CS534 Machine Learning Project Report

Sentiment Analysis on Amazon Products

Kaiwen Zheng
Haoyu Zhang
Xiao Tan

November 2017

Introduction

Nowadays, all kinds of products would be sold on Internet markets, such as Amazon, ebay, taobao. Everyone can write their own user experience or review on the website to help other customers. Meanwhile, reviews would help market know each product whether it is good enough to be recommended. Some reviews would mark their points that is not rely on their reviews. Base on the real sentiment of reviews, people can figure out whether a review is positive and find out the percentage of the real positive review in a whole product's review. Using these data, markets would decide the strategy. Therefore, we consider that classifying users' reviews of products by sentiment analysis is a way to determining whether a review is positive or negative. Also for different kind product, it may have some different sentiment words and some may be same. In this case, use the sentiment analysis would help us to find the different and the same word in the sentiment determination. The review sentiment analysis system can also extract efficient data and evict useless data, such as neutral opinions. People used sentiment analysis on the other domain years ago.

The Sentiment analysis is referred to the use of natural language processing, text analysis, computational linguistics, and biometrics to identify, extract, quantify and study affective states and subjective information. Usually, the sentiment of the review would figure out by the key words. However, sometimes it would be expressed in a subtle manner. Using sentiment analysis would help people find these words, and more accurately predict whether the review is positive. In this paper, we would use the review on the amazon, where has mass data. We would use the feature extraction method methods to mark a review and then use machine learning methods to build the prediction model.

Existing Approaches

As the previous work, the sentiment analysis would use the knowledge-base technology which would would base on the knowledge that would be error by human cognition. At the 2002, Pang B. and et al. publish their paper about the sentiment analysis using on the field of movie-review. In that paper, they filter their data to get features, then they get features frequency or present to apply on their machine learning algorithms. The machine learning algorithms they used are Naive Bayes, Maximum Entropy and support vector machine. With these algorithms to train data and find appropriate model for their system, they may find the different results by different models. They get about 82.9% accuracy with their best model. Moreover, AlecGo and et al. achieved the sentiment analysis at the domain of Twitter. Their feature extractor is almost the same as Pang B. did. For their machine learning methods, they are also Naive Bayes, Maximum Entropy and support vector machine. The best accuracy they got is 83%, which similar to Pang B's. As these two papers showed to us, they collected words frequency or present as their features, then they apply them to a machine learning algorithm. In the end, they got almost 80% accuracy. So, we decided to choose one of these two parts, which feature extractor or machine learning methods, or both of them to try to improve its accuracy.

Our Approaches

Our dataset is gotten from *http://jmcauley.ucsd.edu/data/amazon/*, which is extracted by J. McAuley. The data would be stored as raw data in json format. We need to extract the review data from a case then

analyse each review and exact feature from each one. There is a massive review of amazon products. Due to the size of the data, we can get a class product and split them as training set, dev set and test set. And these raw data include source 1-5 stars. Then we can pick 4,5 star review as positive label and 1,2 star review as negative label first. If time is available, we could make the 3 star as neutral to find the result .

The dataset is from 5-core which all which all users and items have at least 5 reviews. The format is one-review-per-line in json. The sample review is following:


```
{
       "reviewerID":"A2SUAM1J3GNN3B",
       "asin":"0000013714",
       "reviewerName":"J.McDonald",
       "helpful":[2,3],
      "reviewText": "I bought this for my husband who plays the piano.  He
is having a wonderful time playing these old hymns.  The music  is at times
hard to read because we think the book was published for singing from more
than playing from.  Great purchase though!",
        "overall": 5.0,
        "summary": "Heavenly Highway Hymns",
        "unixReviewTime": 1252800000,
        "reviewTime": "09 13, 2009"
}
```


It is a type of dict. We just need the information “reviewText” which is the text of review and “overall” which is the rating of product.

Due to the data we get, we need to deal the raw data. In the reviews, there are plenty of punctuation expressions, but these punctuations are useless for our prediction. So we will remove punctuations at first. Moreover, there are some useless words, these words won’t give any information to our model, so we remove them as well, by using regular expression.

