Assignment 1. Movie reviews sentiment analysis

[Report]

Sokolov Aleksey
University of California San Diego
sokolov@ucsd.edu

1. INTRODUCTION

Sentiment analysis is a challenging subject in machine learning. People express their emotions in language that is often obscured by sarcasm, ambiguity, and plays on words, all of which could be very misleading for both humans and computers. Purpose of this project is to predict emotion of a review by understanding meaning and relationships between words.

The dataset used in this project consists of 50,000 IMDB labeled movie reviews, specially selected for sentiment analysis in the publication [1]. The sentiment of reviews is binary, meaning the IMDB rating < 5 results in a sentiment score of 0, and rating >=7 have a sentiment score of 1. No individual movie has more than 30 reviews.

Typical training data looks similar to the following sample:

8196_8; 1; "
I dont know why people think this is ...".

Each review has the following fields:

- id - Unique ID of each review
- sentiment - Sentiment of the review
- review - Text of the review

Originally the data contained some HTML tags, stopwords and punctuation symbols. To make it suitable for analysis we use BeautifulSoup library to remove the HTML tags. We also use re package for dealing with regular expressions to remove punctuation, though we still keep part of it, since we are tackling a sentiment analysis problem, and it is possible that "!!!" or ":-(" could carry sentiment, and should be treated as words. Finally we get list of English-language stop words from python Natural Language Toolkit and filter all of those from our data corpus and utilize Stemming package to treat same meaning words as one word.

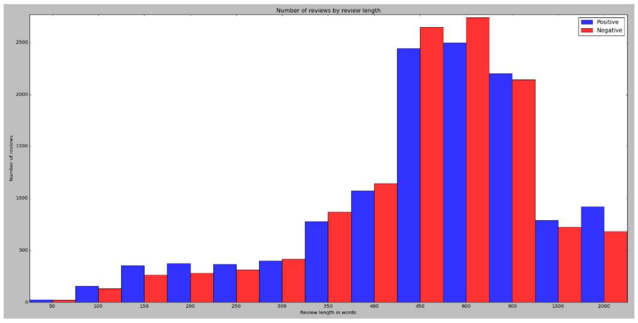

Figure 1: Average lengths of the reviews

On average each review consists of about 121 words. 10 words most frequently found in the reviews and number of corresponding occurrences are: movie - 24955, film - 19213, one - 13135, like - 11238, even - 7684, good - 7419, bad - 7394, would - 7036, really - 6262, time - 6208.

However this doesn't give us any useful information about data, since both of the labels contain those words. Therefore we divide the reviews into two halves according to their labels, find several hundreds most frequent words for both of them and throw away the words which appeared to be in both halves (see Table 1).

This approach lets us visualize some features of our data by plotting labeled reviews as points of different colors. Each of two axis on the following plots represent the number of occurrences of particular word in a review (see Figures **??**-5).

Figure 2 shows us that a movie has higher chances of negative rating if the review contains word "worst". However word "better" doesn't guarantee positive rating even though it is one of the most frequently occurring words in positive reviews.

On Figure 3 we are analyzing our data on two negative sentiment words dimensions and observe significantly lower amount of positive reviews, number of which decreases even more as the reviews have larger numbers of one or both of the words.

We can observe interesting difference between Figure 4 and Figure 5. They represent the same features of the data - "excellent" and "awful" dimensions. However the words fre-

Sentiment words count			
Positive	Negative	Positive	Negative
ever - 2733	many - 2909	works - 848	unfortunately - 931
acting - 2435	show - 2861	city - 790	crap - 895
plot - 2433	great - 2640	strong - 767	guess - 888
say - 2414	man - 2517	sometimes - 746	flick - 816
better - 2382	worst - 2479	late - 730	decent - 815
m - 2039	life - 2427	age - 699	annoying - 795
bad - 1907	still - 2282	stories - 696	tries - 790
excellent - 1681	love - 2152	simple - 695	ridiculous - 789
wonderful - 1370	best - 2096	roles - 693	hour - 784
nothing - 1301	awful - 1556	james - 665	god - 764
perfect - 1241	poor - 1481	relationship - 656	gore - 752
loved - 1087	boring - 1476	episodes - 656	save - 746
amazing - 1058	stupid - 1428	fantastic - 654	attempt - 735
favorite - 962	terrible - 1391	david - 652	car - 733
heart - 959	waste - 1359	murder - 646	except - 732
today - 934	worse - 1248	brother - 646	blood - 731
brilliant - 926	supposed - 1186	experience - 642	obviously - 730
enjoyed - 879	oh - 1084	oscar - 628	thinking - 720
highly - 870	horrible - 1046	including - 627	ok - 714
fine - 857	couldn - 1042	musical - 625	lack - 698

Table 1: Sentiment words count

quencies in the figures were calculated in different manner. For the first we evaluated each individual review independently of other reviews and for the second we concatenated all reviews of each movie into corresponding large review. Joined reviews tend to give us mixed distribution of both features and the separated reviews have clear difference in dimensions. This fact means that when several users are reviewing the same movie the chances are that their judgment and rating of the movie will be different, which means the reviews will contain both positive and negative sentiment words with high probability.

