Sentiment Analysis on Movie Reviews

Introduction

Sentiment Analysis, the process defined as "aims to determine the attitude of a speaker or a writer with respect to some topic" in Wikipedia, has recently become an active research topic, partly due to its potential use in a wide spectrum of applications ranging from "American idol" popularity analysis to product user satisfaction analysis. With the rapid growth of online information, a large amount of data is at our fingertips for this kind of analysis. However, the sheer volume of information was a daunting challenge itself. To separate relevant information from the irrelevant, and to gain knowledge from this unprecedented deluge of data, automatic algorithm is essential. In this project, we explored the use of various supervised machine learning algorithms in learning sentiment classifier and tested the effectiveness of different feature selection algorithms in improving those classifiers.

Input Data

The input data for this project originated from the Rotten Tomatoes dataset: a corpus of movie reviews originally collected by Pang and Lee [1] for sentiment analysis. According to the information on Kaggle.com where we downloaded the data, Socher et al. [2] used Amazon's Mechanical Turk to create fine-grained labels for all parsed phrases in the corpus during their work on sentiment treebanks.

One pre-labeled training dataset and one unlabeled testing dataset are included. The sentiment labels use integers from 0 to 4, with 0 being the most negative and 4 the most positive. The training dataset contains 156,060 data instances and the testing dataset 66,292. Each data instance comprises attributes: "phrase id", "sentence id", "phrase". The "phrase" attribute contains the actual content of the represented text phrase, which were transformed into numeric vectors usable by machine learning algorithm using the feature extraction process discussed below.

Feature Extraction

For text analysis, one standard feature extraction approach is to represent text phrases as n-grams, i.e. subsequences of n words with or without skips. To account for the fact that a word preceded by a negative word such as "barely" has opposite sentiments, we experimented with both unigram and bigram features in this project; the feature vectors thus produced span a very high dimensional feature space and hence is expected to be very sparse. The special nature may cause deterministic machine learning algorithms to "overfit". This issue will be revisited later in the feature selection section.

Methodology

Naïve Bayes algorithm, Softmax (Logistic) Regression and Support Vector Machine are all known to be suitable for text classification but varies in its assumption and formulation of the problem. All were explored in this project.

Naïve Bayes Classifier (NBC)

As a generative learning algorithm, the Naïve Bayes algorithm models class prior p(y) and conditional feature prior p(x|y) from training samples. NB Classifiers learned in this project is based on the multivariate Multinomial event assumption.

Assumption:

The multivariate Multinomial event assumes that: given a sentiment of a phrase or data instance i, word in one position in the phrase tells us nothing about words in other positions; word appearance does not depend on position.

Formulas for Prior Estimates:

The p(x|y) for a feature k is modeled as: $\emptyset_{k|y=c} = \frac{\sum_{i=1}^{m}\sum_{j=1}^{n_i} 1\{x_j^{(i)}=k \cap y^{(i)}=c\}+1}{\sum_{i=1}^{m} 1\{y^{(i)}=c\}n_i+|V|}$. Here, n_i denoting the total frequency of all features appeared in a data instance i; $x_j^{(i)}$: the word or more precisely the index of the word at position j of data instance i; k: index for the k^{th} word in the vocabulary dictionary; $c \in 0,1,2,3,4\}$;V, $|V|$: vocabulary and its size; m: number of data instances; $y^{(i)}$: class label for instance i.

The Laplace smoothing is applied to guarantee that the basic probability axioms: probability being nonnegative and sum to 1, will still hold in the case when a data instance contains an n-gram not found in the dictionary and the estimate without smoothing will otherwise result zero for all sentiment classes. The estimate without the Laplace smoothing is basically the fraction of times a word *k* appears across all data instances of label c.

The prior p(y) can be modeled as the fraction of data instances with sentiment label c: $\emptyset_{y=c} = \frac{\sum_{i=1}^{m} 1\{y^{(i)}=c\}}{m}$.

