The Report of HW2:

Feature engineering and word2vec based sentiment analysis

Cheng Zhong 16307110259

1. Principle

Feature engineering is a method of judging the sentiment of the whole sentence by using the emotion of the words in the sentence as a priori information. First, we calculate the conditional probability of each word in different emotions in the training set. Then we can aggregate the probability of each words in sentence to judge its sentiment.

Word2vec is a method of encoding words using vectors. Based on the hypothesis that "words appearing in the same sentence are more likely to be similar", we can optimize different word vectors by gradient descent method, so that words with similar meanings can get closer in vector.

Negative sampling is a strategy to improve the speed for skip-gram training. For each pair of words, we randomly select a word from the dictionary to take place of the target word in it, then set the label of correct word pair to 1, and set the error one to 0 for optimization.

2. Data Processing:

Corpus: Stanford Sentiment Treebank

Model: word2vec (SoftMax/negative sampling), Bag of words.

Environment: Python 3.7, Windows 10

3. Formula derivation

3.1 SoftMax

Since we made the word prediction with the SoftMax function:

$$J_{softmax-CE}(o, v_c, U) = CrossEntropy(\mathrm{y}, \hat{y}) = -log\ \hat{y}, \qquad \hat{y} = p(o|c) = \frac{\exp(u_o^{\mathrm{T}} v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

and y is the expected word one-hot vector, so we can calculate the gradient for different vector.

(1) The gradients with respect to Vc

$$\frac{\partial}{\partial v_c} - \log\big(p(o|c)\big) = -\frac{\partial}{\partial v_c} \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)}$$

$$= -\frac{\partial}{\partial v_c} \log \exp(u_o^T v_c) + \frac{\partial}{\partial v_c} \log \sum_{w=1}^{V} \exp(u_w^T v_c)$$

$$= -u_0 + \frac{1}{\sum_{w=1}^{V} \exp(u_w^T v_c)} \sum_{w=1}^{V} \exp(u_w^T v_c)\, u_w^T$$

$$= \sum_{w=1}^{V} \hat{y}_w u_w - u_0$$

(2) The gradients with respect to Uw

$$-\frac{\partial}{\partial \mathrm{u_w}} \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)} = -\frac{\partial}{\partial \mathrm{u_w}} \log \exp(\mathrm{u_o^T v_c}) + \frac{\partial}{\partial \mathrm{u_w}} \log \sum_{\mathrm{w=1}}^{\mathrm{V}} \exp(\mathrm{u_w^T v_c})$$

$$= \frac{\exp(u_w^T v_c) v_c}{\sum_{i=1}^{V} \exp(u_i^T v_c)}$$

$$= \hat{y}_w v_c \ (u \neq 0)$$

As for U0,

$$-\frac{\partial}{\partial \mathrm{u_0}} \log \frac{\exp(u_o^T v_c)}{\sum_{w=1}^{V} \exp(u_w^T v_c)} = -\frac{\partial}{\partial \mathrm{u_0}} \log \exp(\mathrm{u_o^T v_c}) + \frac{\partial}{\partial \mathrm{u_0}} \log \sum_{\mathrm{w=1}}^{\mathrm{V}} \exp(\mathrm{u_w^T v_c})$$

$$= (\hat{y}_0 - 1) v_c$$

3.2 Negative sampling

Since $\mathrm{J_{neg-sample}(o, v_c, U) = -\log(\sigma(u_o^T v_c)) - \sum_{k=1}^{K} \log\left(\sigma(-u_k^T v_c)\right)}$,

$$\frac{\mathrm{d}}{\mathrm{dx}} \sigma(\mathrm{x}) = \sigma(\mathrm{x})(1 - \sigma(\mathrm{x}))$$

we can calculate the gradient through the same method as.

(1) The gradients with respect to Vc

$$\frac{\partial}{\partial v_c}\left(-\log(\sigma(\mathrm{u_o^T v_c})) - \sum_{\mathrm{k=1}}^{\mathrm{K}} \log\left(\sigma(-\mathrm{u_k^T v_c})\right)\right) = (\sigma(\mathrm{u_o^T v_c}) - 1) u_0 + \sum_{k=1}^{K} (1 - \sigma(\mathrm{u_o^T v_c})) u_k$$

(2) The gradients with respect to Uw

$$\frac{\partial}{\partial \mathrm{u_k}}\left[-\log(\sigma(\mathrm{u_o^T v_c})) - \sum_{\mathrm{K=1}}^{\mathrm{K}} \log\left(\sigma(-\mathrm{u_k^T v_c})\right)\right] = -\frac{\partial}{\partial \mathrm{u_k}} \log\left(\sigma(\mathrm{u_k^T v_c})\right) - \frac{\partial}{\partial \mathrm{u_k}} \sum_{\mathrm{K=1}}^{\mathrm{K}} \log\left(\sigma(-\mathrm{u_k^T v_c})\right)$$

$$= \left(1 - \sigma(-\mathrm{u_k^T v_c})\right) \mathrm{v_c} (k \neq 0)$$

As for U0,

$$\frac{\partial}{\partial u_0}\left[-\log(\sigma(u_o^T v_c)) - \sum_{K=1}^{K} \log(\sigma(-u_k^T v_c))\right] = (\sigma(u_o^T v_c) - 1) v_c$$

From the gradients we calculate above, we can optimize the vector to encoding words.

