Amazon Fine Food Reviews Classification

Zhangyi Ye (Rocky)

Abstract

The purpose of this paper is to summarize machine learning approaches to text classification described in past papers and also describe the experiment and results for my text classification project. I'll start by talking about some of the best practices and tips of text classification I found in literature reviews. I'll then go through my independent project which involves classifying Amazon find food review into categories. I will finish by talking about further improvement and the approaches that could be taken in the future.

1 Summary of Text Classification Approaches

My text analytics project takes the form of text classification. More specifically, sentiment analysis. Text classification is the one of the most common text analytics projects in the industry. Text classification is based on machine learning techniques that automatically build a classifier by learning the characteristics of the categories from a set of pre- classified documents (Sebastiani, 2002). It is used in information extraction and summarization, text retrieval, and question-answering etc. Most of the text data for classification is collected from the web and vary in sources and formats. So there is quite some processing and cleaning involved especially with traditional machine learning approaches that tend to deliver fast and robust solution to industry problems. Traditional machine learning approaches involve the following common steps:

Document Representation

Document Representation represents a given document in a form which is suitable for data mining system. There are several ways in which the conversion of documents from plain text to instances with a fixed number of attributes in a training set can be carried out. Bag-Of-Words (BOW) is the most commonly used word-based representation method (Jindal, 2015). It is also common to apply stemming and remove irrelevant words to the tokens to further reduce dimensions.

Constructing a Vector Space Model

This next step applies weights measuring the importance of each term in the document. TFIDF, which stands for Term Frequency Inverse Document Frequency (Jindal, 2015).

Application of a data mining algorithm

Decision trees, naive-bayes, rule induction, K-nearest neighbors and support vector machines are all common machine learning techniques used (Thangaraj, 2018).

As time goes on, deep learning approaches have emerged as a prospect for achieving these aims with their ability to capture both syntactic and semantic features of text without requirements for high-level feature engineering, as is the case in earlier methods (Hai Ha Do, 2019).

However, from the review of the state-of-the-art in aspect-level sentiment analysis and presented in this paper, it is clear that deep learning is still in the early stages. Given the relationship between aspect and opinion, improved performance can be obtained by joint extraction and classification of aspect, category and sentiment. However, many robust studies opt to perform only aspect extraction or categorization, and those who jointly perform aspect detection and sentiment analysis, have not yet achieved optimal performance (Hai Ha Do, 2019). Also, most of the industry implementation of text classification does not deep learning as it takes a long time to train neural nets and the performance is not guaranteed to be higher than traditional machine learning approaches. In the future, deep learning text classification may be more common when more powerful computers are

introduced or more efficient nnets with fewer parameters are available.

2 Experiment Design

For this independent project, I chose a text classification project which aims to classify amazon fine food reviews into good sentiment and bad sentiment.

Dataset

Dataset description: the dataset I used is called Amazon find food review dataset. It is around 300 MB and has 56,8454 rows and each row contains a customer review. The dataset can be downloaded from Kaggle. All documents have labels. Here the label is review score ranging from 0 to 5. Here is a distribution of scores:

5 63.8%
4 14.1%
1 09.1%
3 07.5%
2 05.2%

As we can see that more than half of the reviews are 5 and as scores get lower, the proportion gets smaller too. The average length of the text is 436 words. After grouping 5 and 4 into 'high_score' and 1,2,3 into 'low_score' the new distribution is

High_score: 78%
Low_score: 22%

Data Cleaning

To tackle this text classification, I applied the standard data science work-flow. After I downloaded the data from Kaggle, I held out 25% of the data as the test data set. All the cleaning and processing afterwards are only applied to the training set. The test set remains unseen.

Next I cleaned the reviews:

1. Get rid of all punctuations, numbers, and words that contains any special characters or have length less than 2 characters.