With the priors thus modeled, the Bayes rule is used to derive the posterior distribution on y given x and the class with highest posterior probability is picked as the prediction of the sentiment class. Give a data instance i vectorized as $x^{(i)} = \{x_1, x_2, \dots, x_n\}$, the mathematical formulation of the predication is as follows:

$$\hat{y} = \underset{c\in\{0,1,2,3,4\}}{\operatorname{argmax}}\, p(y=c|x_1,x_2,\dots,x_n) = \underset{c}{\operatorname{argmax}} \frac{p(x_1,x_2,\dots,x_n|y=c_j)p(y=c)}{p(x_1,x_2,\dots,x_n)}$$
$$= \underset{c}{\operatorname{argmax}}\, p(x_1,x_2,\dots,x_n|y=c)p(y=c)$$

Softmax (by Stochastic Gradient Descent)

Softmax algorithm is another algorithm other than Naïve Bayes, which we implemented using Matlab scripting language for this project. Unlike Naive Bayes, it belongs to the deterministic algorithm family and aims to learn mappings (using a set of parameters such as θ below) directly from the input space X to labels y.

The model assumes $p(y=c|x;\theta) = \frac{e^{\theta_c^T x}}{\sum_{c\prime\in\{0,1,2,3,4\}} e^{\theta_{c\prime}^T x}}$ and $h_c^{(i)}(\hat{\theta}) = \frac{e^{\hat{\theta}_c^T x^{(i)}}}{\sum_{c\prime\in\{0,1,2,3,4\}} e^{\hat{\theta}_{c\prime}^T x^{(i)}}}$ is its estimate.

The Softmax problem of this project is formulated as:

$$\hat{\theta} = \underset{\hat{\theta}}{\operatorname{argmax}}\, l(\theta) = \underset{\hat{\theta}}{\operatorname{argmax}} log \prod_{i=1}^{m} p(y^{(i)}|x^{(i)};\theta) = \underset{\hat{\theta}}{\operatorname{argmax}} \sum_{i=1}^{m} log \prod_{c\in\{0,1,2,3,4\}} \left(\frac{e^{\theta_c^T x^{(i)}}}{\sum_{c\prime\in\{0,1,2,3,4\}} e^{\theta_{c\prime}^T x^{(i)}}}\right)^{1\{y^{(i)}=c\}}$$

The partial gradient of $l(\theta)$ with respect to $\theta_{c,k}$ can be derived as (due to page limit, no detailed steps are included):

$$\frac{\partial}{\partial\theta_{c,k}} l(\theta) = \left(\mathbf{1}\{\boldsymbol{y^{(i)}} = \boldsymbol{c}\} - h_c^{(i)}(\hat{\theta})\right)\boldsymbol{x_k^{(i)}}$$

With $\theta_{c,k}$ being the parameter for class c and word k; $x_k^{(i)}$: the frequency of word k in data instance $\boldsymbol{i}$;
$x^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \dots, x_{|V|}^{(i)}\}$: vector representing how frequent each word in V appeared in data instance $\boldsymbol{i}$;
Considering that the whole data instances from big data often can't be fit in the memory, stochastic gradient descent algorithm is implemented to solve the parameters for Softmax problem.

Initialize θ_c to vector of zeros for each label class $c \in \{0,1,2,3,4\}$
Loop through each data instance i { for i=1 to m, {
$\theta_{c,k} := \theta_{c,k} + \alpha\left(\mathbf{1}\{\boldsymbol{y^{(i)}} = \boldsymbol{c}\} - h_c^{(i)}(\theta)\right)\boldsymbol{x_k^{(i)}}$ (for each feature k and each label class c)
}}

Support Vector Machine

Support Vector machine is another machine algorithm fit for text classification problem for reasons elaborated by Joachims [3]. The implementation of SVM in this project leveraged a third part library: liblinear [4]. The problem is formulated using the standard linear SVM formulation with $l1$ regularization.

$$[w, b, \xi] = \underset{w,b,\xi}{\operatorname{argmin}} \frac{1}{2} w^T w + C\sum_{i=1}^{m} \xi_i$$

s.t.
$$y_i(w^T x_i + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0, i = 1,2\dots,m$$

Feature Selection

Realizing most words in a phrase do not convey sentiments, two feature ranking algorithms are implemented to automatically identify the corresponding "uninformative" features.

