Sentiment analysis of movie reviews using k-skip-n-grams

Anton Zuev [1]

***Abstract*— The paper describes how k-skip-n-grams features could be extracted and used to classify movie reviews from keras.imdb dataset. Several experiments were conducted to find n and k to extract features. Also it was shown how these features works in Probability-based model.**

I. Introduction

Sentiment analysis of movie reviews is a process during which we want to know whether this particular review is positive or negative. It could be done using probabilistic models or using neural networks (sometimes even with context representation, e.g. LSTM).

However, sometimes is much more important to extract features from the text (or movie review in our case) than to build a model. There are cases when a small preprocessing of the input data yields to a significant change of the accuracy.

The motivation behind this paper is the following. For human it is enough to know only 5-10 words from the movie review to know whether this review is positive or negative. However, the difficult task is to find such words.

In this work I am focused on extracting these 5-10 words to classify movie reviews. Then the simple probability-based model was build to check this hypothesis.

II. Concepts

A. Features extraction

First of all, the features extractor is implemented. I wouldn't consider it in the details. It receives n_value, k_value and a review and extract k-skip-n-gram from this review.

B. Probability-based model

This model was implemented during the work and is used to check whether k_value and n_value were taken correctly.

It works very simple. During the initialization of the model k_value and n_value have to be passed.

When "fit" method is called, the model divides the reviews that were passed into two groups: one for positive, another one for negative. Then both for positive and for negative reviews k-skip-n-grams are extracted and creating the distribution of the k-skip-n-grams per each type of review.

When "predict" method is called, the model for each review computes probability to be generated by both positive and negative k-skip-n-grams distributions. Then the review marked as positive if the probability to be generated by positive k-skip-n-grams distribution is greater that to be generated by negative k-skip-n-grams distribution. Otherwise the review is marked as negative.

[1]Anton Zuev is a 4th year student in Innopolis University, Russia a.zuev@innopolis.ru

When "predict_proba" method is called, exactly the same things happened as while calling "predict" method. The only difference is that it returns probability of belonging to each class rather then the final label.

C. BiProbability-based model

This is some kind of extension of the Probability-based model. The idea is that this model encapsulates two different Probability-based model.

When "fit" method is called, this model just calls corresponding methods of each model.

When "predict" method is called, the model calls "predict_proba" method of each model, then sum up probabilities per each review per each type. Finally, it compares probabilities and returns label that corresponds to the greater probability.

The logic behind "predict_proba" method is the same as for Probability-based model.

III. Experiments and results

In this section all experiments are going to be described. We will start from Probability-based model and will try to optimize the model by changing values of n and k. Then we will try to combine this knowledges and create BiProbability-based model. During the experiments we also will try to use different amount of top words and different length of the review.

A. Probability-based model: n = 3, k = 1

This subsection describes a baseline. Here we use n = 3 and k = 1, let number of top words be 5000 and length of review is 150 words (everything was chosen randomly).

The result that we got is the following:

TABLE I
THE RESULT OF THE BASELINE

Train	Test
0.97596	0.75768

The result for the training data is quite good, for the test data we want to have something more.

B. Probability-based model: k- and n-values optimization

In this subsection the optimization process is described, the results for different n- and k-values are provided as well.

The optimization method is very simple. We have a range of values for n (1, 2, 3) and for k (0, 1, 2, 3, 4, 5, 6). Method just creates all possible pairs out of n- and k-ranges. Then for

every such pair the model is created, fitted and then evaluated both on training and on test data.

The result of this process is shown below:

TABLE II
THE RESULT OF THE OPTIMIZATION

n	k	Train	Test
1	0	0.83068	0.80288
1	1	0.83068	0.80288
1	2	0.83068	0.80288
1	3	0.83068	0.80288
1	4	0.83068	0.80288
1	5	0.83068	0.80288
1	6	0.83068	0.80288
2	0	0.96056	0.85624
2	1	0.96544	0.83216
2	2	0.96512	0.80872
2	3	0.9636	0.79052
2	4	0.96192	0.78256
2	5	0.96264	0.77808
2	6	0.95984	0.77212
3	0	0.98232	0.8458
3	1	0.97596	0.75768
3	2	0.96884	0.69908
3	3	0.96792	0.67056
3	4	0.96548	0.66496
3	5	0.96468	0.65872
3	6	0.96396	0.65744

We can see from the Table II that the best result for training data is 0.98232 (n = 3, k = 0) and it corresponds to 0.8458 for the test data.

However, the best result for the test data is 0.85624 (n = 2, k = 0). Also there is one more interesting case (n = 2, k = 1), accuracy for training data is 0.96544 and 0.83216 for the test. This result is close to (n = 3, k = 0) but here we use 1-skip-2-gram. May be it is a good idea to combine benefits from two models : (n = 3, k = 0) and (n = 2, k = 1)?