2. Covert all words to lower case, delete stop words and delete words that appear only once.

3. Compute a vocabulary file that include all the words from the cleaned text data.

4. Convert all training data to Bag of Words (BoW)

All the cleaning and normalizing is done using the nltk library which provides easy-to-use text wrangling functions. After processing the training data, I applied both traditional machine learning and deep learning approach to extract insights from the data.

Traditional Machine Learning

Traditional Machine learning approaches involve transforming the data set that each feature is a unique word in the document. The value or weight for each feature is represented using Term Frequency Inverse Document Frequency (FIDF). I also enlarged the feature space by choosing to include bi-grams.

I then applied two classification machine learning algorithms: Logistic Regression, and Support Vector Machine (linear Kernel). Both algorithms are linear as the underlying patterns in text classification is often linear, and both are fast-to-compute and easier to interpret.

For both machine learning models, I adjusted three sets of parameters:

1. N-Gram (1 vs 2)
    - 1-gram includes only individual tokens and 2-gram includes both individual tokens and every sequential pair of tokens.
2. Use IDF (Yes vs No)
    - 'Yes' means enabling inverse-document-frequency reweighting which is a method for reducing the importance for words that appear in more training documents.
3. Loss Function (L1 vs L2 for Logistic Regression and Hinge vs Squared Hinge for SVM)
    - L1 regularization adds sum of absolute values of coefficients as penalty whereas L2 regularization adds sum of squared values of coefficients are penalty.
    - 'hinge' is the standard SVM loss (used e.g. by the SVC class) while 'squared_hinge' is the square of the hinge loss.

All of the training is implemented using sklearn. 10-fold cross validation performance including accuracy, precision, recalls are measured for each set of these parameters. The best model is the one that has the highest cross validation accuracy.

Deep Learning

I used Convolutional Neural Network (CNN) to train the model as CNN has feature invariance which can capture positive or negative sentiment

anywhere in the review. CNN is also fast to train on GPU since it can process multiple features at the same time.

Similar to the previous approach, the input to the nnet is cleaned bags of words. There are several parameters I tuned:

1. Embedding size (100 vs 300)
2. Embedding type (pretrained vs random)
    - Pretrained embedding is glove embeddings from the Stanford webpage. Specifically, vectors trained on Wikipedia data.
3. Dropout rate (0.3 vs 0.5)
    - 0.5 means more regularization
4. Convolution Layers (2 vs 3)

For other parameters: n_filters = 100, window_size = 8, l2_penalty = 0.0003. I applied early stopping and 20 max training epoc for each set of parameters.

All of the training is implemented using Keras. Validation performance especially accuracy, is calculated for each epoc of the training. At the end, the highest performance is recorded for each set of these parameters. The best model is the one that has the highest test accuracy.

Prediction

After the best models is selected and saved, I use it to predict future reviews. It will be able to predict whether the review is positive or negative. The model first cleans the input text (the same way the training data is cleaned) and then extract vocabularies that are only present in the training data. After that it uses the best model on the cleaned list of tokens. The model can only generalize well if the review is fine food related as the training data is only from the amazon fine food review dataset. It may generalize poorly if other types of texts are used. The final product/API will heavily rely on the prediction script.

Productization:

I finally wrap the model within a REST API which transports input and output using GET and POST methods. I also used Swagger which provides an easy to use interface to execute commands. I have also converted all Jupyter notebooks to python scripts so that results from the model are reproducible.

3 Results

Logistic Regression

N_gram	idf	Penalty	Accuracy
1	Yes	L2	0.90
1&2	Yes	L2	**0.93**
1	Yes	L1	0.90
1	No	L2	0.90

We see that two grams performs better than 1 gram; IDF performs better than non-IDF; L2 penalty performs slightly better than L1 penalty. L2 penalty, IDF, 2 gram is the best model.