3.3 Bag of words

In training set, we can compute the priori probability for each words and sentence in different sentiment by the Bayes' theorem and add-1 smoothing.

$$\hat{\mathrm{p}}(\mathrm{w|c}) = \frac{\mathrm{count(w,c)} + 1}{\sum_{\mathrm{w \in V}} \mathrm{count(w,c)} + |\mathrm{V}|}, \hat{P}(c) = \frac{N_c}{N}$$

In test set, we can compute the possibility of sentiments for each sentence and accept the sentiment with the highest probability.

$$\hat{p}(sentiment|sentence) = \hat{P}(sentiment) \prod_{words} p(word|sentiment)$$

4. Algorithm

4.1 Word2vec

In this task, we only need to fill in the function of the program, so we only need to update the gradient according to what we derive above.

4.2 Sentiment analysis

We need to find the "optimal" regularization parameter in this task. So, I use the *NumPy. logspace* function to generate different regularization parameter, then iterate and draw in different parameter.

4.3 Bag of words

The whole program is divided into two sections. The training section focuses on the calculation of the prior probability and the test section focuses on the judgment of the sentence.

In the training section, we traverse all the sentences and words and counting it according to different emotional tags, then compute the priori probability using the formula above.

In the test section, we multiply the prior probabilities of the words in the sentence and multiply the prior probabilities of each sentiment. Then we can take the sentiment with the highest probability and compare with the truth.

5. Experimental result

5.1 Word2Vec

After 4000 iteration, we finish the training of word vector. We perform SVD decomposition on the word vector which needs to be visualized and reconstruct the vector by taking the largest two singular values. Then plot the projections of these word vectors on the two-dimensional plane. The figure is as follows.

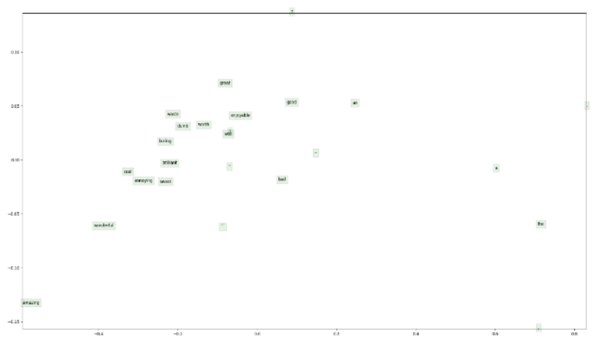

Figure 1 Word_vector.png

As can be seen from the picture, words and punctuation are separately clustered. Words are also divided into several categories according to sentiment. Negative words such as boring, dumb, and waste are closer, and positive words such as enjoyable, well, great, and good are closer.

5.2 Sentiment analysis

By testing different parameters, we can plot the line of the regularization parameter against the accuracy.

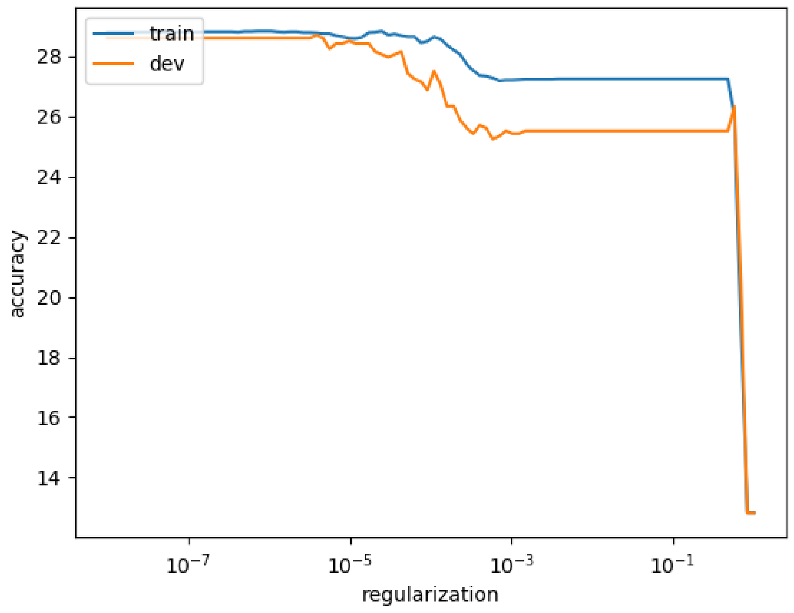

Figure 2 word2vec_acc.png

As can be seen from the figure, the smaller the regularization parameter is, the higher the correct rate is. When the regularization parameter is greater than 1, the correct rate rapidly drops to about 14%. Otherwise the accuracy rate can reach about 25%. There is no significant difference in accuracy between verification set and test set

5.3 Bag of words

By using the training data and test data in the Stanford Sentiment Treebank dataset, the correct rate can reach about 40%, which is better than word2vec method. However, when I removed the stop words in the dictionary, the correct rate dropped to 38%. Maybe the stop word also contains some information of sentiment.

6. Summary

From several experiments, we can find that the bag of words model works best on sentiment analyzing, and the word2vec model can show the relationship between word semantics.

At the same time, the deprecated of stop words does not have a positive effect on sentiment analysis during the experiment.