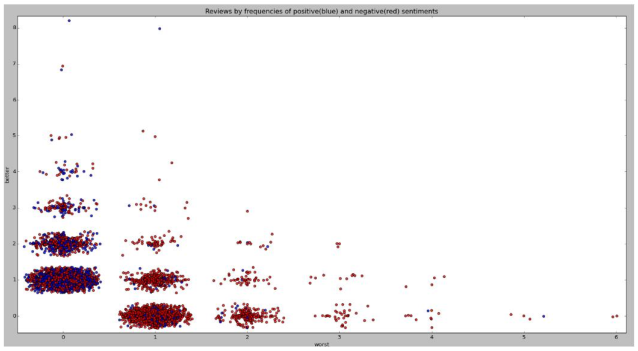

Figure 2: Separated reviews "better-worst" frequencies.

After initial exploratory data analysis we were able to identify some features of the data and their dependencies. As we proceed further it turned out that the accuracy of all models greatly depends on the way we extract and represent the features.

2. PREDICTIVE TASK

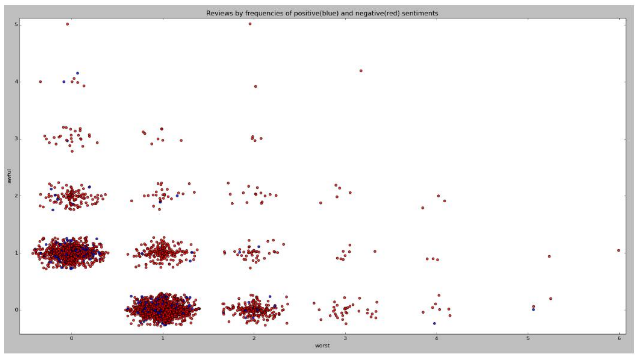

Figure 3: Separated reviews "awful-worst" frequencies.

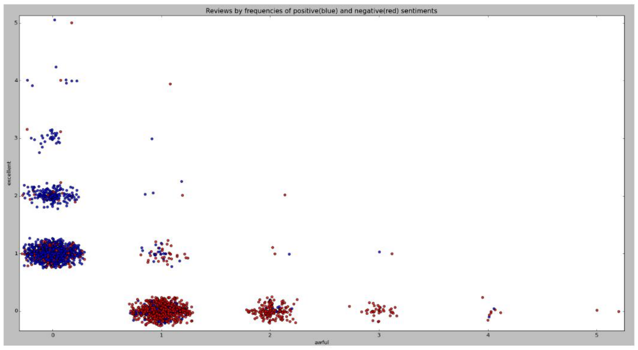

Figure 4: Separated reviews "excellent-awful" frequencies.

Predictive task to be studied on this dataset is identifying the sentiment of a review based on its text content. For this

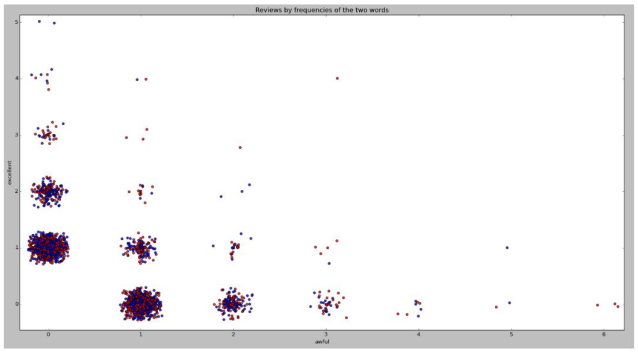

Figure 5: Reviews joined by movie "excellent-awful" frequencies.

purpose we need to explore feature extraction methods, find out which features are more relevant than the others and identify the most efficient classification algorithm for our dataset among support vector machine, logistic regression, random forests and neural networks approaches. The accuracy of the algorithms will be compared on the randomly selected cross validation set and than on the test set.

The straight forward way for building word representation for any natural language processing task is to use the bag of words approach. We first build a fixed dictionary of the 5,000 most frequent tokens for the number of features, but ignore the 40 most rare terms to avoid attaching too much importance to individual movie titles, since any movie in the collection is limited to at most 30 reviews. Thus we create 5000-dimensional feature vector for our every labeled review to use it on the reviews comparison. Using the training feature vector we will fit it to the several most reasonable classifiers and validate the model's predictions on the test set of 25000 labeled reviews. We also use cross-validation set of 15% the size of the training set to tune parameters of our model.