Mutual Information (MI) Rank

Mutual Information (MI) index is the feature ranking method related to Naive Bayes algorithm. High MI indicates that a feature xi is very "informative" about y. In contrast, a very low MI indicates "non-informative" features.

$$MI(x,y) = \sum_{x}\sum_{y} p(x,y) * log\frac{p(x,y)}{p(x)p(y)}$$

MI calculated in two different ways are used in this project. One uses the priors modeled in NBC to calculate MI(k,y) for each word k as:

$$\sum_{c\in\{0,1,2,3,4\}} \emptyset_{k|y=c} * \emptyset_{y=c} log\frac{\emptyset_{k|y=c}}{p(x_j^i = k\ \ \forall i,j)} + (1-\emptyset_{k|y=c})\emptyset_{y=c} log\frac{(1-\emptyset_{k|y=c})}{1-p(x_j^i = k\ \ \forall i,j)}$$

Here, $p(x_j^i = k\ \ \forall i,j) = \sum_{c\in\{0,1,2,3,4\}} \emptyset_{k|y=c} * \emptyset_{y=c}$ is the probability of word k appears in phrase i at position j, which based on NB assumption is the same for all data instance *i* and word position j.

The other is to use the formula given in the chapter 13 of Manning's book [7]. The same formula as given above can be used by changing the definition of conditional prior of word k into the following:

$\emptyset_{k|y=c} = \frac{\sum_{i=1}^{m} 1\{k\ appears\ in\ i \cap y^{(i)}=c\}}{\sum_{i=1}^{m} 1\{y^{(i)}=c\}}$ (the p(x|y) modeled when NB used Multivariate Bernoulli Event assumption).

F-score Rank

The F-score is the correlation coefficient between one of the features and the label. It is very similar to F-test statistics that measures the difference between variance between different population groups and among each population group. Intuitively, it is a good measure of how "discriminative" a feature is regarding sentiment classes. The formula of F-score used in this project is a variant of the one used in Chang [5], which instead is for two classes.

$$F(k) = \frac{\sum_c (\bar{x}_k^{(c)} - \bar{x}_k)^2}{\sum_c \frac{1}{\sum_{i=1}^{m} 1\{y^{(i)}==c\}} \sum_{i=1}^{m} \left(\left(x_k^{(i)} - \bar{x}_k^{(c)}\right)^2 * 1\{y^{(i)}==c\}\right)}, \quad \text{with } \bar{x}_k^{(c)} = \frac{\sum_{i=1}^{m} x_k^{(i)} * 1\{y^{(i)}==c\}}{1\{y^{(i)}==c\}} \text{ and } \bar{x}_k = \frac{\sum_{i=1}^{m} x_k^{(i)}\}}{m}$$

Results, Discussion and Conclusion

The models trained using algorithms discussed above are evaluated on both the training data and the test data set. The prediction for the test data set is scored online by kaggle.com. In the following, some interesting results are presented and interpreted.

Table 1: Comparison of Prediction Accuracy

	Unigram Features			Bigram Features		
	NBC	**SoftMax**	**SVM**	**NBC**	**SoftMax**	**SVM**
Training	67.20%	88.18%	66.40%	69.20%	93.58%	91.00%
Test	60.01%	40.95%	47.74%	59.20%	59.20%	59.20%

Table 1 compares the use of unigram vs bigram and the use of three aforementioned algorithms. On the training data, SoftMax has the best performance and achieves almost perfect predicting accuracy whether using unigram or bigram features; SVM achieved almost perfect prediction accuracy when using bigram. However, NBC consistently beat the other two algorithms on the test data in terms of prediction accuracy. The phenomena can be explained by Vaponic

