Sentiment Classification and Opinion Mining on Airline Reviews

Peng Yuan (pengy@stanford.edu) Yangxin Zhong (yangxin@stanford.edu) Jian Huang(jhuang33@stanford.edu)

1 Introduction

As twitter gains great popularity as an online social networking service in recent years, tweets on twitter become a valuable source of information for companies to get feedbacks from their customers. However, extracting such information from seemingly random comments and reviews is not an easy task due to the scale of data size and the difficulty in semantic analysis. For example, someone tweets: "*@VirginAmerica Hey first time flyer next week – excited! But I'm having a hard time getting my flights added to my Elevate account. Help?*" Is she happy with the flight experience or not? If not, what makes her unhappy? Having machines answer these types of questions and learn the sentiment of each tweet automatically will be extremely helpful for service providers to understand the strength and weakness of their products and to improve them in the future.

In this paper, we focus on reviews on twitter for major U.S. airlines and try to extract sentiment and opinions from these reviews. We explored four learning techniques: Naïve Bayes, Support Vector Machines (SVM), lexicon-based methods[1] and Convolutional Neural Networks (CNN), to predict the sentiment of these tweets – positive, neutral or negative (referred as *sentiment task*). In addition, for negative reviews, we also applied the same techniques to classify the reasons of bad feedbacks to see whether it is due to late flight, cancelled flight, flight booking problems, or customer service issue, etc. (referred as *negative reason task*). We use N-gram features as well as work2vec features[2] as input to our algorithms.

2 Related Work

With the rapid growth of social networks and online discussion forum, the web has been rich in user-generated free-text data, where users can express various attitudes towards products, which have been attracting researchers into sentiment analysis[3, 4]. Sentiment analysis plays an important role in many applications, including opinion retrieval[5], opinion oriented document summarization[6], and word-of-mouth tracking[7].

To determine whether a document or a sentence expresses a positive or negative sentiment, two main approaches are commonly used: the lexicon-based approach and the machine learning-based approach. The lexicon-based approach involves calculating orientation for a document from the semantic orientation of words or phrases in the document[8]. The machine learning-based approach involves building classifiers from labeled instances of texts or sentences[9] and typically trains sentiment classifiers using features such as unigrams or bigrams. Most techniques use some form of supervised learning by applying different learning techniques such as Naive Bayes, Maximum Entropy and Support Vector Machines. In our project, we explored both lexicon-based approach and machine learning-based approach and compared their performance.

3 Dataset and Features

3.1 Dataset

Our data is available online[10]. It has 14640 valid tweets from 2/17/2015 to 2/24/2015 related to reviews of major U.S. airlines, containing *sentiment label, negative reason label*, *tweets content* and other meta information like *location*, *user ID* etc. The data fraction is roughly 15% positive, 65% negative, and 20% neutral. We split the data into 80% as the training set and 20% for testing, and we use 5-fold cross-validation to choose hyper-parameters. Since our training and testing only based on *tweets content,* in later sections we refer *tweets content* as "data". We also assume that each tweet has only one label negative reason label.

What's more, *sentiment label* is one of positive, neutral or negative; and *negative reason label* is one of the 11 classes: not negative (not a real negative reason, serves as a placeholder), can't tell (negative tweets but not indicating any reason), bad flight, late flight, customer service issue, booking problem, lost luggage, flight attendant complaints, cancelled flight, damaged luggage, long lines.

3.2 N-gram Features with Preprocessing

N-gram feature uses concatenated words as feature in the hope that it can capture the information between words and phrases. For example, a sentence "This is a sunny day" extracted by 2-gram features would be sparse vector as ["this is", "is a", "a sunny", "sunny day"].

In addition, to unify emoji and other special words (referred as tokens in later sections), we preprocess the training and test data by replace these emoji or special words into unified tokens as indicated in Table 1.

