Analysis of the number of words in sentiment classification

BumJin Park
department of Mathematics
Chung-Ang University

ABSTRACT

Every day people write something with a mobile phone and laptop and most of the data people produce is not a number, but a language. Recent studies have been conducted on the text and there are many resources such as a blog, twitter, and even reviews[1]. It is not easy to create data from the text for machine learning training. A single sentence is composed of many words and you need to embed words for the sentence. However, the size of training data grows with word embedding. This paper discusses how to set the number of words for natural language data.

***K*eywords** Natural Language Processing • Machine learning • Domain-specific-dictionary • Classification

1 Introduction

The most basic problem in natural language processing is sentiment analysis. It is a matter of aligning the label of the sentence to positive or negative. The famous research is Twitter sentiment analysis [2]. The label of a sentence is determined by not only the number of positive words and negative words but also the context of the sentence and the combination of words. You can consider the context by using neural networks in deep learning architecture [3] but to make the problem simple, I focus on the number of words and use machine learning algorithms. There are many machine learning algorithms you can use such as SVM, KNN, Random Forest. I tried many machine learning algorithms whose accuracy rate is more than 70%. In natural language processing, determining the impact of the single word to the sentence is important. A single word may have a different meaning in a different domain. That is why I used a domain-specific dictionary [4].

2 Data

Natural language data is not clean and requires many preprocessing tasks. Kaggle[1] provides good natural language resources that are well organized. I used IMDB movie review data[2] consisting of 50000 records. Each record has a sentiment label of positive or negative. Figure1 shows the distribution of the number of words in a single sentence. As it shows, the number of words in a single sentence is quite many. Most of the sentences have more than 100 words. This data has not only words but also punctuation and symbols. Before training, it is required to clean the data.

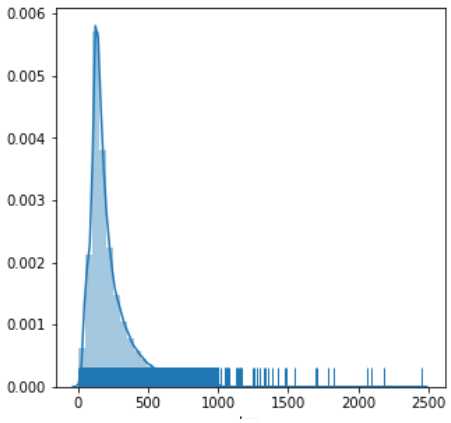


Figure 1 number of words in the sentence

[1] www.kaggle.com

[2] https://www.kaggle.com/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

3 Methods

3.1 Preprocessing

3.1.1 Stop words

There are meaningless words like (a, the). These are filtered out before or after processing of text. However, in our experiment, we select words with bounds. the stop words appear many times and they are filtered out when we bound the word frequency.

3.1.2 Punctuation

Symbols like (, . ? ! / * &) must be filtered out to reduce the number of words.

3.1.3 Uppercase

The beginning of a sentence is uppercase. Two words, 'best' and 'Best' should not be distinguished, but they are. Make every uppercase alphabet to lowercase. The proper noun will be filtered out by the bound of word count.

3.2 Creating data for training

Natural language must be changed to numbers to train. If a word occupies one dimension, there must be a space of size word count for a single word. As a consequence, the data size is (#records * #words). It is too large to train because RAM only has a limited memory size. Even if you can load data, you must consider that the large number of words that you use doesn't guarantee good performance. We made data of size (#records * #small-words) by choosing a lower bound *L* and upper bound *H*. Figure 3 represents an algorithm on how to make data.

	don't	movie	Like	best	great
this movie is best	0	1	0	1	0
I don't like it	1	0	1	0	0
great great movie	0	1	0	0	2

Table 1 Example of making data

Figure 2 Algorithm Making Data

Algorithm Make Data

1. count all the words for sentences like Table1.
2. *W1* be a list of all words.
3. Let *L, H* be some numbers to bound words.
4. Choose words whose total appearance is between *L* and *H*
5. *W2* be a list of bounded words.
6. Count all the words that are also in *W2* for sentences like Table1 again

3.4 Machine learning methods

1) SVM

Support Vector Machine is a supervised learning model. For hyperplane $(\mathrm{w}\cdot\mathrm{x}-\mathrm{b}=0)$, $(\mathrm{w}\cdot\mathrm{x}-\mathrm{b}=1)$ and $(\mathrm{w}\cdot\mathrm{x}-\mathrm{b}=-1)$ are hyperplanes that separate the two classes of data, so that the distance between them is as large as possible.

$$\max\left(\frac{2}{||w||_2^2}\right)\rightarrow\min\frac{1}{2}||w||_2^2 \qquad (1)$$

2) Gaussian Naïve Bayes

If $\mathrm{x}=(\mathrm{x}_1,\ldots x_d)$ is continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a normal distribution. First, we segment the data by the class and then compute the mean and variance of x in each class. The desired probability for each class can be obtained by the production of probabilities for x_d.