Chervonenkis (VC) dimension analysis. The hypothesis function for Softmax use |V|*|C| (|V|: number features; |C|: number of classes) number of parameters; while the hypothesis for SVM uses |V|*(|C|-1) number of parameters, since each set of parameters that defines a hyper-plane to separate one class from the rest uses has |V| dimension. More parameters for linear or generalized linear models like Softmax (the decision boundary for Softmax is essential linear.) and SVM translate into a hypothesis space with higher VC dimension, which itself translates into higher upper bound on the discrepancy between the hypothesis error and the true error, or "overfitting" on testing data. This is one cause which contributes to better performance of Softmax over SVM on training data and overfitting of both on test data. On the other hand, the higher dimensional space resulted from Bigram renders features vectors sparser and hence higher chances to be separable, which contributed to the improved prediction accuracy of SVM on the training data in the case with Bigram in comparison to the Unigram case. Naïve Bayes, on the other hand, is a generative model, which is not as susceptible to the curse of VC dimension or "overfitting". Instead, according to central limit theorem, the empirical probability based NBC tends to become quite accurate with a large number of data instances. However, the more rigorous assumption of NBC also makes it less accurate in modeling the training data.

Figure 1: Prediction Accuracy of NBC and SVM vs. Number of Features

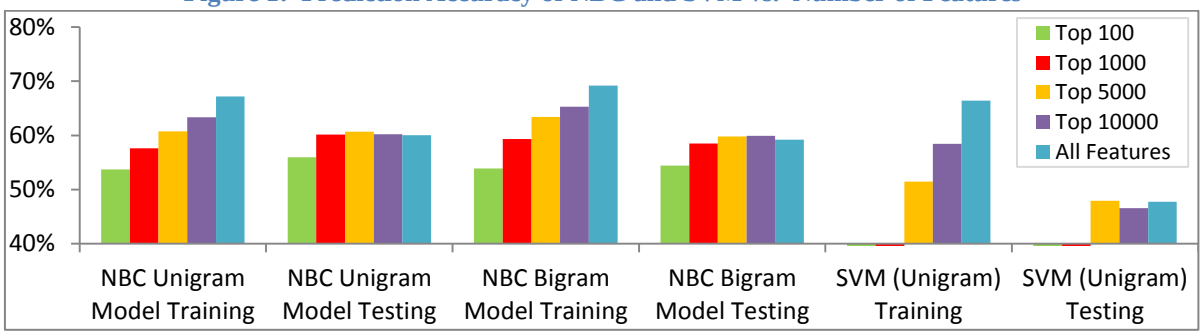

Figure 1 illustrates the effectiveness of feature selection by MI index for NBC and by F-score for SVM. It shows that the top 1000 features turn out to do an almost as good job as 10,000 features. In addition, the prediction accuracy doesn't improve proportionally with the increase in features included in the model. The larger the number of features already included the smaller the improvement in accuracy by adding more features. On the test data, the model sometimes performs better after irrelevant features had already been tossed out but still having enough relevant features kept.

Table 1: Most Informative vs Least Informative by Feature Rank

NBC Features Ranked by MI		SVM Features Ranked by F-score	
5 most informative	5 most informative	5 most informative	5 least Informative
'escapades'	'infantilized'	'flopped'	'henry'
'adage'	'unhappiness'	'repugnant'	'tearing'
'goose'	'glosses'	'execrable'	'pedigree'
'gander'	'hunky'	'under-inspired'	'harrison'
'amuses'	'relish'	'not-at-all-good'	'buddy'

Note 1: The MI shown here is calculated using the formula given in the chapter 13 of Manning's book [7].