Then, tokens appear too frequent like word "the" or too seldom is unlikely to contain useful information, thus we remove all tokens that appears too frequent of too seldom. The threshold is searched via grid searching using 5-fold cross validation, more details in Section 4.1 and Section 4.2.

Table 1 Token replacement

From	To
Links, urls	"theurladdresstoken"
Telephone numbers	"thephonenumbertoken"
Number stars with '$' symbol	"themoneytoken"
Other numbers	"othernumbertoken"
;) :) (space)xD ;D :D (space)XD ;-D :-D ;-) :-)	"thehappyfacetoken"
:(:-((space)TT(space)	"theunhappyfacetoken"

3.3 Word2vec Features

Word2vec is a state-of-the-art method proposed by Tomas Mikolov[2] for word representation learning, using Skip-Gram and continuous bag of words (CBOW) models. In word2vec, a word is represented by a real-valued vector of fixed low dimension (usually by several hundreds). It is empirically shown to preserve linguistic regularities and semantic relationships between words. For example, V(king) – V(queen) ≈ V(man) – V(woman), where V denotes "vector of". It is also believed that words that are highly correlated are likely to be closer to each other in the vector space compared with others that co-occur less frequently.

4 Methods

4.1 Naïve Bayes

This method is based on probabilistic inference rules. Given $x = (x_1, x_2, ...x_n)$ be some n features of some test sample, and we want to label it to class with $\max P[C_k \mid x_1, ..., x_n]$, the probability of the data being for each possible class C_k. In our case, x_i is each N-gram feature and C_k is the target class. Using Bayes' theorem, the conditional probability is rewritten as $P[C_k \mid x_1, ..., x_n] \propto P[C_k]P[x_1 \mid C_k]...P[x_n \mid C_k]$, where $P[C_k]$ is referred as prior and $P[x_1 \mid C_k]...P[x_n \mid C_k]$ is referred as likelihood. The training part is to find $P[x_n \mid C_k]$ on the dataset by counting, and the prediction is by labeling one sample to class y by

$$y = \underset{k \in \{1,...,K\}}{\arg\max}\, P[C_k]P[x_1 \mid C_k]...P[x_n \mid C_k]$$

4.2 Support Vector Machines (SVM)

SVM is based on finding the maximum margin between different classes by determining w and b in $w \cdot x + b = 0$, the hyperplane separating classes. Mathematically, binary-class SVM minimizes

$$\frac{1}{n} C \sum_{i=1}^{n} max(0, 1 - y^{(i)}(w \cdot x^{(i)} + b)) + \lambda \| w \|^2$$

where $y^{(i)} \in \{-1, 1\}$ is the true label, λ is the regularization term, C is the penalty parameter of error term, and $max(0, 1 - y^{(i)}(w \cdot x^{(i)} + b))$ is the loss of miss classification, also known as hinge loss. Note that there are many versions of SVM, what we use is a linear SVM with hinge loss due to computational efficiency. The training is done by finding w and b via coordinate descent and the prediction is done be $y = \mathrm{sgn}(w \cdot x + b)$.

4.3 Lexicon-based Method

Lexicon-based methods are based on a pre-trained word (lexicon) list which can tell whether a word has a positive or negative sentiment in general and to what extent. For each text, it will check every word and aggregate the word sentiment score into the sentence/paragraph sentiment score. A simple aggregating way is to average the word scores: if the average score is larger than a threshold α , then the sentence will be predicted as positive; if the score is less than another threshold β , it will be predicted as negative; otherwise it will be considered as neutral. Advanced lexicon-based methods also involve other factors (e.g. negation, stress) in the aggregation formula.

4.4 Convolutional Neural Networks (CNN)

Convolution neural network (CNN) is a combination of neural network and convolution operator, where neural network can learn the non-linear relationship between explicit and hidden features and convolution operator can capture the local region information in features. Kim [11] proposed a CNN architecture with two channels and multiple filter sizes for general text classification tasks. Our CNN are based on his architecture, which has only one channel but with an extra hidden layer. The structure of our neural network is shown in Figure 1.

