Review Extraction via Mining Microblogs

Weiyue Huang, Chenhao Zhu, and Kan Ren

Shanghai Jiao Tong University, Shanghai, China
{hwy, chzhu, kren}@apex.sjtu.edu.cn

1 Introduction

Recently, social trend becomes more and more obvious in our daily life. Almost every app or website develops their social module to activate the user community. Large number of people read articles, communicate, post ideas and find other persons on social media sites.

Social Network (SN) produce great volume of data every day. In the report[1], some large social media digest petabyte level of information in one day. Moreover the information produced by SN contains great value.

However, mining social media data requires some professional techniques. Machine learning algorithms may help to discover latent patterns in social media data. There are some applications in this field. Pattern mining such as association rules[1] is used for user browsing behavior mining. Clustering methods (e.g. K-means) are applied in interest group finding. Recommender systems are built in many e-commerce websites[6]. IR techniques are used for product review extraction and clustering[8].

As is known, there is gigantic volume of valuable information in SN. Many people may find whatever they want in SN. However, noise information is filled in data flows of SN. Algorithms should be well designed to filter out noise while preserving useful information. Moreover, the information should be cleaned and help to make more value in the future.

Our goal is to find actual review comments about some real things (e.g. restaurants) posted in SN and discard advertisements, meaningless articles or any other noisy information. We focus on Weibo[2] website which is one of the largest SN in China. We apply 3 models (SVM, Naive Bayes and Decision Tree) for review classification. Different features such as mutual information, topic modeling are used to capture the latent information of data. We also compare the performance of different models with different parameters. Experiments show comprising results of our system.

2 Task

Microblog Review Extraction task is to find real reviews for restaurants in microblog data and filter out noise. Given a set of microblogs, our system will judge whether each microblog is a real review.

[1] http://www.theregister.co.uk/2012/11/09/facebook_open_sources_corona/
[2] http://www.weibo.com

3 Dataset

The dataset which we used in this project is collected from Weibo website, one of the largest social network website in Chine. The dataset contains 123,136 microblogs along the time line with some related information such as id, time, publishing device, number of people who retweet the weibo and POI (Point of Interest). And the contents of microblogs are segmented by general segmentation tool. We also manually labeled 8,498 of them whether a microblog is a review or not. In this project, a K-fold cross validation method is used to improve our experiment result.

Details of our data set is shown in Table 1.

Table 1. Data set and examples

Name	Description	Example
weiboId	the id of the weibo	M_Blu8QBwGG
like	the number of people who like this weibo	0
retweet	the number of people who retweet the weibo	1
comment	the number of the comments of the weibo	0
pic	pictures in the weibo	1
POI	Point of Interest	B20...55 (in short)
from	the device from which the weibo is published	iPhone Client
time	the publish time of the weibo	2014-11-22 08:00:00.0
content	segmented weibo content	Icecream is so delicious

4 Features

From the dataset, we get several features about a microblog text. To make the most of these features, we pay more attention to the most important one, which is the content of a microblog text. In our work, we adapt several feature selection methods which are commonly used in text categorization.

4.1 Mutual Information

Given a category c and a term t, let A denote the number of times c and t co-occur, B denotes the number of times t occurs without c, C denotes the number of times c occur without t, and N denotes the total number of documents in c. The mutual information criterion between t and c is defined as Equation 1:

$$MI(t,c) = \log \frac{P(t,c)}{P(t) \times P(c)} \tag{1}$$

and is estimated using Equation 2:

$$MI(t,c) \approx \log \frac{A \times N}{(A+C) \times (A+B)} \tag{2}$$

These category-specific scores of a term are then combined to measure the goodness of the term at a global level. Let $\{c_i\}_{i=1}^{m}$ denote the set of categories in the target space. Typically it can be calculated as Equation 3.[9]

According to [4], the mutual information compares the joint probability of observing t and c together with the probabilities of observing t and c independently. $MI(t,c)$ has a positive value when there is a genuine association between t and c. It has a zero value when there is no significant relationship between t and c and a negative value when t and c are in complementary distribution.

$$MI_{avg}(t) = \sum_{i=1}^{m} P(c_i) MI(t,c_i) \tag{3}$$

In our method, after the computation of these criteria, thresholding is performed to achieve the desired degree of feature elimination from the full vocabulary of a document corpus. Here we use Equation 4 to formulate it.

$$\bigcup_{i=1}^{m} \{t | MI(t,C_i) \textit{ is the top } K \textit{ largest ones in } \{MI(W,c_i)\}\} \tag{4}$$

4.2 Topic Modeling

In machine learning and natural language processing, the topic model is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents[3] (microblogs). Intuitively, given that a document is about a particular topic, one would expect particular words to appear in the document more or less frequently.

