Sentiment analysis and Naïve Bayes Classifier

Yu Fan
Weiqiang Zhu

Abstract

Social media is becoming more and more popular these days. People put up lots of information on the Internet, which can be of great commercial value. Through analyzing the headlines people posted on social media like Twitter or Facebook, we can detect their attitudes towards a certain topic. By detecting the opinions from customers, companies can make improvements to their products timely and policy makers can adjust the policies accordingly. This kind of sentiment analysis is widely used in many of the commercial entities and is still a hot topic to discuss today.

There are many ways to perform sentiment analysis. Naïve Bayes approach is considered to be an efficient and simple way to do it. In this project, we built our Naïve Bayes classifier in R, and trained the model with a movie review data set sourced from rotten tomato. We then validate the model using the second part of that dataset and the result is satisfactory. Next, we tried out our model with the data collected through Tweeter API. We did some analysis on the result and discussed some problems as well as possible approach to improve the Naïve Bayes algorithm.

1. Introduction

Sentiment analysis has a very comprehensive content to discuss and a lot of papers were written on this topic.

1. Unsupervised learning: Text scoring

A simple way to perform a sentiment analysis is to score the words. For example, "good" would get a score of "+2", excellent would get a positive score of "+4", and "terrible", "-4".

AFINN is a list of English words rated for valence with an integer between minus five (negative) and plus five (positive). The words have been manually labeled by Finn Årup Nielsen in 2009-2011.

(http://www2.imm.dtu.dk/pubdb/views/publication_details.php?id=6010)

The idea is to using the rate of words to rate the sentences and rate the whole review at last to decide whether the review is positive or negative one, but experimental result is not that satisfactory and often yield out an accuracy rate of 60% to 70%, if we don't adjust the rates of the words according to the topic.

2. Other approaches

Bo Pang and *Lillian Lee*(2002) compared three method to perform sentiment analysis on movie reviews.

	Features	# of features	frequency or presence?	NB	ME	SVM
(1)	unigrams	16165	freq.	**78.7**	N/A	72.8
(2)	unigrams	"	pres.	81.0	80.4	**82.9**
(3)	unigrams+bigrams	32330	pres.	80.6	80.8	**82.7**
(4)	bigrams	16165	pres.	77.3	**77.4**	77.1
(5)	unigrams+POS	16695	pres.	81.5	80.4	**81.9**
(6)	adjectives	2633	pres.	77.0	**77.7**	75.1
(7)	top 2633 unigrams	2633	pres.	80.3	81.0	**81.4**
(8)	unigrams+position	22430	pres.	81.0	80.1	**81.6**

Source: *Thumbs up? Sentiment Classification using Machine Learning Techniques* (Bo Pang and Lillian Lee Shivakumar Vaithyanathan)

SVM(Support vector machine) seems to work the best, but there is actually no huge accuracy differences between the three methods, Naïve Bayes(NB), Maximum Entropy(NE), and Support Vector Machine.

For a more practical purpose, in this project, we define our goal as to build a classifier that can detect the opinions on a specific topic, positive, negative or neutral. For our experiments, we chose to work with movie reviews. There are several reasons that we choose movie reviews as our experiment data. First, this domain is experimentally convenient because there are large on-line collections of such reviews and the dataset is easy to find. We source our data from___, which contains approximately 1000 labeled data for each category, so that we don't need to hand-label our data. Second, movie reviews is less ambiguous and the structure of the text is simpler than other topics like politics.

2. Naïve Bayes on movie reviews

1. The bag of words representation[1]

Let's take a closer look at the problem, how we can we analyze a movie review like this:

"I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun… It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet."

The first thing we do is to represent the whole text with keywords which express the attitudes towards the movie. The movie review can be probably represented like this:

"x **love** xxxxxxxxxxxxxxxx **sweet** xxxxxxx **satirical** xxxxxxxxxxxxxxxxxxxxx **great** xxxxxxxxxxxxxxxxxxxxxxxxxx **fun** xxxxxxxxxxxxxxxxx **whimsical** xxxx **romantic** xxxx **laughing** xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx **recommend** xxxxx xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx **several** xxxxxxxxxxxxxxxxxxxx **happy** xxxxxxxxx **again** xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx!"

By extracting the keywords of the text and filtering out the other words that do not contain much of the information we need, the problem is simplified: how do we detect people's attitude with these bag of keywords?

2. Assumptions and Bayes inference

When using Naïve Bayes we have two important assumptions in this case which are consistent with a general Bayes approach.

