C, which may be because of the use of regular expressions and finite automata.

References

[1] Decomposing fit semantics for product size recommendation in metric spaces Rishabh Misra, Mengting Wan, Julian McAuley RecSys, 2018