A document typically concerns multiple topics in different proportions. In our settings, we use topic modeling method to discover several abstract "topics" contained inside microblogs. Then we try to estimate topic probability distributions of given microblogs (i.e. documents). We could obtain a feature vector after inference topic distribution for each microblog.

We note the feature vector as $\boldsymbol{x}_i = (x_1, x_2, \ldots, x_k, \ldots, x_K)^T$ where K is the number of topics and $\boldsymbol{x}_i$ is the feature of the ith microblog. Each element in the vector is a probability of a specific topic with regard to the microblog. Here $x_k = P(t_k|m_i), k = 1, 2, \ldots, K$.

After generation of feature vectors for the microblogs in the train dataset, we use some classification method to classifiy microblogs (as whether thie microblog is a real revew or not). In our experiments, we use Support Vector Machine to achieve this goal.

Figure 1[4] shows the plate notation of Latent Dirichlet Allocation (LDA)[2]. LDA is a generative model which allows sets of observations to be explained

[3] http://en.wikipedia.org/wiki/Topic_model
[4] http://en.wikipedia.org/wiki/Latent_Dirichlet_allocation

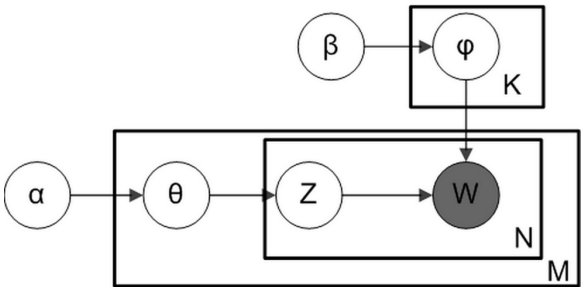

Fig. 1. Plate Notation of LDA

by unobserved groups (latent semantic topics) that explain why some parts of the data are similar. In our settings, the observations are words collected in microblogs, it posits that each microblog is a mixture of a small number of topics and that each word's creation is attributable to one of the microblog's topics.

In LDA, the topic distribution is assumed to have a Dirichlet prior. With plate notation in Figure 1, the dependencies among the many variables can be captured concisely. The boxes are "plates" representing replicates. The outer plate represents microblogs, while the inner plate represents the repeated choice of topics and words within a microblog.

M denotes the number of microblogs, N the number of words in a microblog. Here α is the parameter of the Dirichlet prior on the per-microblog topic distributions, β is the parameter of the Dirichlet prior on the per-topic word distribution, θ_i is the topic distribution for microblog i, ϕ_k is the word distribution for topic k, z_{ij} is the topic for the jth word in microblog i, and w_{ij} is the specific word.

The w_{ij} are the only observable variables, and the other variables are latent variables. ϕ is a $K \times V$ (V is the dimension of the vocabulary) Markov matrix (transition matrix), and each row of which denotes the word distribution of a topic.

The generative process is as follows. Documents are represented as random mixtures over latent topics, where each topic is characterized by a distribution over words. LDA assumes the following generative process for a corpus D consisting of M microblogs with length N_i:

- Choose $\theta_i \sim \mathrm{Dir}(\alpha)$, where $i \in \{1, \ldots, M\}$ and $\mathrm{Dir}(\alpha)$ is the Dirichlet distribution for parameter α.
- Choose $\phi_k \sim \mathrm{Dir}(\beta)$, where $k \in \{1, \ldots, K\}$
- For each of the word positions i, j, where $j \in \{1, \ldots, N_i\}$, and $i \in \{1, \ldots, M\}$
  - Choose a topic $z_{i,j} \sim \mathrm{Multinomial}(\theta_i)$.
  - Choose a word $w_{i,j} \sim \mathrm{Multinomial}(\phi_{z_{i,j}})$.

We use EM[5] algorithm to train topic model with LDA. We omit the detail of the inference.