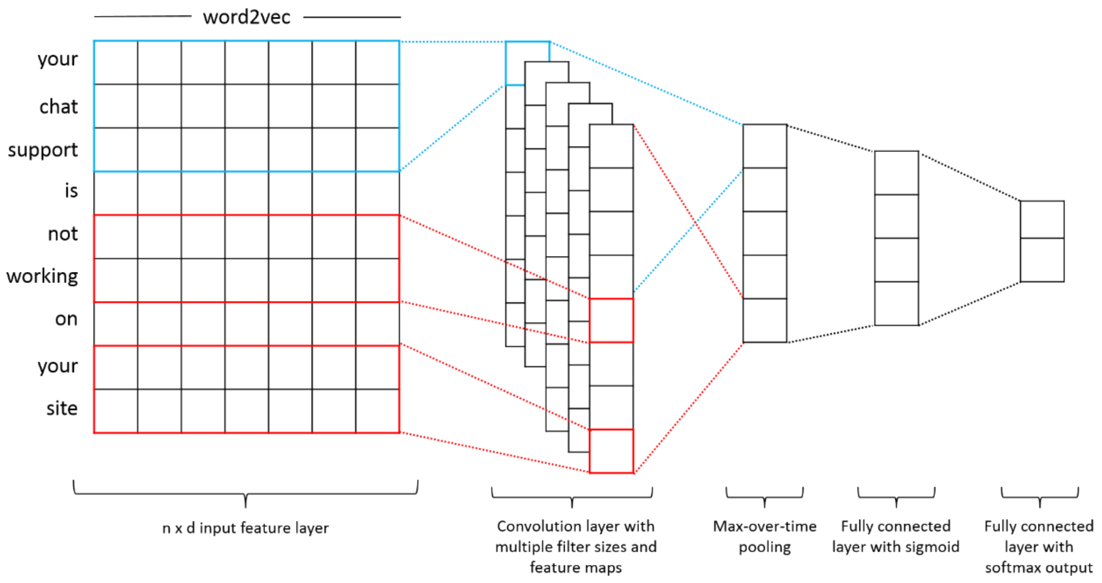
word2vec
your
chat
support
is
not
working
on
your
site
n x d input feature layer
Convolution layer with multiple filter sizes and feature maps
Max-over-time pooling
Fully connected layer with sigmoid
Fully connected layer with softmax output

Figure 1 CNN structure in detail

Our network has five layers in total. The first layer is the input feature layer. For each sentence or paragraph, the rows in its feature matrix correspond to each of its word in the same order. More concretely, each row is a feature vector (e.g. word2vec feature) of the corresponding word. The second layer is convolution layer. We applied convolution on the input layer using multiple filters with different sizes. Note that the width of each filter must be the same as length of word2vec feature. This is because different dimensions of word2vec feature have no correlation to each other and thus don't contain "local region feature" (apply convolution among them will be useless).

The third layer is a max-over-time pooling layer. Unlike pooling layer in computer vision, this layer will not take maximum values over local regions of feature maps from previous layer; instead, it takes the maximum value over each entire feature map (max-over-time) to form a more compact feature map. The idea is to capture the most important feature of each feature map while dealing with variable length of text data[11]. The fourth layer is a fully connected hidden layer with sigmoid function. And the fifth layer is a fully connected layer with softmax output.

The idea behind this network is to use convolution layer to capture the local feature among words (similar to N-gram features), use two fully connected layers to learn high-level features appropriate for specific classification tasks, and use pooling layer to prevent overfitting and deal with different lengths of text data.

5 Experiments, Results and Discussion

5.1 Naïve Bayes and SVM

We use N-gram features for Naïve Bayes method and as described in Section 3.2, we remove tokens that appear too frequent or too seldom, the threshold is chosen by grid searching on 5-fold cross validation. The searched hyper-parameters' space is shown in Table 2, where *min-df* is the minimal occurrence of a token in the training dataset; *max-df* is the maximal occurrence rate of a token in the training dataset; *N* is the N in the N-gram, the number of adjacent words we took into account when extracting each feature.

