CUSTOMER REVIEW FEATURE EXTRACTION

Heng Ren, Jingye Wang, and Tony Wu

Abstract

Popular products often have thousands of reviews that contain far too much information for customers to digest. Our goal for the project is to implement a system that extracts opinions from these reviews and summarizes them in a concise form. This allows customers to quickly get an overview of a product and manufactures to efficiently process product feedbacks.

In the past, we focused on the feature extraction directly on the word level. First, we would use association rule to extract all the <noun => adj.> rules and then use Pointwise Mutual Information to judge the polarization of the adjective. Processing directly on word level neglects the information contained on the sentence level. In this paper, we discuss how to extract features by first investigate on sentence level and then dive into the word level.

Introduction

Extracting features directly on word level neglects the information contained in the sentence level. For example, there might be a sentence such as "*Today is a good day*" appearing in the reviews. If we only perform the algorithm on the word level, we would consider <day, good> as a feature of the product, but obviously it is not.

Here we propose a method that first determines which sentences in a review might contain product features. If we are confident that a sentence does have feature, we would further process the sentence using the association rule and PMI to extract the features.

To judge whether a sentence has any features, we used several Machine Learning methods. We will compare the performance of each method.

Algorithms and Implementation

Naïve Bayesian

Naïve Bayesian technique is a powerful method for classification. In our problem, the two classes are Review Sentences with product feature (C_1) and Review Sentences without product feature (C_0).

$$\Pr(C_1) = \frac{\sum 1\{C_i = C_1\}}{\text{training Size}}$$

$$\Pr(C_0) = 1 - \Pr(C_1)$$

The features we used for training are individual words within each sentence. We denote each sentence as S_i and each word as W_j. The probability of the word S_i appearing in a sentence of class C_k is given by:

$$P(W_j|C_k) = \frac{\sum_{Si} 1\{C_i = C_k\}\#\{W_j \text{ in } S_i\}}{\sum_{Wj}\sum_{Si} 1\{C_i = C_k\}\#\{W_j \text{ in } S_i\}}$$

Since some word might never appear in sentences of a particular class, we do not want to have a sentence with 0 probability of being in a particular class. We deploy a slightly modified version of Laplace Smoothing where $\lambda = 0.1$ instead of 1.

$$P(W_j|C_k) = \frac{\sum_{Si} 1\{C_i = C_k\} * \#\{W_j \text{ in } S_i\} + 0.1}{\sum_{Wj}\sum_{Si} 1\{C_i = C_k\} * \#\{W_j \text{ in } S_i\} + 0.1 * |W|}$$

Finally, we have the following Naïve Bayesian classifier to determine the probability of a sentence being in a particular class, using the words in that sentence as features:

$$P(C_1|S_i) = \frac{P(C_1)\prod P(W_j \text{ in } S_i|C_1)}{\sum_{Ck} P(C_k)\prod P(W_j \text{ in } S_i|C_k)}$$

$P(C_0|S_i) = 1 - P(C_1|S_i)$

To avoid underflow, we can alternatively use log for calculation:

$$\log(P(C_1|S_i)) - \log(P(C_0|S_i)) = \log P(C_1) - \log P(C_0) + \sum \log P(W_j \text{ in } S_i|C_1) - \sum \log P(W_j \text{ in } S_i|C_0)$$

If the value obtained above is greater than 0, we claim that the sentence is more likely to have product feature(s) than not.

The words that appear more often in featured sentences are indicative of a sentence likely to have features. The following list contains the top feature indicator words from our studies:

fits, convenient, strong, clips, packed, variable, emails, exceptional, freezes, glitches

Conversely, the following list of words indicates that the sentence is likely to not have any product features mentioned:

ordered, olympus, sell, consumers, ran, shopping, compete, register, replacing, assume

Overall, we achieve an accuracy of 78% by training on 8600 labeled sentences and testing on 600 sentences.

Spy EM

Sometimes, labeled data may be hard to come by. This is where spies become useful. Given a small set of positively labeled data and a large set of unlabeled data, we designate a subset of the positive data as spies and unleash them into the unlabeled set. Then we use the same Naïve Bayesian algorithm that we just discussed to train a fake classifier pretending that the entire unlabeled set is negatively labeled. The spies will enable us to develop a threshold for being negative. Any sentence that is more unlikely to be positive than all the spies are considered to be Reliably Negative (RN).

Spy Step in S-EM:

1. RN = NULL;
2. S = Sample(P, 15%);
3. Us = U $\cup$ S;
4. Ps = P - S;
5. Assign each sentence in Ps class label 1;
6. Assign each sentence in Us class label -1;
7. NB(Us, Ps);
8. Classify each sentence in Us using the NB classifier;
9. Determine a probability threshold th using S;
10. **for** each sentence Si $\in$ Us
11. **if** its probability Pr(1|Si) < th **then**
12. RN = RN $\cup$ {Si};

After obtaining the RN set, we will run an EM algorithm, which is essentially an iterative version of the Naïve Bayesian algorithm. The only difference here is that instead of having class labels be {0, 1}, we use a continuous range [0, 1] for the probability of each sentence being in a particular class. The indicator functions in Naïve Bayesian are now probabilities instead.