5 Models and Techniques

5.1 Naive Bayes

Naive Bayes is a conditional probability model. Given a problem instance to be classified, represented by a feature vector $\mathbf{t} = (t_1, t_2, ..., t_n)$, it assigns to the instance probabilities for each category. Then the category result is determined by choosing the one with the highest probability. The bayes rule in plain English is showed in Equation 5 .

$$posterior = \frac{prior \times likelihood}{evidence} \tag{5}$$

Namely, we can formulate it using Equation 6.

$$p(c_i|\mathbf{t}) = \frac{p(c_i) \times p(\mathbf{t}|c_i)}{p(\mathbf{t})} \tag{6}$$

Naive Bayes assumes the conditional independence among the features. So according to the chain rule, we can get the Equation 7.

$$p(c_i|t_1, t_2, ..., t_n) \propto p(c_i) \times p(t_1|c_i) \times p(t_2|c_i) \times ... \times p(t_n|c_i) \tag{7}$$

5.2 SVM

SVM is short for Support Vector Machine, it is a useful technique for data classification. A support vector machine constructs a hyperplane in high-dimensional space. A good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class, which is called the function margin, since in general the larger the margin, the lower the generalization error of the classifier[5].

According to [3], given a training set of instance-label pairs $(\mathbf{x}_i, y_i)$,$i = 1, ..., l$ where $\mathbf{y} \in \{1, -1\}^l$, the support vector machines require the solution of the optimization problem showed in Equation 8.

$$\begin{aligned} min_{\mathbf{w},b,\xi} \quad & \frac{1}{2}\mathbf{w}^T\mathbf{w} + C\sum_{i=1}^{l}\xi_i \\ subject\ to \quad & y_i(\mathbf{w}^T\phi(\mathbf{x}_i) + b) \geq 1 - \xi_i \\ & \xi_i \geq 0 \end{aligned} \tag{8}$$

5.3 Decision Tree

A decision tree is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences. In our settings, we use decision tree to model the token-appearance decision for classifying microblogs.

[5] http://en.wikipedia.org/wiki/Support_vector_machine

Decision tree learning uses a decision tree as a predictive model which maps observations about an item to conclusions about the item's target value. We use C4.5[7] algorithm to train the decision tree model. C4.5 builds decision trees from a set of training data using the concept of information entropy. The training data is a set $S = s_1, s_2, ...$ of already classified samples (microblogs). Each sample s_i consists of a p-dimensional vector $(x_{1,i}, x_{2,i}, ..., x_{p,i})$, where the x_j represent attributes or features of the sample, as well as the class in which s_i falls. In our settings, $x_{p,i}$ represents a vectorized microblogs.

At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain g. The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sublists.

Our decision tree result is presented in the experiment part.

6 Experiment

In this section, we introduce our experimental details and show our performance on the test data set. The evaluation criteria we used is defined in Equation 9.

$$\begin{aligned} precision &= \frac{\#\ of\ true\ reviews}{\#\ of\ true\ reviews + \#\ of\ false\ reviews} \\ recall &= \frac{\#\ of\ true\ reviews}{\#\ of\ true\ reviews + \#\ of\ false\ non-reviews} \\ F1-Score &= \frac{2precision\ \times\ recall}{precision + recall} \end{aligned} \tag{9}$$

6.1 Baseline

To evaluate the performance of our methods, we design a simple experiment as the baseline. In this method, we simply predict the label "1" to each test data. This method gets a F1-score of 0.46 with a precision score of 0.30 and a recall score of 1.0.

6.2 Statistic-based Naive Bayes

Intuitively, the words with higher frequences are much more important. As a result, firstly we make a statistics of the occurrences of all the words. And Then we use a threshold to select the meaningful words. In our method, all the rest words are seen as a same word, which we can call it as the unknown word. And meanwhile, it also works for the new words existing in the test dataset. Since we use the co-occurrence information between a word and a target label, we make a smoothing to avoid the situation that the joint probability is zero. Figure 2 shows the performance with different threshold.