Afterwards we will try more sophisticated feature extraction techniques as n-grams, tf-idf or neural networks based vector space in case bag of words will not provide low enough error rate.

3. LITERATURE

The original publication our dataset comes from also tried to predict review binary label but used more complex model for that purpose [1]. It presents a model that uses a mix of unsupervised and supervised techniques to learn word vectors capturing semantic termâĂŞdocument information as well as rich sentiment content. The proposed model can leverage both continuous and multi-dimensional sentiment information as well as non-sentiment annotations. The reason for such a mixed model is that unsupervised vector-based approaches to semantics can model rich lexical meanings, but largely fail to capture sentiment information that is central to many word meanings and important for a wide range of NLP tasks. The model combines latent semantic analysis, latent Dirichlet allocation and vector space model. One of the datasets similar to ours is Rotten tomatoes dataset, which was collected and utilized for sentiment categorization by Pang and Lee [2]. They presented three types of approaches for support vector machine - done-vs-all, regression and metric labeling and considered positive-sentence percentage as similarity measure.

The state-of-the-art techniques for our data type we explored include support vector machines, random forest [5], logistic regression for classifying a review and n-gram models, term frequency-inverse document frequency, neural network algorithms for feature extraction.

Feedforward neural network language model[4], implemented in word2vec library follows four-gram neural net language model architecture. It can be used in our project for the purpose of extracting features representing them by continuous bag-of-words or continuous skip-gram model. The training is done using stochastic gradient descent, backpropagation and softmax in the output layer. Though the model is not efficient for purely getting word vectors, but rather for deep semantic and syntactic analysis of the review corpus.

Random forest is method for classification that constructs a multitude of decision trees at training time and outputs the classes, correcting decision trees' habit of overfitting to training set.[5] The training algorithm applies bootstrap aggregating to tree learners. Given a training set $X = x_1, , x_n$ with responses $Y = y_1, , y_n$, it repeatedly selects a random sample with replacement of the training set and fits trees to samples. After training, predictions for unseen samples x' can be made by averaging the predictions from all the individual regression trees on x'.

4. FEATURES

Features extraction turned out to be the critical part of this assignment, because different classification algorithms resulted in extremely close accuracy on the same feature representation formats. Since all data we've got is the reviews texts, rating will always be the function of the review words of different weights, depending on a model. First of all we cleaned the data by removing irrelevant tags, stopwords, punctuation and stemming the words with similar meaning. Then we have chosen the most frequent words to represent our feature vector. This naive "bag of words" model can only provide moderate results, since it assumes independence of all the words in the review and doesn't take word order into account.

The model accuracy can be improved by applying n-gram to analyze contiguous sequences of the words. So far we are classifying a review based on the most frequently appearing words in the dataset, assuming that it won't cover only a tiny portion of words and that all words are equally relevant, both of which are not true. So even better approach will be to use term frequency-inverse document frequency model, which assigns weight to every word based on it's frequency. This way we consider rare words more relevant and don't need to remove stopwords which can potentially contain some useful information any more. Finally we are utilizing the word2vec library to build the feature vector for us.

5. MODEL AND RESULTS

We are fitting every type of feature vector we received from bag of words, word2vec, n-gram and tf-idf to logistic regression, support vector machine, random forest algorithms. We used sklearn implementations, tuned the model parameters, but didn't bring any changes into their algorithms, so we will skip their theoretical description. To increase our classifiers performance we applied principal component analysis to reduce the data dimensionality. It didn't work well, since reducing the variance of this data on 1% only reduced the feature vector by 20%, which is not that significant. However normalizing the feature vector helped a lot, especially for support vector machine which took several times more amounts of time to converge. To avoid overfitting and apply grid search for parameter tuning we used randomly selected the cross-validation set.

We originally expected the support vector machine to give the best results as it is often considered the-state-of-art approach for classification problems. However we discovered that all three approaches have almost the same error rate when applied to the same feature vectors (though SVM does take much more time). More than that all of them achieved almost 100% accuracy on the test set, which means that the data features dimensions are easily separated. So the accuracy of the classification depends primarily on the feature extraction models. Among classification algorithms the random forest approach provided slightly higher result and low execution time, so we only use it for further classification of feature vectors representations.

Also we expected the word2vec model created in Google to outperform simple models as n-gram and tf-idf. But it performed slightly better than the bag of words model. Our assumption is that our corpus of data is not large enough for this library to achieve highest results - its performance is best on much larger datasets, consisting of many millions of entries. However it gave us some insight on the semantic properties of the data and distinguished the words sentiments. For example it distinguished that the word "awful" is similar to words "terrible, horrible, dreadful, laughable, horrendous, ridiculous, pathetic" with decreasing probabilities, which is pretty amazing result.