As for SVM, we use a linear SVM with hinge loss and turn the multi-classification into binary classification by one-versus-all method, that is for each class train a model that separates it from the rest. As for features and hyper-parameters for SVM, we still use the N-gram features and using 5-fold cross validation to search hyper-parameters in Table 2, plus a $C \in \{1,10,100,500,750,1000,1250,1500,2000\}$.

For the sentiment task, gird search on hyper-parameters gives the best combination as min-df=3, max-df=0.9, N=1 for Naïve Bayes and min-df=1, max-df=0.95, N=(1, 2) - meaning the combined form of 1-gram and 2-gram, and C=1250 for SVM. The detailed performance is in Table 3.

Table 2 Hyper-parameters search space

Parameter	Range
min-df	0, 1, 2, 3, 5, 7
max-df	0.8, 0.9, 0.95, 0.97
N	1, 2, 3

Table 3 Naïve Bayes and SVM performance for sentiment task

	Naïve Bayes			Linear SVM		
Overall Accuracy	0.712			0.796		
Category	Precision	Recall	F1-measure	Precision	Recall	F1-measure
Positive	0.916	0.328	0.482	0.812	0.678	0.739
Negative	0.697	0.988	0.818	0.831	0.918	0.873
Neutral	0.718	0.246	0.367	0.667	0.559	0.608

For the negative reason task, gird search on hyper-parameters gives the best combination as min-df=7, max-df=0.8, N=(1, 2, 3) - meaning the combined form of 1-gram, 2-gram and 3-gram, for Naïve Bayes; and min-df=0, max-df=0.97, N=(1, 2, 3), C=1500 for SVM. The detailed performance is in Table 4, where N/A in some rows mean that classifier failed to predict any example as the corresponding classes.

Table 4 Naïve Bayes and SVM performance for negative reason task

	Naïve Bayes			SVM		
Overall Accuracy	0.576			0.648		
Category	Recall	Precision	F1-measure	Recall	Precision	F1-measure
not negative	0.566	0.941	0.707	0.776	0.810	0.793
can't tell	0.667	0.069	0.125	0.355	0.238	0.285
bad flight	1.000	0.008	0.015	0.521	0.285	0.368
late flight	0.676	0.461	0.548	0.588	0.694	0.637
customer service issue	0.553	0.670	0.606	0.582	0.684	0.629
booking problem	N/A	0.000	N/A	0.419	0.255	0.317
lost luggage	0.842	0.122	0.213	0.628	0.710	0.667
flight attendant complaints	N/A	0.000	N/A	0.551	0.278	0.370
cancelled flight	0.789	0.302	0.437	0.596	0.711	0.648
damaged luggage	N/A	0.000	N/A	1.000	0.100	0.182
long lines	N/A	0.000	N/A	0.300	0.094	0.143

Naïve Bayes is more like a baseline model for us, it performs worst as expected, however it runs much faster than SVM and CNN, making it useful when dataset grows even larger. After the grid search, we also noticed that hyper-parameters can influence the performance a lot, it improves SVM's performance by near 8 percent after the grid search. SVM performs well in both tasks as expected, since the data is somewhat separable by support vectors/words like "good" or "bad" for sentiment task and "late", "expensive" for negative reason task.

5.2 Lexicon-based Method

Lexicon-based method can only be applied on sentiment task, not the negative reason task, due to the public database is trained only on the sentiment data. In detail, we use three different trained word lists[12-14], and weight the score of each list to get a final sentiment score of 3293 positive and 6493 negative words, which cover 15% words of our corpus. Our negation rules include: 1) Negating from negation word (e.g. "no", "never", "n't", "not", "little") till the end of sentence; 2) Negating from negation word (e.g. "although", "though", "even if") till next punctuation; and 3) Negating from the beginning till the negation word (e.g. "but", "however"). We use averaging as the aggregation function. We did a grid search to find the most appropriate threshold parameters, and the final values we used are: $\alpha = 0.3,\ \beta = -0.7$.