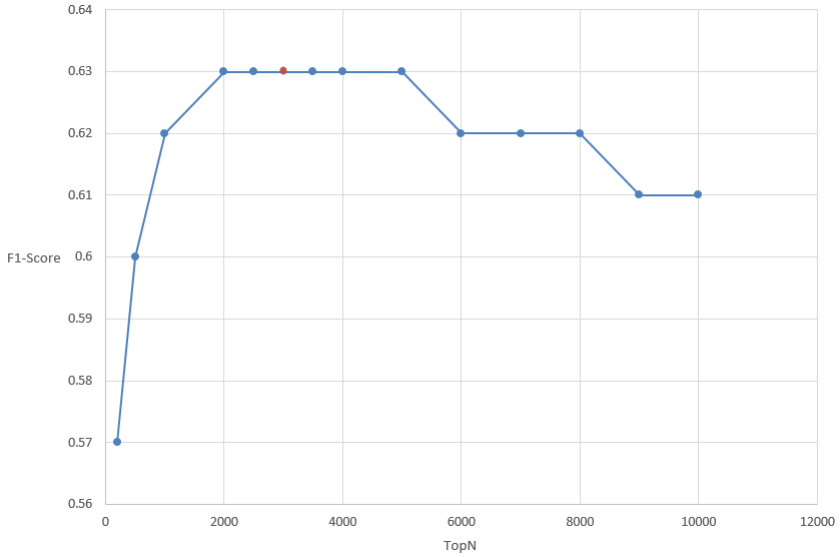

Fig. 2. Result of Statistic-based Naive Bayes

As we can see in Figure 2, this method achieves the best performance when the threshold is set as 2500. At this point, we get a F1-score of 0.63 with a precision score of 0.63 and a recall score of 0.63.

In this method, we also consider the influence of stop words by ignoring the top N words having higher frequences. However, it doesn't make sense. As showed in Figure 3, although we achieve the best performance when we ignore the top 20 words, the F1-score only improve 1 percentage. When we set the parameter N a little value, such as 50, it even performs worse. That is to say, in our experimental dataset, the most common words are meaningful to the category result.

6.3 Mutual Information

In this method, we use all words' mutual information in a document as the criteria. We formulate it using the Equation 10.

$$MI_{avg}(c) = P(c) \sum_{i=1}^{n} MI(t_i, c) \tag{10}$$

As the same situation in the method of statistic-based Naive Bayes, the size of document corpus is too big. It has a lot of noises. So we need to pick up the most meaningful ones. In our work, we use the strategy expressed in Equation 4. We calculate the performance with a different parameter N indicating the number of features, and the result is showed in Figure 4.

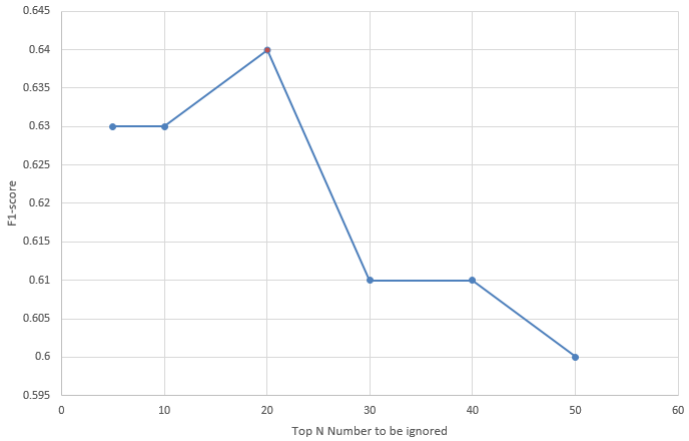

Fig. 3. Result of Statistic-based Naive Bayes Ignoring top N Words

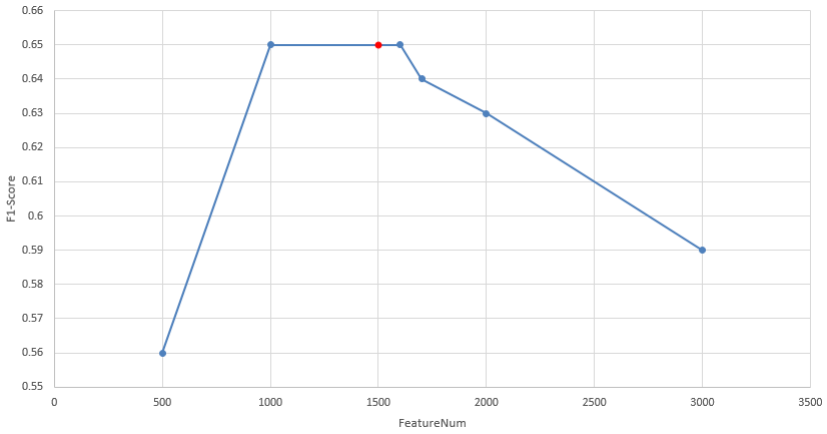

Fig. 4. Result of Mutual Information

As shown in Figure 4, this method achieves the best performance when the number of features is around 1500. And at this point, we get a F1-score of 0.65 with a precision score of 0.55 and a recall score of 0.81.

6.4 SVM using Mutual Information

In this method, we use the mutual information as features and try to use SVM to make the classification. Just using the same strategy as Section 6.3, we get the selected keywords. However, the mutual information value of each word is calculated in a different way since it needs to include all relatedness with different target categories. Here we use the method defined in Equation 3.