- **Exchangeability**. The order of the keywords that appears in the sentence does not affect the opinion of the sentence.
- **Conditional independence**. The appearance of the words in a sentence is independent.

These two assumptions hold for most of the time, but there are also some situations that do not fit in, and we would discuss later.

Based on these two assumptions, we can do a simple Bayes inference:

$$\boldsymbol{P(X_1, X_2 \ldots X_n|Y) = \prod_{i=1}^{n} P(X_i|Y)}$$

In this case, that means the posterior, the probability a review is positive given the bag of words:

$$P(pos|love, sweet, \dots, happy) \propto P(love|pos) * P(sweet|pos) * \dots * P(happy|pos) * P(pos)$$

3. Training Naïve Bayes:

We train our classifier using the training set to obtain the marginal likelihood and conditional probabilities of the keywords by their frequency.

Prior: P(pos) = 0.35 P(neg)= 0.30 P(neu)=35,

In our training set, 35% reviews are positive, 30% are negative and 35% are neutral.

Likelihood:

The conditional probability is the frequency of a bag of words in certain category.

P(word|Pos)P(word|neu) P(word|Neg)

Love 0.038	Love 0.008	Love 0.007
.	.	.
.	.	.
.	.	.
Fun 0.062	Fun 0.002	Fun 0.032

We can see that positive words are more often appeared in a positive movie review, so that:

P(s|pos)>P(s|neu) >P(s|neg)

4. Laplace smoothing

Another problem here is that if there is a word that never appears in the conditional probability word bag, say the word "gorgeous" has never appeared in our word bag. P(gorgeous|pos)= 0 and our posterior P(pos|gorgeous, good,…)=0, and our classifier fails. In order to solve this problem we have to remove those "0", and a simple way to do the smoothing is:

$$P(word|pos) = \frac{(\#\ pos\ training\ reviews\ containing\ word) + 1}{(\#pos\ training\ reviews) + (\#words\ in\ vocabulary)}$$

We simply add one to the numerator to get rid of the 0s, this is also known as the "add one" smoothing.

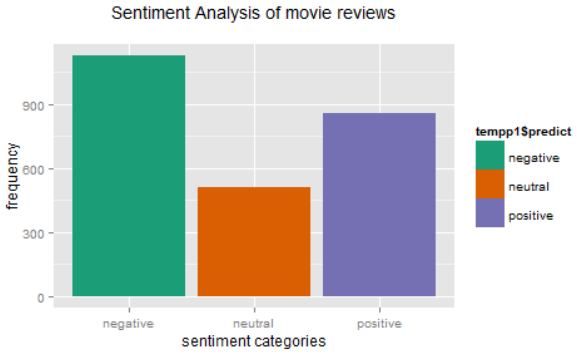
5. Classification Result

We build the classifier in R, and validate the model using the second dataset and the result is satisfactory:

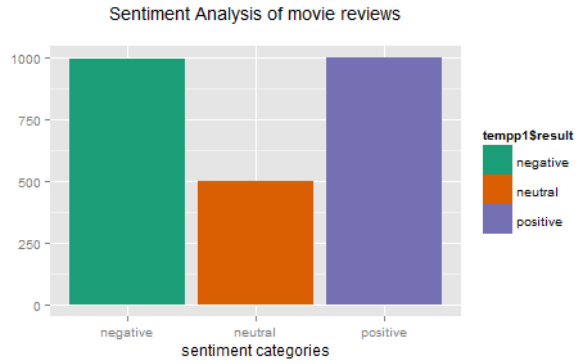
		predicted			**accuracy**	**Training Data**
		negative	**neutral**	**positive**		
Actual	**negative**	880	56	63	***88%***	1200
	neutral	88	311	101	***62%***	1000

	positive	159	145	696	***70%***	1200

We had an accuracy of 88% on predicting negative results, 62% on neutral, and 70% accuracy on positive results.



(Prediction)



(Actual result)

6. Application on Twitter API

After our classifier turns out to work well on the dataset, we tried it using the data collected from Twitter API. Because of the regulation policy, we are not able to source historical data from Twitter anymore, so the data we can collected is very limited and it's real-time data.

		predicted			**accuracy**	**Training Data**
		negative	**neutral**	**positive**		
	negative	6	6	5	35%	17
Actual	**neutral**	3	39	36	50%	78
	positive	0	26	71	73%	97