Then, besides the words frequency feature extracting method, we can apply a better way to do feature extraction, which is TF-IDF, short for term frequency–inverse document frequency. It is used to evaluate how important a word is in a document in a collector or a corpus. The basic idea of TF-IDF is that multiply term frequency and inverse document frequency to calculate words weight. If a word shows high frequency in one file, but low frequency in the whole document, it tells this word is important, so we will assign a high weight to this word. The TF and IDF like the following equations show:

$$tf_{ij} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

$tf_{i,j}$, term frequency, means frequency of given word in files. $n_{i,j}$is the number of times that term t_i occurs in documents d_j. the denominator is the all times of all terms occurs in document d_j.

$$idf_i = log \frac{|D|}{|\{j: t_i \in d_j\}|}$$

idf_i, inverse document frequency, is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient. $|D|$is the number of files in the corpus. $|\{j: t_i \in d_j\}|$is the number of files which include the term t_i. Often, it will do smoothing, in case of t_i does not exist in corpus and this will lead to a division-by-zero.

$$tfidf_{i,j} = tf_{i,j} \times idf_i$$

$tfidf_{i,j}$a high weight in tf-idf is reached by a high term frequency and a low document frequency of the term in the whole corpus. Then the tf-idf tend to filter out common terms.

We use the perceptron, and Primal SVM(pegasos) as the classifying machine to identify the review. By choosing different C in Pegasos, we would choose the best model we get. Support Vector Machine (SVM) is an efficient and useful classifying technique, but it would slower than online model due to O(n^2) calculation. We choose the pegasos that is linear kernel as the machine and use the TF-IDF value as each review's feature. Similarly, we choose the average perceptron as another model we use. This two different model is different with the loss function. The problem can be define to find the minimize problem:

$$\min_w \frac{\lambda}{2}\|w\|^2 + loss\ function$$

Therefore, the loss function in the average perceptron is 0-1 loss , after we get the average result to get the model. Similarly, the loss function of pegasos hinge loss. Base on that we can get the update function:

$$w_{t+1} \leftarrow \left(1 - \frac{1}{t}\right) w_t + \eta_t [y_{i,t}\langle w_t, x_{i,t}\rangle < 1] y_{i,t} x_{i,t}$$

where $\eta_t = \frac{1}{\lambda t}$.

Results and Discussions

First, because of the huge dataset, which over 100 millions reviews, we decided to choose parts of this dataset in "videogame" category as our training set and dev set, which may exist some potential words that show the customers' sentiment. So we remove the some words which start as number and seems useless word in sentence such as 40d or 1987. In table 1, it should our result. It show that it would get the a little bit improvement, when remove some words.

Table 1. Classifying Error Rate On "Tools" Dataset (size=20000)

Model	Average Perceptron	SVM(Pegasos) C=100	SVM(Pegasos) C=2
Unigram + remove some words	16.02%	19.48%	14.62%
Unigram	16.05%	19.62%	14.72%

Table 2. Top Ten Positive And Negative Words (size=20000)

Model	Average Perceptron	SVM(Pegasos) C=100	SVM(Pegasos) C=2
Unigram + remove some words	Positive 10:definately, superstar, combos, correct, best, aching, amazed, textures,ix ,types Negative 10: bowie, worst, leisure, mas, labyrinth, stormtroopers, relive, weite, rune, promises	Positive 10:correct, superstar, definately, amazed, owner, aching, squaresoft, best, textures, types Negative 10: bowie, worst, leisure, mas, weite, labyrinth, stormtroopers, promises, rune, compaired	Positive 10: best, the , great, awesome, love, and, highly, addictive, well, amazing Negative 10: worst, horrible, terrible, boring, awful, sucks, unplayable, frustrating, disappointment, worse
Unigram	Positive 10: superstar, definately, amazed, correct, types, combos, scared, awsome, best, aching Negative 10: bowie, worst, leisure, rune, worse, weite, frustrating, stormtroopers, relive, plod	Positive 10:definately, superstar, amazed, correct, aching, mighty, best, owner, types, exchanged Negative 10: bowie, leisure, worst, stormtroopers, rune, labyrinth, promises, reunion, oversahdowed, suxs	Positive 10: best, the, great, awesome, love, and, highly, addictive, Well, amazing Negative 10: worst, horrible, terrible, boring, awful, sucks, frustrating, unplayable, disappointment, worse

Also, we see the different top ten positive and negative words, it just little bit different. In other words, the words we remove has no or just a little impact on the sentiment. Because the videogame data may be trick. We use the dataset about tools review.