SVM (linear kernel):

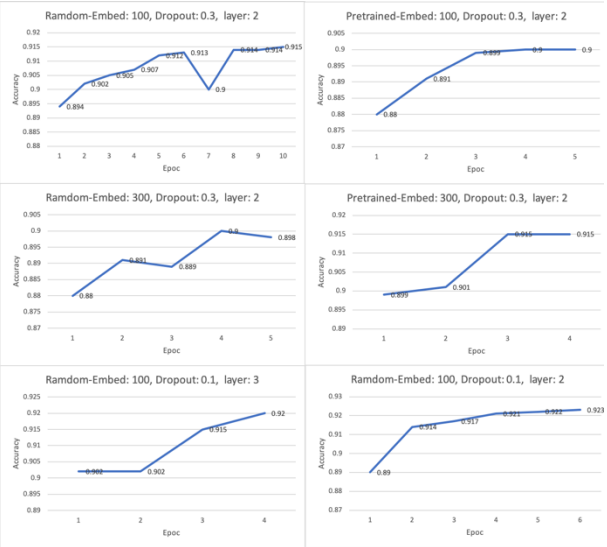
N_gram	idf	Regression Loss	Accuracy
1	Yes	squared_hinge	0.91
1&2	Yes	squared_hinge	**0.94**
1	Yes	hinge	0.91
1	No	squared_hinge	0.90

We see that two grams performs better than 1 gram; IDF performs better than non-IDF; squared_hinge has the same performance as hinge. Squared_hinge, IDF, 2 gram is the best model.

CNN:

Embedding	Pret-train	Dropout	Conv Layers	Accuracy
100	No	0.3	2	0.915
100	Yes	0.3	2	0.900
300	No	0.3	2	0.898
300	Yes	0.3	2	0.915
100	No	0.1	3	0.92
100	No	0.1	2	**0.923**

Below are detailed graphs depicting validation accuracies every epoc for each set of parameters:



Notice that some training sessions have fewer epocs than other due to early stopping. We see that not pretrained embedding with size 100, dropout 0.1 and 2 cov layers performs the best.

4 Conclusion

We conclude that the best models is the SMV (linear kernel) model with Squared_hinge, IDF, 2 gram. It has validation accuracy of 0.94. Logistic regression model with L2 penalty, IDF, 2 gram. It has accuracy 0.93. The convolutional nnet with embedding with size 100, dropout 0.1 and 2 cov layers came third. It has accuracy 0.923. The difference between the best of these 3 methods are really small (within 0.02). So they all work really well on learning the sentient on this particular data set. Many of the 6% classification error might be irreducible error since many people have different standards of their rating after they leave a review. For example, after personally reviewed the data set, I found that for similar reviews, some people give a 3 and others give a 4. These situations are very difficult for machines and humans alike to classify. As the data is fairly balanced 78% vs 22%, 0.94 accuracy is really good. Both recall and precision are really close to accuracy meaning that the model did well classifying both positive and negative sentiments.

I used the SVM model as the final model. It is both fast and accurate. The Deep learning model slow to train so that it may not apply to many of the industry tasks that requires efficient training for fast iterations. Finally, I incorporated this model into the production process.

Overall, I think the project is successful. The data set is informative and of reasonable size. Most of the reviews are easy to classify that is why tradition machine learning approaches beat deep learning. In the future, I would love to use LSTM or Transformers for this task since they consider the whole context of reviews rather than breaking it down into bag of words. I would also love to try multi-classification and predict the actual ratings instead of the sentiment.

5 References

Jindal, Rajni et al. “Techniques for text classification: Literature review and current trends.” Webology 12 (2015): n. pag.

Thangaraj, Sivakami. “Text Classification Techniques: A Literature Review.” Interdisciplinary journal of information, knowledge, and management 13 (2018): 117–135. Web.

Sebastiani, F. (2002). Machine learning in automated text categorization. ACM Computing Surveys, Vol. 34.

Hai Ha Do, PWC Prasad, Angelika Maag, Abeer Alsadoon, Deep Learning for Aspect-Based Sentiment Analysis: A Comparative Review, Expert Systems with Applications, Volume 118, 2019.