Firstly we try different parameters which indicates the top number of keywords we used and the result is showed in Figure 5. The kernel function we used in Figure 5 is the linear one, and we can see that it achieves the best performance when the parameter is set as 2000. At this point, it achieves a F1-score of 0.64 with a precision score of 0.76 and a recall score of 0.55.

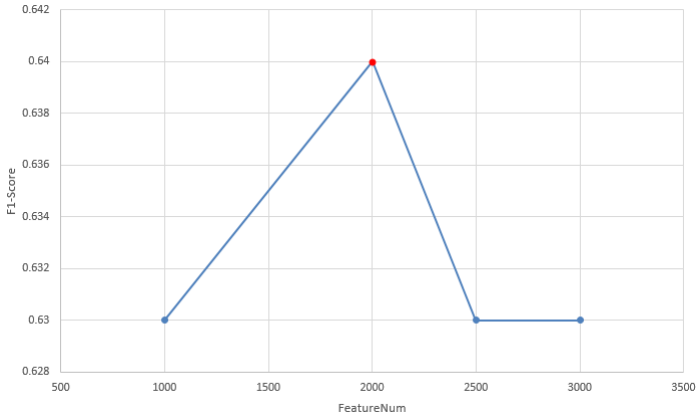

Fig. 5. Result of SVM Using Mutual Information

At the same time, we adapt several kernel functions to test the performance with the number of keywords set as 2000. As showed in Figure 6, we try three types of kernel functions. Namely, a sigmoid kernel function, a radial basis function and a linear one. It shows that we get the best performance with the linear one. At the same time, the linear kernel has the best performance on speed.

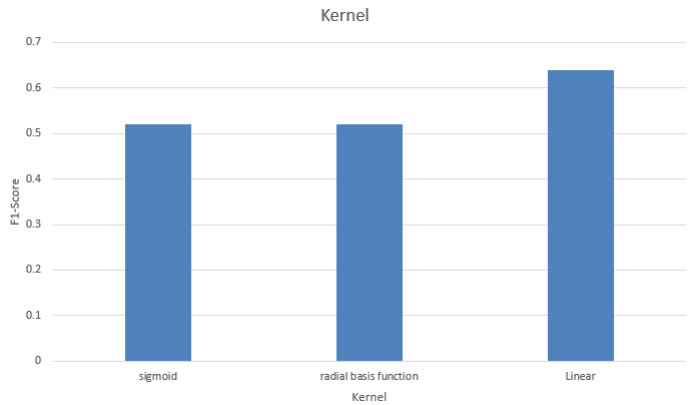

Fig. 6. Result of SVM Using Mutual Information with different Kernel Functions

Since we have other information in our data set, such as the like number, the comment number, the retweet number. We make a comparison experiment using these information. Since these features are represented by Integer, we try two methods to represent them. The first one is using a binary value to indicate whether the corresponding number is zero or not, the second one is to normalize them as a percentage. The result is showed in Figure 7. We can see that they have little difference. That is to say, compared with the content information of the microblog text, the other information in the data set doesn't make sense.

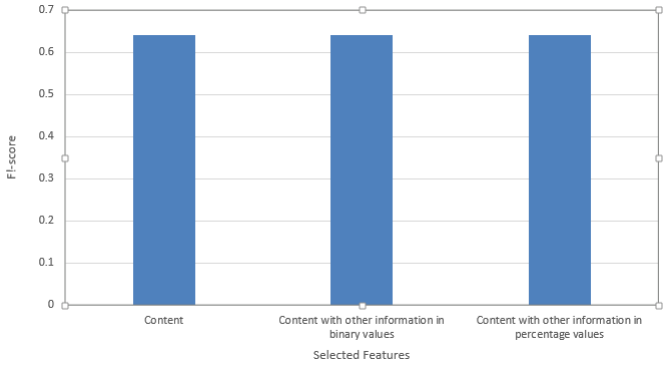

Fig. 7. Result of SVM Using different features

6.5 Topic Modeling with SVM

In the training phase, we get the training data set which contains microblogs and corresponding labels.

Firstly we train a topic model using LDA method, then we get K topics and corresponding topic distributions with regard to microblogs. For each microblog, we generate a K-dimensional vector from the topic distribution. The value of each element in the vector is $P(t_k|m_i)$ which means the probability of the ith microblog belonging to the kth topic.

After topic modeling, we use a classifier to train a model with the K-dimensional topic feature vector. In our settings, SVM classifier is used and we choose the linear kernel.

The abstract topics generated by topic modeling is shown in Figure 8. Each topic is followed by a list of top 10 words with regard to $P(t_k|w_n)$. The first column is the topic ID. And the second column is the distribution probability for the given topic against this microblog.