C. BiProbability-based model

In this subsection two cases of using BiProbability-based model are described.

- (n = 3, k = 0) and (n = 2, k = 1)
- (n = 2, k = 0) and (n = 2, k = 1)

The results are shown in Table III.

TABLE III
THE RESULT OF THE BIPROBABILITY-BASED MODEL

n1	k1	n2	k2	Train	Test
3	0	2	1	0.97752	0.80748
2	0	2	1	0.97472	0.82856

We see that the accuracy of the model becomes less than it was in Probability-based model.

D. BiProbability-based model: removing most popular words from the input data

The motivation behind this experiment is to try to remove noise from the data. There are some words that are used very often (e.g. prepositions, etc) but don't have any meaning in terms of sentiment analysis. For this reason two experiments were conducted:

- Remove 100 most popular words from the training and test data
- Remove 10 most popular words from the training and test data

However, it's a bit dangerous to do because word "not" is quite popular but, on the other hand, it changes completely the meaning of the phrase (e.g. "This film is good" and "This film is not good"). Anyway, let's see results in Table IV.

TABLE IV
THE RESULT OF THE BIPROBABILITY-BASED MODEL WITH MOST POPULAR WORD FILTERING

Words_removed	n1	k1	n2	k2	Train	Test
100	3	0	2	1	0.99372	0.75916
10	3	0	2	1	0.99388	0.85984
100	2	0	2	1	0.97472	0.82856
10	2	0	2	1	0.98656	0.85804

The best result for the training data is 0.99388 and for the test 0.85984 (for model n1 = 3, k1 = 0 and Words_removed = 10)

IV. EVALUATION AND DISCUSSION

A. Background

In this section I want to discuss steps that I performed during the work and analyze what exact meaning they have.

In the very I wanted to see the task from the different points of view.

First of all, this task could be considered as a classification task, where we have only two different classes. Then we need to extract some features and based on them try to recover original dependency.

Another approach is to try to cluster all reviews into two clusters. Let's imagine that we don't know what label each review has. Then let's try to group them based on theirs similarity. Before talking about similarity we need to map all reviews to a vector space in order to measure distance between them. Then try to understand how well our clusters correspond to positive and negative clusters. The ideal thing is to find such way to transform review into vector that one cluster will consist of only positive reviews and another only of negative ones. The task from this point of view is to find an algorithm according to which review-to-vector space mapping could be done.

The third approach is to use probability model. The idea is the following. Let P and N be a functions that generates positive and negative review correspondingly. The important thing that the result of these functions is random. Since we have dataset with reviews, we can divide training data into two groups and using these two groups define P and N functions. The perfect case would be when P-function

generates positive review with probability equals to 1 and negative review with probability equals to 0 (and vice versa). It looks very similar to classification task. However, it is almost impossible to get such function in real word. Let's modify a bit the conditions for the functions: P-function generates positive review with greater probability that negative one (and vice versa). The logic behind it is very simple: it is more likely to generate positive review by P-function than by N-function (and vice versa).

B. Removing most popular words

This step should be discussed a bit. When we remove most popular words, on the one hand, we remove something that is very likely to occur both in positive and negative reviews. However, there are several significant question to discuss:

- what if in one group (e.g. positive reviews) the number of such words greater than in another? Therefor, some information is lost.
- how will it affect the k-skip-n-grams?

The answer to the first question is known and we just have to keep it in mind while we perform such transformation of the input data.

The second question is much more interesting. The idea behind k-skip-n-grams is to try to capture some relation between words if there are some words between them (e.g. in "I was excited by this great film" we can capture "I excited" omitting word "was" without meaning in terms of sentiment analysis). When we remove popular words from the review, according to common sense, k-skip-n-grams have no benefits opposite to n-grams. If so, then how it's possible to explain that accuracy becomes better when we use them? May be there are some important relations between words that are useful and quite close to each other but there is one or two words between. If so, it makes the model less interpretable. However, using this trick the best result was achieved (0.85984).

V. Conclusion

During this work the possibility to get a good accuracy by using k-skip-n-grams was shown. In contrast to the neural networks, it is more interpretable and efficient (faster fitting).

There is a drawback: huge number of possible k-skip-n-grams could be generated (2 413 744 in my case) therefor it is required to have a lot of main memory to store it.

However, if more k-skip-n-grams would be introduced to the model than it will become less interpretable and, I think, the probability of overfitting would increase.

References

[1] Source code, https://github.com/AnZuev/k-skip-n-gram-sentiment-analysis-movie-reviews