Final results of our random forest classifier on the test set are 95.4% for feature vectors extracted using tf-idf, 89.7% - for bigram, 82.1% - for bag of words approaches.

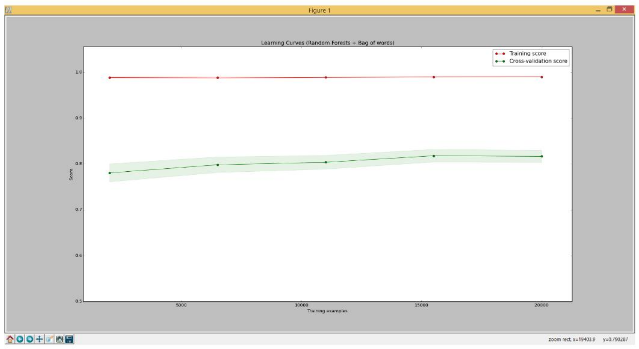

Figure 6: Learning curve of random forest + bag of words models

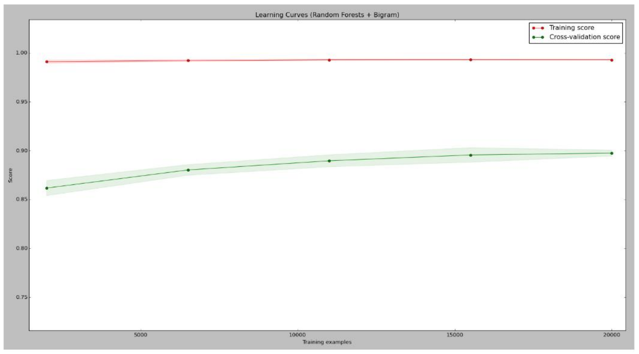

Figure 7: Learning curve of random forest + bigram models

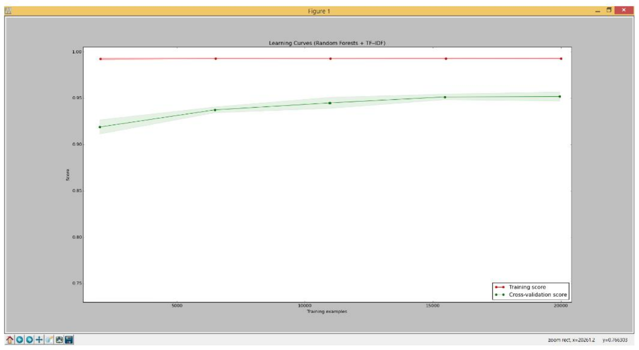

Figure 8: Learning curve of random forest + tf-idf models

6. CONCLUSIONS

In this project we studied the quality of vector representations of words derived by various models on a collection of rated IMDB movie reviews. We observed that it is possible to get high quality word vectors using very simple model architectures, compared to the popular neural network models(word2vec). We discovered that the accuracy of the classification depends primarily on the feature extraction models. We were also able to identify some semantic and sentiment information of the words. Best result were achieved using combined random forest and tf-idf models.

7. REFERENCES

[1] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. (2011). "Learning Word Vectors for Sentiment Analysis." The 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011).

[2] Pang and L. Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In ACL, pages 115âĂŞ124.

[3] Ronan Collobert and Jason Weston (2000). Natural Language Processing (almost) from Scratch. In Journal of Machine Learning Research 1 (2000), pages 1-48.

[4] Mikolov, Tomas, Chen, Kai, Corrado, Greg and Dean, Jeffrey. "Efficient Estimation of Word Representations in Vector Space." CoRR abs/1301.3781 (2013)

[5] Breiman, L. 2004a. 'Consistency For A Simple Model

Of Random Forests,' Technical Report 670, Statistics Department University Of California at Berkeley, September 9, 2004.

[6] Christopher M. Bishop (2006). Pattern Recognition and Machine Learning. Springer. p. 205. "In the terminology of statistics, this model is known as logistic regression, although it should be emphasized that this is a model for classification rather than regression."

[7] Cristianini, Nello; and Shawe-Taylor, John; An Introduction to Support Vector Machines and other kernel-based learning methods, Cambridge University Press, 2000. ISBN 0-521-78019-5

[8] 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).

[9] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in The adaptive web, pp. 325-341, Springer, 2007.

[10] A. Ansari, S. Essegaier, and R. Kohli, "Internet recommendation systems," Journal of Marketing research, vol. 37, no. 3, pp. 363-375, 2000.

[11] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka, "Training algorithms for linear text classifiers," in Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 298-306, ACM, 1996.

[12] S. Salas and E. Hille. *Calculus: One and Several Variable*. John Wiley and Sons, New York, 1978.

[13] Deniz Demir, Olga Kapralova, Hongze Lai, "Predicting IMDB Movie Ratings Using Google Trends," Dept.Elect.Eng,Stanford Univ., California, December, 2012