Since lexicon-based method doesn't need training, we test the algorithm on the whole dataset and get the following result in Table 5.

Table 5 Lexicon-based method performance for sentiment task

Category	Recall	Precision	F1-measure	Overall Accuracy
Positive	0.650	0.475	0.549	
Negative	0.800	0.744	0.769	0.652
Neutral	0.222	0.435	0.294	

Compared to supervised learning method, lexicon-based method has a poor overall performance. We also found that this method is not good at classifying neutral reviews. That' s because only when the review get a near zero average score will it be considered as neutral, which usually occur when only few words of the review are covered by word list. In lexicon-based methods, the low coverage rate of word list can cause problem in prediction. And in the airline reviews corpus, many positive reviews are misclassified because these reviews also give some negative feedbacks/suggestions while thumbing up, which disturb the prediction a lot. Even when we use a low positive threshold, where $\alpha = 0.3$, the performance on positive reviews are not so good; and if we use an even lower threshold α, the performance on neutral reviews will go down instead. This is the trade-off in lexicon-based method.

5.3 Convolutional Neural Networks (CNN)

We trained word2vec features of 100 dimensions over 500,000 unlabeled tweets using Skip-Gram with negative sampling. The filter heights we chose are 2, 3 and 4, which is because empirically 5 or more gram features often produce noises in text classification. And we used 10 different filters for each filter heights in the network. So after max-over-time pooling, we will have a 30 x 1 feature map. The number of neurons used in the first fully connected layer is 20, which yield a best result among 10, 15, 20 and 25. A more thorough grid search on these hyper-parameters cannot be applied due to the long training time of CNN.

The results for sentiment task and negative reason task are in Table 6 and Table 7 respectively, where N/A in some rows mean that classifier failed to predict any example as the corresponding classes.

Table 6 CNN performance for sentiment task

Category	Recall	Precision	F1-measure	Overall Accuracy
Positive	0.655	0.795	0.718	
Negative	0.937	0.796	0.861	0.790
Neutral	0.418	0.687	0.520	

Table 7 CNN performance for negative reason task

Overall Accuracy	0.601		
Category	Recall	Precision	F1-measure
not negative	0.649	0.854	0.738
can't tell	0.270	0.197	0.228
bad flight	0.265	0.155	0.196
late flight	0.617	0.532	0.571
customer service issue	0.581	0.711	0.639
booking problem	0.163	0.132	0.146
lost luggage	0.629	0.538	0.580
flight attendant complaints	0.169	0.113	0.136
cancelled flight	0.607	0.653	0.629
damaged luggage	N/A	0.000	N/A
long lines	N/A	0.000	N/A

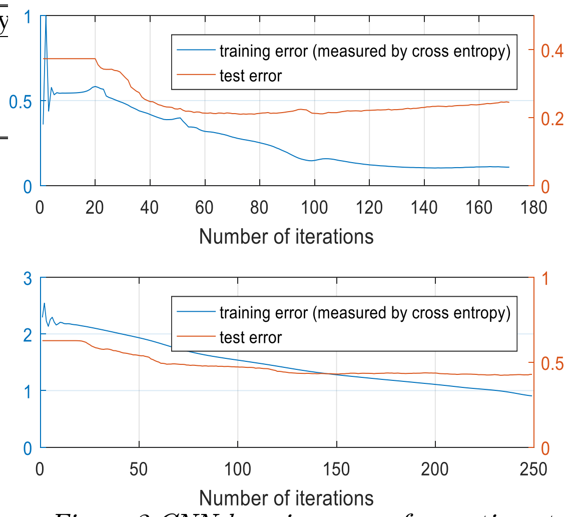

Figure 2 CNN learning curve for sentiment task (above) and negative reason task (below)