Top 5 SVM tokens in order based on F rank turned out to be quite interesting. The top informative token "flopped" unambiguously conveys negative feeling. While the top least informative token "henry" is a name; names usually don't convey any feeling unless it happens to be something like "Voldemort". On the contrary, the result of feature ranked by MI doesn't seem to be as intuitive. After some investigation, we found out that the high MI rank for sentiment neutral words like "goose" and "adage" is due to the fact that these words seem to concentrate in the phrases with class label 2. Hence, MI algorithm identified those as very informative about class label 2, which happens to represent neutral sentiment and also constitutes the largest proportion of the document. This inspired the thought that maybe a modified MI ranking which identifies features most informative about the four more extreme sentiments could have served a better job. Due to time limitation, the idea is not pursued further in this project.

Figure 2: Convergence of SoftMax Parameters with Passes of Data with learning rate=0.5.

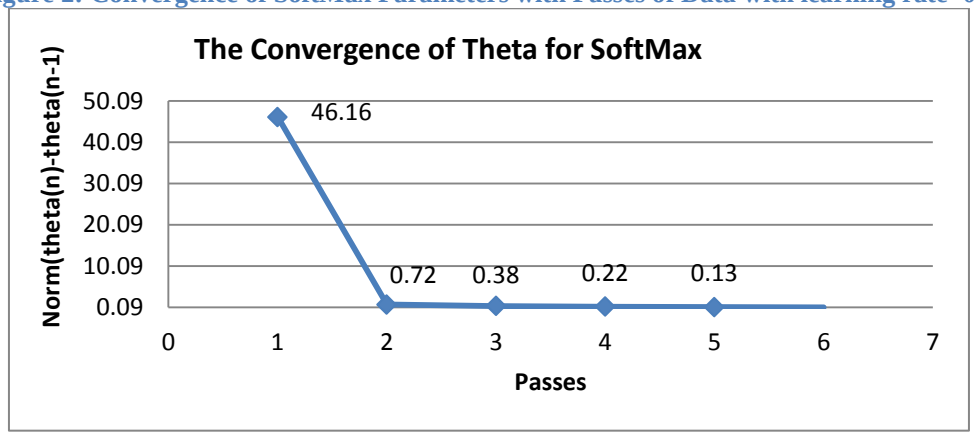

Figure 2 shows the convergence rate of SoftMax parameters with the number of passes through the training data using 0.5 as the learning rate and with 10,000 top features selected using F-score. Though the algorithm converges quickly, due to the large number of data instances, each pass takes a long time to finish. In this aspect, the SoftMax is the least efficient among the three algorithms tested.

The feature selection using F-score turns out to work well for SoftMax algorithm. The model with the top 10,000 unigram features achieves a 44.85% accuracy rate on the test data which is better than the accuracy rate 40.95% shown in table 1 for Softmax Model using all features. The model with the top 5000 features resulted 40% accuracy rate on the test data, which is almost the same as the model using all features.

In conclusion, through this project, we learned that the deterministic algorithms like SoftMax and SVM is liable to over-fitting in a high dimensional feature space. On the other hand, with its relative rigorous assumption of conditional independence of features, Naïve Bayes algorithm doesn't perform too well on the training data but also has less problem of over-fitting. Furthermore, feature selection methods can improve performance and the top features identified by different feature selection methods could differ.

Future work:

In the future, other feature selection methods or hybrid of feature selection methods will be explored, for instance combining the top features from the MI ranking with those from F-score ranking. In addition, More sophisticated text analysis algorithms, such as MaxEm or deep learning/neural network or others will be implemented.

Reference:

[1] Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In ACL, pages 115–124.
[2] Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank, Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Chris Manning, Andrew Ng and Chris Potts. Conference on Empirical Methods in Natural Language Processing (EMNLP 2013).
[3] Thorsten Joachims, "Text Categorization with Support Vector Machines: Learning with Many Relevant Features", http://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf
[4] http://www.csie.ntu.edu.tw/~cjlin/papers/libsvm.pdf
[5] Yin-Wen Chang and Chih-Jen Lin, "Feature Ranking using linear SVM" http://www.csie.ntu.edu.tw/~cjlin/papers/causality.pdf
[6] Andrew Ng, Stanford CS229 Course Notes, http://cs229.stanford.edu/materials.html
[7] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008.Introduction to Information Retrieval. Cambridge University Press.