In Figure 9, the result shows the diversification of accuracy with regard to different topic numbers. Here we may find that the topic number has great influence on classification performance. 10 topics is not sufficient. So we set the topic number as 100 in the final test.

TID	Probabity	Word List
0	0.001	1 (75) 2 (62) 3 (58) 后 (53) 放入 (50) 倒入 (45) 分钟 (42) 鸡
1	0.002	火锅 (161) 新 (108) 四川 (89) 生仕鱼林港 (81) 辣 (79) 中心 (7
2	0.002	三 (161) 牛蛙 (101) 顾 (89) 车 (87) 馆 (77) 神 (75) 很 (72) ch
3	0.001	虹口店 (77) 中 (71) 1 (68) 唐宫膳 (67) 宝箱 (54) 挖 (42) 专辑
4	0.010	不 (1490) 人 (941) 会 (694) 都 (444) 爱 (286) 想 (260) 很 (23
5	0.002	house (159) flour (154) 穀屋 (152) 接 (106) 链 (102) i (91) m
6	0.001	1 (100) 2 (66) 花 (49) 微乐 (45) 感觉 (42) 价格 (39) 元 (37)
7	0.004	元 (125) 手机 (91) 下载 (88) 12 (88) 地址 (84) 5 (74) 2 (70)
8	0.402	吃 (992) 不 (321) 都 (260) 很 (254) 好 (246) 上海 (244) 还 (2
9	0.002	好 (63) 里 (46) 中 (45) 漂亮 (44) 都 (44) 快来 (36) 图 (35) 试
10	0.005	分享 (445) 明天 (156) 链接 (134) 址 (128) 图片 (121) 天堂 (91)
11	0.001	投票 (149) 参与 (139) 快来 (133) 投 (132) 表态 (119) 发起 (11
12	0.001	名 (96) 男 (51) 女 (49) 星座 (43) 羊 (36) 双 (35) 最 (29) 第二
13	0.002	餐厅 (290) 店 (164) 新 (81) 茶 (71) 南京 (48) 查 (47) 东路 (4
14	0.001	领 (70) 都 (59) 保暖 (51) 领到 (44) 防寒 (42) 费 (40) 无聊 (3
15	0.265	秦时 (245) 做 (151) 破 (129) 百万 (128) 明月 (123) 霸王 (123)
16	0.002	红 (161) 房子 (148) 西菜馆 (90) 谈 (78) 情 (77) 好 (66) 炉边

Fig. 8. Topic with High Probability Words

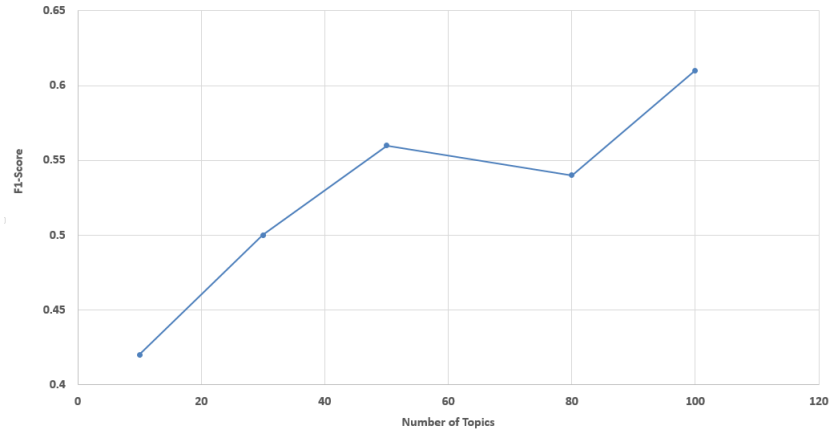

Fig. 9. Performance with Topic Number

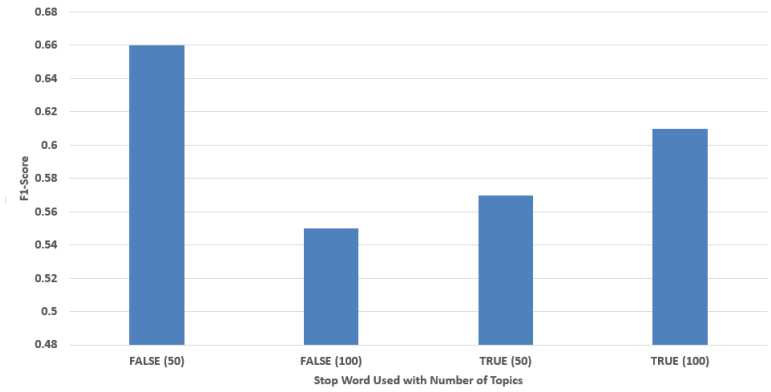

Fig. 10. Performance with Topic Number

At first, we use stop word list to filter out those words which appears very frequently. Figure 10 shows the performance with and without stop word removal operation. However we may find that the performance without stop word removal is better than removing stop words with topic number set to 50. When topic number is set to 100, the performance shows that stop words play some important role in training.

