Sentiment analysis based on feature engineering and word2vec

Yixiong Chen 16307110231

2019.11.16

Introduction

In this article, we will talk about a sentiment analysis task. Our dataset consists of about 10 thousand sentences, each sentence has a label from 0-4 (representing "very negative", "negative", "neutral", "positive", "very positive"), our goal is to establish models to predict what attitude does a piece of text hold. This article will cover two methods for this task. The first one is based on feature engineering, using naive Bayes model to predict which class a text belongs to. The second one is word2vec based, using skipgram model to predict the distribution of words near a central word, and analysis the text class.

Methods

1.Naive Bayes model

The naive Bayes method is quite simple, please see the Bayes theorem below:

$$P(c|d) = \frac{P(c,d)}{P(d)}$$

The probability of a document belonging to a class can be calculated by the joint probability and the prior probability of a document. If we

want to find which class a document should belong to, we can maximize this probability over all classes.

$$max_c\, P(c|d) = \frac{P(c,d)}{P(d)}$$

Because the denominator is not relevant to class, so this optimization can be simplified as

$$max_c\, P(c,d) = P(c)P(d|c) = P(c)P(x_1, x_2, ..., x_n|c)$$

Here we still have difficulty obtaining $P(x_1, x_2, ..., x_n|c)$, so we can make use of the assumption of "Bag of Words". We regard the positon of each word meaningless and the conditional probabilities of them are independent. Therefore,

$$P(x_1, x_2, ..., x_n|c) = P(x_1|c)P(x_2|c)...P(x_n|c)$$

So finally the goal of this model is to find

$$argmax_c\, P(c)P(x_1|c)P(x_2|c)...P(x_n|c)$$

When meeting unseen words, the model uses Lapace smoothing to avoid 0 probability.

2.Word2Vec model

The Word2Vec method (Skip Gram) aims to predict the distribution of surrounding m words given a central word. The cross entropy (negative log likelihood) cost function we want to minimize can be written as follows:

$$J(u,v) = -\frac{1}{T}\sum_{t=1}^{T}\sum_{-m\le j\le m, j\neq 0} log\frac{exp(u_{t+j}^T v_t)}{\sum_{w=1}^{V} exp(u_w^T v_t)}$$

There are two ways to solve the optimization of J. The first one is hierarchical softmax, we derive the gradient of J w.r.t u and v respectively and use SGD to minimize J directly, but the dimension would be too high and is computational consuming. The second one is called negative sampling, in this method we choose several negative samples, and only calculate the gradient w.r.t the target word and negative samples. Usually, the negative

sampling method can be much faster and more efficient. After finding word vectors, we use the mean of word vectors of a sentence as the feature and use softmax to do sentiment analysis (classification) task.

Results

After experiments, the two implemented methods gave us several results:

1.Naive Bayes model

The naive Bayes model performs stable and quick. There are no hyper parameter for this model to choose, so the result is from the simplest plain implementation.

In this sentiment analysis task, naive Bayes model gave us the accuracy of 40.633% on test set. The total training time was 6.6 seconds.

2.Word2Vec model

Word2Vec model relies on SGD and regularization. So learning rate and regularization are picked for our hyper-parameters. First let learning rate be the default value 3.0, and do experiments to find the best regularization parameter. Because the regularization performance has little to do with the learning rate, we can fix it for the second experiment. After this two experiments, the best regularization parameter was 10^{-5} and the best learning rate was 3.0. (For each experiment, please refer to graphs "reg_acc.png" and "lr_acc.png").

In this sentiment analysis task, Word2Vec model gave us the accuracy of 27.240% on the test set. The training of word vectors has token about 3 hours.

In addition, the Word2Vec model is mainly used to generate word vectors. An experiment for visualizing word vectors shows how they look like by reducing the dimension of them to 2 with PCA. The graph is in the file folder called "word_vectors.png". In this graph, we can easily see that not the words have the similar meaning are placed together, but the words with the same form and opposite meaning are placed near to each other. For example, the words "amazing" and "annoying", the words "good" and "bad",

and the words "dumb" and "wonderful" etc. This maybe the result of how the texts look like, there is a high probability that these similar but opposite words share the same neighbor words in the document, which makes the words embedding takes place at the same position.

Analysis

The sentiment analysis task needs the models to classify texts into 5 types, indicating the attitude feature of these documents. Although not exactly, but this task can be seen approximately as text classification. After understanding the problem in this form, it's not surprising that naive Bayes model works better than Word2Vec model.

As we know, naive Bayes model works especially for classification. Bayes theorem acts as its theory support. Although simple, but this model can learn how each class has its word distribution. Given a new document, it can tell which class it belongs to by matching the word distribution of this document and every class. In this sentiment classification task, word distribution cannot be distinguished well enough because of the limitation of length of training sentences and wording similarity of each attitude. However, it still gives us a stable and efficient result.

The Word2Vec model usually works for word embedding. It aims for representing every word with a vector. After training, the weight matrix of neural network is the word vectors we need. Though it is a good method for solving word embedding problem, it doesn't work well in this sentiment analysis task as we expected. Just training Skip Gram model and getting word vectors cannot provide enough information of how a sentence is generated and what emotion it wants to show. This model can only give a classification accuracy a bit higher than plain guesses (5 classes, each class has 20% prediction accuracy) after a long training time compared with naive Bayes model.

All in all, in sentiment analysis task Word2Vec model doesn't work well, but naive Bayes model based on BagofWords performs reletively accaptable.