$$\mathrm{P}(\mathrm{x}=\mathrm{v}\,|C_k)=\frac{1}{\sqrt{2\pi\sigma_k^2}}e^{-\frac{(v-u_k)^2}{2\sigma_k^2}} \qquad (2)$$

$$\mathrm{P}(\mathrm{C_k}|x)\propto\prod_{d=1}^{D}P(x_d|C_k)P(C_k) \qquad (3)$$

3) Decision Tree

For finite discrete domains and a single target class, a decision tree make a decision by its nodes. Node is trained to maximize the information gain

$$H(T) = I_E(p_1, p_2, \ldots, p_n) = -\sum_{i=1}^{n} p_i \log_2 p_i \quad (4)$$

4) Random Forest

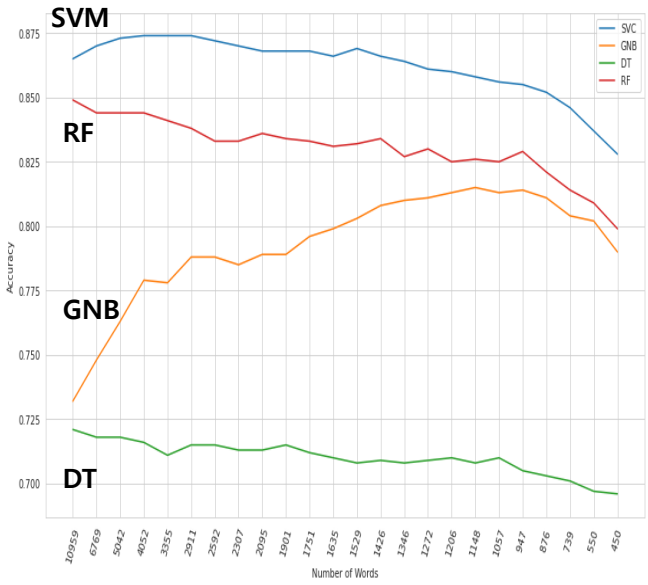
Random Forest operates by constructing a multitude of decision trees at training time. The outputting class is a mode or mean prediction of the individual trees.

4 Hypothesis

Let H be a maximum of word count and L be a minimum of word count. We're going to choose the words whose count is between L and H. Let S be the number of records (50K with the above data). Since we have 2 classes, we may assume that if there is a special word which is a good discriminator of classes such as 'good' or 'bad', then every record of the same class may have it with high probability. There are approximately $S/2$ numbers for a given discriminator word. I set the maximum $H = S$. It is not easy to consider minimum S and I tried many kinds of a lower bound. It is also proportional to S like H. L is Starting from $0.001*S$ to $0.05*S$

5 Results

Overall, the performance of the SVM was the best, and the model predicted correctly in the order of RF, GNB, and DT. It should be noted that, when the number of words is 10K and 1K, the difference in accuracy was not severe in prediction despite 10 times more words (0.87 and 0.85). This means that removing words with low word frequency does not significantly affect performance. Also, in the case of Gaussian Naïve Bayes, the more words, the worse the performance. Consider that the 87.4 percent performance of SVM, which reduced the word count to 4K based on the total word count of 223K, is the best. It is possible to obtain meaningful performance even if you train based on frequently occurring words.



Hypothesis Word bound (S=#records)	Number of words	SVC	GNB	DT	RF
All words	223234	NaN	NaN	NaN	NaN
0.001S ~ S	10959	0.865	0.732	**0.721**	**0.849**
0.002S ~ S	6769	0.870	0.748	0.718	0.844
0.003S ~ S	5042	0.873	0.763	0.718	0.844
0.004S ~ S	4052	**0.874**	0.779	0.716	0.844
0.005S ~ S	3355	0.874	0.778	0.711	0.841
0.006S ~ S	2911	0.874	0.788	0.715	0.838
0.007S ~ S	2592	0.872	0.788	0.715	0.833
0.008S ~ S	2307	0.870	0.785	0.713	0.833
0.009S ~ S	2095	0.868	0.789	0.713	0.836
0.010S ~ S	1901	0.868	0.789	0.715	0.834
0.011S ~ S	1751	0.868	0.796	0.712	0.833
0.012S ~ S	1635	0.866	0.799	0.710	0.831
0.013S ~ S	1529	0.869	0.803	0.708	0.832
0.014S ~ S	1426	0.866	0.808	0.709	0.834
0.015S ~ S	1346	0.864	0.810	0.708	0.827
0.016S ~ S	1272	0.861	0.811	0.709	0.830
0.017S ~ S	1206	0.860	0.813	0.710	0.825
0.018S ~ S	1148	0.858	**0.815**	0.708	0.826
0.020S ~ S	1057	0.856	0.813	0.710	0.825
0.023S ~ S	947	0.855	0.814	0.705	0.829
0.025S ~ S	876	0.852	0.811	0.703	0.821
0.030S ~ S	739	0.846	0.804	0.701	0.814
0.040S ~ S	550	0.837	0.802	0.697	0.809
0.050S ~ S	450	0.828	0.790	0.696	0.799

References

[1] GARG, Surbhi; VERMA, Neetu. Study of Sentiment Classification Techniques. 2018.

[2] KOULOUMPIS, Efthymios; WILSON, Theresa; MOORE, Johanna. Twitter sentiment analysis: The good the bad and the omg!. In: *Fifth International AAAI conference on weblogs and social media*. 2011.

[3] DUNCAN, Brett; ZHANG, Yanqing. Neural networks for sentiment analysis on Twitter. In: *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*. IEEE, 2015. p. 275-278.

[4] PRÖLLOCHS, Nicolas; FEUERRIEGEL, Stefan; NEUMANN, Dirk. Generating Domain-Specific Dictionaries using Bayesian Learning. In: *ECIS*. 2015.