EM Step in S-EM:

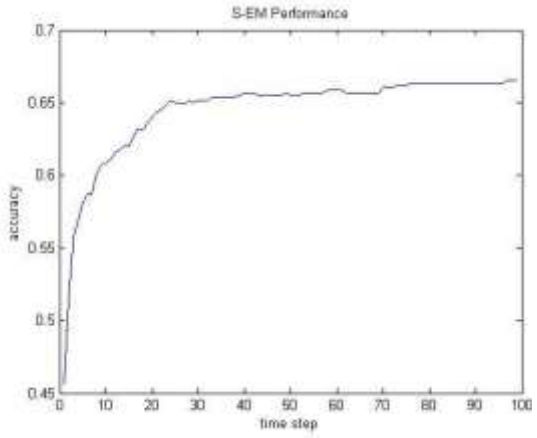
1. Assign each sentence in *P* the class label 1;
2. Assign each sentence in *RN* class label -1;
3. Each sentence *in Q* (= *U* - *RN*) is assigned a probabilistic label, *Pr*(1|*Si*).

E Step: Revise P(class|sentence)

M Step: Calculate the new P(class) and P(word|class)

4. Run the EM algorithm until it converges.

The following plot shows test set accuracy as a function of EM iterations. The EM algorithm seems to converge after approximately 100 steps.



Overall, S-EM achieves an accuracy of 67%. This is lower than Naïve Bayesian because we are pretending that the negatively labeled set is unlabeled. In a more realistic situation where we do have a large amount of unlabeled data, only using Naïve Bayesian will not enable us to solve the problem.

Rocchio Method

Rocchio Method trains the classifier by the following way. First, each document d is represented as a vector

$$d = (q_1, q_2, q_3, \dots, q_n).$$

Each element q_i represents a word w_i in the document and is calculated as

$$q_i = tf_i \times idf_i$$

where tf_i is the term frequency, the number of times that word w_i occurs in d; and idf_i is the inverse document frequency,

$$idf_i = \log\left(\frac{|D|}{df(w_i)}\right)$$

Here $|D|$ is the total number of documents and $df(w_i)$ is the number of documents where word w_i occurs at least once.

Rocchio Method could be described as follows.

1. Let $c^+ = \alpha \frac{1}{|P|} \sum_{d \in P} \frac{d}{|d|} - \beta \frac{1}{|N|} \sum_{d \in N} \frac{d}{|d|}$;

2. Let $c^- = \alpha \frac{1}{|N|} \sum_{d \in N} \frac{d}{|d|} - \beta \frac{1}{|P|} \sum_{d \in P} \frac{d}{|d|}$;

3. if $<c^+, d'> \leq <c^-, d'>$
   classify d' as negative
   else
   classify d' as positive

Here, according to [Buckley et al., 1994], we used $\alpha = 16$ and $\beta = 4$. Moreover, we used the inner product to measure the similarity of two d vectors. For each test document, if it is more similar to positive c^+, we assign it as positive.

SVM based on Rocchio

Since in Rocchio, we only used the inner product to measure the similarity of two passages, the accuracy rate was not so good. Hence now we switch to using SVM, based on the Rocchio formulation. Let the training sample set be $\{(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)\}$, where x_i is the representation of a passage in Rocchio's format, and y_i is the label for the passages. To deal with noisy labels, we used the soft margin SVM, which is

$$\min \frac{1}{2} \omega^T \omega + C \sum_i \xi_i$$

$$s.t. \;\; y_i(\omega^T x_i + b) \geq 1 - \xi_i, i = 1, 2, \dots n$$

Here C is the parameter that controls the amount of training errors allowed.

We employed different types of kernels in SVM method – linear kernel, polynomial kernel and radial basis function kernel. Based on our training set, we compared the performance of each type of kernel as follows:

Kernel	Precision	Recall	F
Linear	63.68	98.15	79.14
polynom	70.17	86.28	79.38
Gamma	68.41	100	81.12

Here we use Precision, Recall, and F-score as the measure of performance. F score takes into account of both recall and precision, F = 2pr/(p+r). The numbers above are all based on the best performance for each method after tuning the parameters.

Biased-SVM

Up until now we assume that all samples are correctly tagged. However, tagging the data is tough and tedious and if we cannot get enough tagged data, the training algorithm would face potential over-fitting problems. To solve this problem, Bing et al proposed the LPU algorithm. It uses both the tagged data and untagged data. First, it assumes that all untagged data are negative. Iteratively, the algorithm would train a SVM to classify the positive and negative samples until the ratio of positive and negative samples converges.

The problem is that, since SVM is sensitive to noises, if in any iteration the trained SVM is largely affected by noises, later ones would be worse.