TID	Probabity	Word List
0	0.678	的 (2555) 了 (1135) 吃 (848) 是 (645) 一 (536) 上海 (472) 小 (456) 我 (4
1	0.078	我 (1396) 在 (1347) 餐厅 (604) 这里 (579) 店 (490) 广场 (300) 栗子 (205)
2	0.005	的 (2741) 是 (1059) 不 (1017) 你 (929) 我 (897) 一 (882) 了 (863) 有 (46
3	0.001	的 (270) 红 (210) 房子 (188) 小姐 (151) 赵 (133) 很 (114) 你 (111) 西菜饷
4	0.114	我 (652) 在 (625) 店 (520) 餐厅 (328) 这里 (280) 南翔 (269) 之 (253) 馒头
5	0.002	的 (459) 我 (285) 了 (195) 你 (119) 个 (114) 来 (107) 元 (103) 1 (102) 分
6	0.002	我 (1072) 在 (916) 这里 (424) of (301) house (296) flour (280) 毅屋 (273
7	0.113	的 (223) 三 (192) 在 (145) 牛蛙 (144) 顾 (122) 天堂 (107) 椰香 (100) 馆
8	0.005	我 (1377) 在 (1117) 了 (885) 吃 (813) 的 (519) 这里 (500) 店 (423) 一 (3
9	0.001	我 (321) 在 (291) 料理 (165) 日本 (119) i (118) m (115) 虹口店 (114) 唐宫
0	0.139	人间 (190) 萤七 (112) 餐厅 (98) 烤肉 (95) 希望 (89) people (89) 很 (85)
1	0.002	红 (191) 房子 (185) 西菜馆 (161) 虹口店 (114) 唐宫膳 (97) 西 (72) 皇朝 (
2	0.002	不 (65) mo (65) 3 (64) 5 (55) 1 (49) 2 (47) 款 (46) 信 (45) 都 (45) 中 (
3	0.354	吃 (1111) 不 (468) 很 (460) 家 (423) 好 (365) 好吃 (341) 还 (332) 广场 (
4	0.141	店 (747) 吃 (301) 小 (278) 广场 (251) 南翔 (233) 馒头 (216) 栗子 (208) 女
5	0.345	三 (162) 馆 (133) 牛蛙 (122) 顾 (108) 1 (99) 元 (93) 生 (91) 煎 (90) 飞龙
6	0.006	不 (890) 人 (413) 都 (371) 会 (330) 好 (239) 想 (191) 还 (185) 说 (183)
7	0.004	house (302) 店 (285) flour (280) 毅屋 (277) 火锅 (203) 茶 (194) 吃 (157)
8	0.003	餐厅 (193) 分享 (191) 料理 (167) 耶里夏丽 (153) 西域 (141) 日本 (123) 天
9	0.004	餐厅 (679) 上海 (226) 吃 (172) 舞台 (169) 威斯汀 (169) 新 (153) 店 (147)

Fig. 11. Performance with Topic Number

Figure 11 shows different word list of topics, with regard to stop words removal or not. The upper part of the word list is learned without stop word removal and the bottom part is learned with stop word removal.