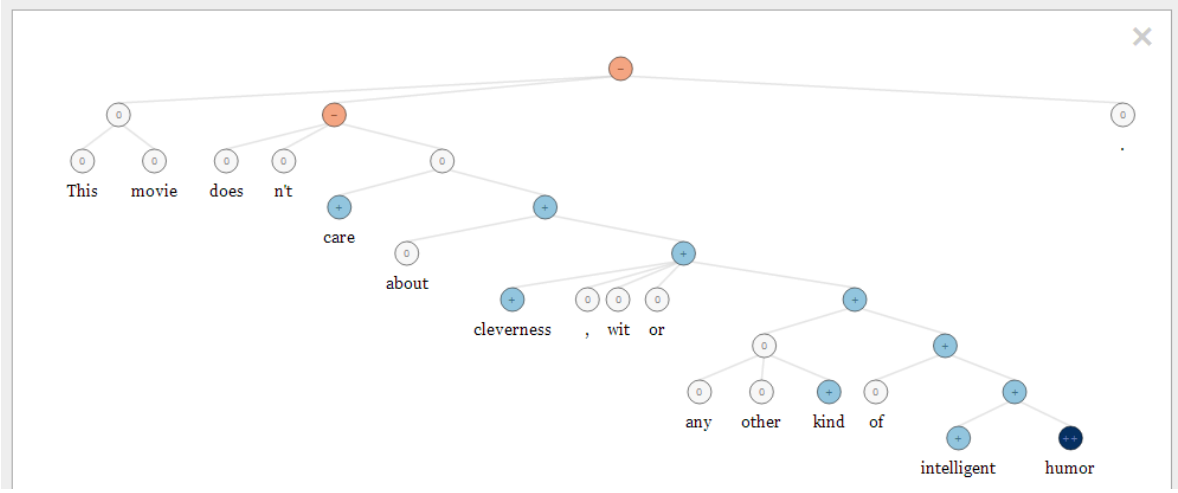
Our classifier performed poorly on the data collected from Twitter API. There are several reasons for that. First, in the process of collecting and hand-label our data, we find the data from Twitter is very "noisy". There are a lot of unrelated data even given the topic. Say the keywords is the name of the movie "Captain American", and we will get a lot of advertisements about movie tickets. Second, the data size is too small to get a reasonable result. Third, when the movie is a liked by most audience, it is very hard to a negative movie review, because the people who do not like would probably not comment on it.

7. More on this topic

There are certain situations that our classifier always failed to make the right prediction. One of these situations is when we get a movie review like this, “the movie is not interesting at all.” We are able to detect the word “interesting”, but we failed to capture the information in “not”. This movie review is apparently negative, but the classifier constantly classify it as a positive one for that it only capture the positive word “interesting”. Even if we don’t filter out “not”, it will still not yield out satisfying result.

[2]Das and Chen propose attaching “NOT” to words occurring close to negation terms such as “no” or “don’t,” so that in the sentence “I don’t like deadlines,” the token “like” is converted into the new token “like-NOT.” They look for specific part-of-speech tag patterns (where these patterns differ for different negation words), and tag the complete phrase as a negation phrase. For their dataset of electronics reviews, they observe about 3% improvement in accuracy resulting from their modeling of negations. Further improvement probably needs deeper (syntactic) analysis of the sentence.

Moreover, Stanford built some very interesting classifier on sentiment analysis using a hierarchical model.



(sourced from http://nlp.stanford.edu:8080/sentiment/rntnDemo.html)

“not”, or “n’t” will be detected as “-“(a negative), and “cleverness” as “+” (a positive). Intuitively, the combination of a “-“after “+” should be “-“negative, but their algorithm seems failed to perform such analysis.

4. Conclusion

The whole project is a start for us to further explore on sentiment analysis. There are several questions and thoughts in mind that we are still trying to figure out after we complete the project.

In this case, the accuracy of a supervised learning may largely depend on the quantity and quality of the training data. The frequency of the words differed from topics to topics. One interesting suggestions will be: if we develop a training set depend on our topics, for example, “Starbucks”, we hand-label 1000 reviews on Starbucks as our training set to predict another testing set on the same topic, and build an on-going classifier on the database, letting it keep predicting and include the newly predicted value into our training set. It would be interesting to see what our classifier will finally become, trash, super accurate or just same as before?

Another thought on this topic is that when the training set is big enough, what kind of predictive model you choose seems not so important anymore, as long as you capture the information in the data, just as the Frequentist and Bayesian. It is sometimes more import for us to collect high quality and large enough dataset from a noisy environment. We experienced some difficulties to do that when we were trying to source Twitter API. There will be definitely more about data generating every day, but at the same time the environment is becoming noisier. Which is more important, good model or good data? That is a question.