Bing et. al. proposed Biased -SVM in [Bing Liu, et.al 2003]. Biased-SVM formulation uses two parameters C_+ and C_- separately to weight positive errors and negative errors. Thus the problem changes to

$$\min \frac{1}{2}\omega^T\omega + C_+\sum_i \xi_i$$

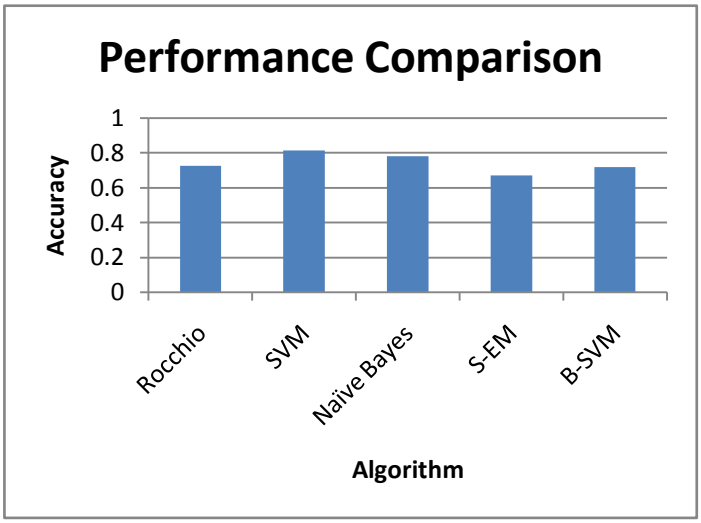
$$s.t.\ \ y_i(\omega^T x_i + b) \geq 1 - \xi_i, i = 1, 2, \ldots n$$

We can vary C_+ and C_- separately. Intuitively, we give a large value for C_+ and a small value for C_- because the untagged set, which is assumed to be negative, also contains positive data.
We used a separate validation set to verify the performance of the resulting classifier with the selected values for C+ and C-.
We used F score as the performance measure.

Kernel	Precision	Recall	F
Linear	62.36	82.11	70.89
polynom	64.39	79.49	71.15
Gamma	64.07	81.17	71.61

Evaluation

Now that we have presented the five algorithms we used to determine if a Review Sentence contains a product feature, we would like to compare their performances:



It seems that for this particular task, SVM with no untagged samples outperforms all other algorithms by achieving an accuracy rate of over 80%. When using spy samples, the performance of SVM decreases largely. However, since we also used iterative algorithm in SVM, we can see that the SVM's performance is still better than S-EM.

Combination

To close the loop, we would like to present some results outside the scope of the project focus, just for completeness. After obtaining product features, we would like to know if customers liked the particular feature or not. This can be done easily with an algorithm called PMI.

Point-wise Mutual Information (PMI) algorithm uses mutual information as a measure of the strength of semantic association between two words. The Point-wise Mutual Information between two words, W_1 and W_2, is defined as follows:

$$\mathrm{PMI}(W_1, W_2) = \log\left(\frac{P(W_1, W_2)}{P(W_1) * P(W_2)}\right)$$

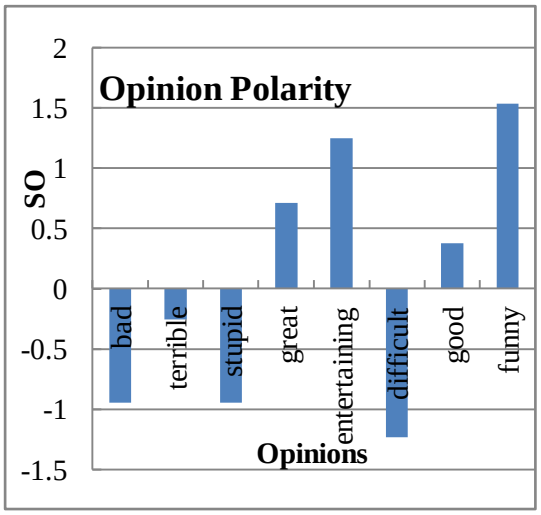
Where $P(W_1,W_2)$ is the probability that W_1 and W_2 both occur in the same phrase.

Semantic Orientation of a phrase is calculated as follows:

SO(phrase) = PMI(*phrase*,"*excellent*")

– *PMI*(*phrase*,"*poor*")

We can use Semantic Orientation to Determine the polarity of a particular word. If the word appear more often with excellent, it is more likely to have a positive connotation. The following plot shows some examples:



Finally, once we have obtained the product features as well as their relative polarity from the customer reviews, we can sort them and present them in a summary form as demonstrated below:

PRODUCT FILE NAME : D-Link DWL-G120
There are 40 review(s) total.
PROS : setup (5.0)
connection (4.0)
price (4.0)
card (2.0)
connections (2.0)
CONS: performance (-5.0)
port (-2.0)
sensitivity (-2.0)

Conclusion

In this paper, we showed a two-step method for extracting features from customer reviews. As the first step, we would judge whether each sentence has a feature or not; if it does, we would further process the sentence and extract the features. The first step is mainly based on Machine Learning methods and the second step is mainly based on Data Mining and Natural Language Processing Methods. We have demonstrated in this project that it is tractable to determine customer opinions on individual product features. When all training data are correctly tagged, we showed that SVM is the best and when not all training data is tagged, S-EM and B-SVM almost have the same performance.