We see that CNN performs pretty well in the sentiment classification task. This is because word2vec extracts semantic features of words and the convolution neural network can capture some local feature among adjacent while learning useful high-level features through training, but in this task CNN didn' t outperform SVM as expected. Also, in the negative reason classification task, CNN got a much lower overall accuracy than SVM. The learning curve is in Figure 2, where the training error is measured by cross entropy and the test error is measured by (1 - accuracy). We find that when we got a decreasing training error, the test error didn't go down but increased at the end. These results are probably because: 1) the size of our dataset is not large enough for the training of CNN, which results in overfitting; 2) the word2vec feature we trained might has noises and not accurate enough; 3) the convolution operation on text data might not be as effective to capture local features as applied on image data; and 4) the max-over-time pooling operation might lose some important local information.

6 Conclusion and Future Work

In conclusion, we explored four learning techniques and successfully applied them on our learning problem. SVM performs best and yields 79.6% accuracy in sentiment task and 64.8% accuracy in negative reason task. We also tried CNN with word2vec features, which gives promising results and would be useful if we have a larger labeled dataset for training.

In the future, we consider to combine Recurrent Neural Networks (RNN) and CNN[15] since RNN can 'remember' all previous information of a tweet and CNN can well capture the inter-word information.

7 References

[1] M. Taboada, J. Brooke, M. Tofiloski, K. Voll, and M. Stede, "Lexicon-based methods for sentiment analysis," *Comput. Linguist.,* vol. 37, no. 2, pp. 267-307, 2011.

[2] T. Mikolov, K. C. and, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space," *CoRR,* vol. abs/1301.3781, 2013.

[3] B. Liu and L. Zhang, "A Survey of Opinion Mining and Sentiment Analysis," in *Mining Text Data*, C. C. Aggarwal and C. Zhai, Eds. Boston, MA: Springer US, 2012, pp. 415-463.

[4] B. Pang and L. Lee, "Opinion Mining and Sentiment Analysis," *Found. Trends Inf. Retr.,* vol. 2, no. 1-2, pp. 1-135, 2008.

[5] S. O. Orimaye, S. M. Alhashmi, and E.-G. Siew, "Performance and trends in recent opinion retrieval techniques," *The Knowledge Engineering Review,* vol. 30, no. 1, pp. 76-105, 2015/001/001 2015.

[6] J. Liu, S. Seneff, and V. Zue, "Dialogue-oriented review summary generation for spoken dialogue recommendation systems," presented at the Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Los Angeles, California, 2010.

[7] B. J. Jansen, M. Zhang, K. Sobel, and A. Chowdury, "Micro-blogging as online word of mouth branding," presented at the CHI '09 Extended Abstracts on Human Factors in Computing Systems, Boston, MA, USA, 2009.

[8] P. D. Turney, "Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews," presented at the Proceedings of the 40th Annual Meeting on Association for Computational Linguistics, Philadelphia, Pennsylvania, 2002.

[9] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up?: sentiment classification using machine learning techniques," presented at the Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10, 2002.

[10] "Data Source: https://www.crowdflower.com/wp-content/uploads/2016/03/Airline-Sentiment-2-w-AA.csv."

[11] Y. Kim, "Convolutional Neural Networks for Sentence Classification," *CoRR,* vol. abs/1408.5882, 2014.

[12] "The General Inquirer Lexicon: http://www.wjh.harvard.edu/~inquirer."

[13] "MPQA Subjectivity Cues Lexicon: http://mpqa.cs.pitt.edu/lexicons/."

[14] "Bing Liu Opinion Lexicon: http://www.cs.uic.edu/~liub/FBS/opinion-lexicon-English.rar."

[15] T. H. Nguyen and R. Grishman, "Combining Neural Networks and Log-linear Models to Improve Relation Extraction," *CoRR,* vol. abs/1511.05926, 2015.