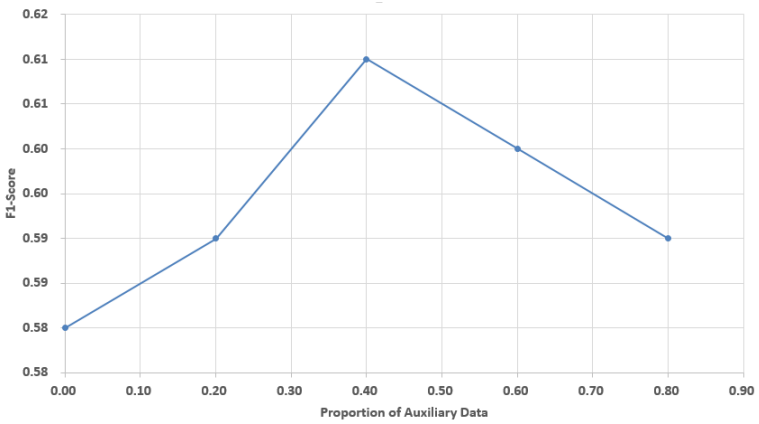

Fig. 12. Performance with Auxiliary Data Proportion

Moreover, we find that all the training data set is about restaurants. Intuitively, we want to model different abstract topics over more general microblogs. So that we add some general microblog data along the timeline. The size of the auxiliary data is proportional to the training data set. And the proportional ratio is set as a parameter r, $0 \leq r \leq 1$. Figure 12 shows the performance of different size of auxiliary data set. We may find that more auxiliary data (until 0.4 with regard to training data set) makes better performance, though the improvement is not large. And adding more auxiliary data (more than 0.4) has reverse influence.

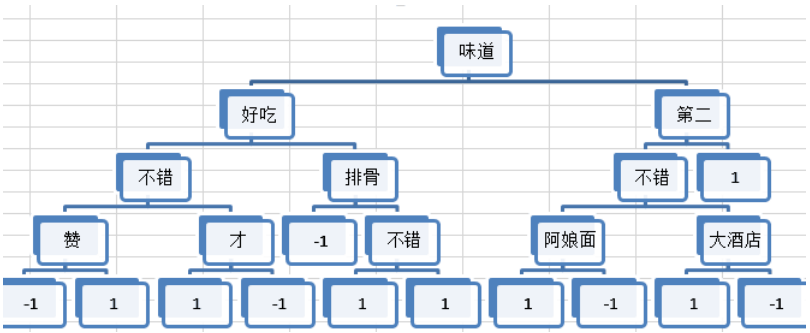

Fig. 13. Decision Tree

6.6 Decision Tree

Figure 13 shows the decision tree we obtained. The left side direction of each decision is negative, otherwise on the right side. Label "-1" means the model classify the data as non a review while "1" as true review.

6.7 Overall

Overall, all the performance of our models are listed in Table 2. Here we can see that all of our methods preform better compared with the baseline. And also, different methods achieve similar performance considering the F1-score. Naive Bayes achieves balance performance between the precision score and the recall score, Mutual Information and Topic Model with SVM achieve higher recall scores, while SVM using Mutual Information and Decision Tree achieve higher precision scores.

Table 2. Overall Performance

Method	Precision	Recall	F1-Score
Baseline	0.30	1.0	0.46
Naive Bayes	0.63	0.63	0.63
Mutual Information	0.55	0.81	0.65
SVM using Mutual Information	0.76	0.55	0.64
Decision Tree	0.83	0.41	0.55
Topic Model with SVM	0.59	0.73	0.66

7 Discussion

In our work, we design several methods to extract features from the natural language text. And meanwhile, we try different models combined with our selected features to evaluate our methods. The final results show that all of the methods make sense. That is to say, the feature selection method works well.

Our method achieves a F1-score of 0.65. One of the reasons may be that the training data set is not enough, we have less than ten thousand texts in total. We need to adapt the training data to calculate the word frequences and document frequences, which depends closely on the training document corpus. So when the data is bigger, our methods may work better. In the other hand, Since all of the different models we use achieve similar performance, the work we need to do in the future should also focus on the work of feature selection. We will try more feature selection methods about text classification in the future.

References

1. Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: ACM SIGMOD Record. vol. 22, pp. 207–216. ACM (1993)
2. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. the Journal of machine Learning research 3, 993–1022 (2003)
3. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the fifth annual workshop on Computational learning theory. pp. 144–152. ACM (1992)
4. Church, K.W., Hanks, P.: Word association norms, mutual information, and lexicography. Computational linguistics 16(1), 22–29 (1990)
5. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the royal statistical society. Series B (methodological) pp. 1–38 (1977)
6. Koenigstein, N., Dror, G., Koren, Y.: Yahoo! music recommendations: modeling music ratings with temporal dynamics and item taxonomy. In: Proceedings of the fifth ACM conference on Recommender systems. pp. 165–172. ACM (2011)
7. Quinlan, J.R.: C4. 5: programs for machine learning. Elsevier (2014)
8. Tsaparas, P., Ntoulas, A., Terzi, E.: Selecting a comprehensive set of reviews. In: Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 168–176. ACM (2011)
9. Xu, Y., Jones, G.J., Li, J., Wang, B., Sun, C.: A study on mutual information-based feature selection for text categorization. Journal of Computational Information Systems 3(3), 1007–1012 (2007)