Table 3. Classifying Error Rate On “Tools” Dataset (size=30000)

Model	Average Perceptron	SVM(Pegasos) C=100	SVM(Pegasos) C=2
Unigram + remove some words	11.87%	12.38%	9.63%
Unigram	11.88%	12.18%	9.68%

Table 4. Top Ten Positive And Negative Words (size=30000)

Model	Average Perceptron	SVM(Pegasos) C=100	SVM(Pegasos) C=2
Unigram + remove some words	Positive 10: great, perfectly, best, highly, perfect, excellent, easy, assuming, love, helps Negative 10: probable, vibrates, disappointing,	Positive 10: great, perfect, best, excellent, highly, perfectly, easy, situation, pricey, assuming Negative 10: probable, useless, vibrates, returned,	Positive 10: great, easy, perfect, excellent, best, perfectly, highly, love, and, well Negative 10: not, returned, useless, return,

	vision, returned, not, joke, useless, fell, poorly	vision, disappointing, joke, fell, cheaply, worst	poor, worst, poorly, disappointing, returning, waste
Unigram	Positive 10: great, best, perfect, excellent, perfectly, highly, easy, situation, love, assuming Negative 10: probable, useless, vision, vibrates, disappointing, not, joke, worst, fell, lacks	Positive 10: great, easy, perfect, excellent, perfectly, best, highly, situation, amazing, love Negative 10: probable, disappointing, vision, useless, vibrates, returned, joke, lacks, cheaply, not	Positive 10: great, easy, perfect, excellent, best, perfectly, love, highly, well, and Negative 10: not, returned, useless, return, poor, poorly, worst, disappointing, waste, returning

In the above two table, it can show that the result would same as using the videogame dataset. In the model, if just remove some stop words, it may has a little influence on the model learning (a little bit improvement). About the tools, it contain about 18.97% negative review of the total, contain 17.23% negative review of the total video game. In the videogame dataset, we may get the worst model that would get 19.23% which is higher than the negative review percentage.

Therefore, consider that using the bigram with unigram can improve the performance. Due to the bigram feature size, just use 10000 data on the tool. s. From the table 5, we can find that using the bigram can't not usefully improve the performance. However, when it contain combine the unigram, it would get the improvement on the result.

Table 5. Classifying Error Rate On "Tools" Dataset (size=10000)

Model	Average Perceptron	SVM(Pegasos) C=3	SVM(Pegasos) C=2
Unigram + Bigram	10.82%	9.42%	9.78%
Unigram	13.03%	11.48%	11.32%
Bigram	13.63%	10.97%	11.48%

Conclusions

By many times test, finding that using unigram and bigram would help the model have the better performance. Actually, if remove some stop words, the machine would also learn from the rest words, remove some useless words would have no impact on the result. Then positive and negative words would not change so much in the result. However, using some videogame dataset, it exist some noisy that can influence the top 10 words result. When C=100, the majority top 10 result would different with C=2. In the "stable" dataset tools, the different would be small. When using the bigram to replace the original unigram, it would not get the improve. Probably, it would reduce the sentiment behind a word. Therefore

using bigram and unigram as feature engineer would help the machine learn the model better. We also consider apply the POS in the feature engineer, but consider that some words have different words form, we just keeping it. In the result that shows in table 4. “Poor” and “poorly” contains the POS in the form, but such as “return” and “returned” would be same. In the future word, it would return this kind of form influence on the result. In our result, we would get the good result with using bigram and unigram and pegasos C=3 we would get 9.42% error rate.

Reference

[1] B. Pang, L. Lee, and S. Vaithyanathan. Trumbs up? Sentiment classification using machine learning techniques. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 79–86, 2002.
[2] AlecGo, Richabhayani, LeiHuang. 2009. Twitter Sentiment Classification using Distant supervision [R]. Technical report, Stanford Digital Library Technologies Project.
[3] R. He, J. McAuley. Modeling the visual evolution of fashion trends with one-class collaborative filtering. WWW, 2016
[4] J. McAuley, C. Targett, J. Shi, A. van den Hengel. Image-based recommendations on styles and substitutes. SIGIR, 2015