Predicting Amazon ratings using Neural Networks

Hen Su Choi Ortiz (hensu.choi@gmail.com)
University of California San Diego

Rajat Shah (rrs003@ucsd.edu)
University of California San Diego

Abstract

In this work we are comparing different approaches of predict a users numeric rating of a product review solely from the text of the review. We use three different methods, bag-of-words model and the state-of-the-art models word2vec (using RandomForests) and word2vec(using K Nearest Neighbors) using data from Amazon reviews.The three models are evaluated over the same corpus and the results, execution time and MSE are compared.

Introduction

Description of the dataset

The data used is collected from the set of Amazon reviews that can be found at the Stanford University SNAP website (*SNAP Web data: Amazon reviews*,n.d.). The dataset has a JSON format where each example is a user review for a product. The dataset consists of 100,000 user reviews of clothing that are sold by Amazon. The dataset chose was of size 100000 different reviews.

The predictive model for user ratings, the dataset was partitioned into training and testing subsets using a 50/50 split of the overall dataset. Table 1 shows the statistical data for the ratings and fig. 1 shows the distribution of ratings. A word distribution analysis was run over the ratings considered "good" and comparing it with the reviews that gave a rating of 1. The respective wordclouds are shown.(Figures 2 and 3)

Rating	
Min	1
1st quartile	3
Median	5
Mean	3.964
3rd quartile	5
Max	5
Standard Deviation	1.3685

Table 1: Numerical Summary

Literature

The word2vec approach is a tool that takes a text corpus as input and produces the word vectors as output. It first constructs a vocabulary from the training text data and then learns vector representation of words. The resulting word vector file can be used as features in many natural language processing and machine learning applications.

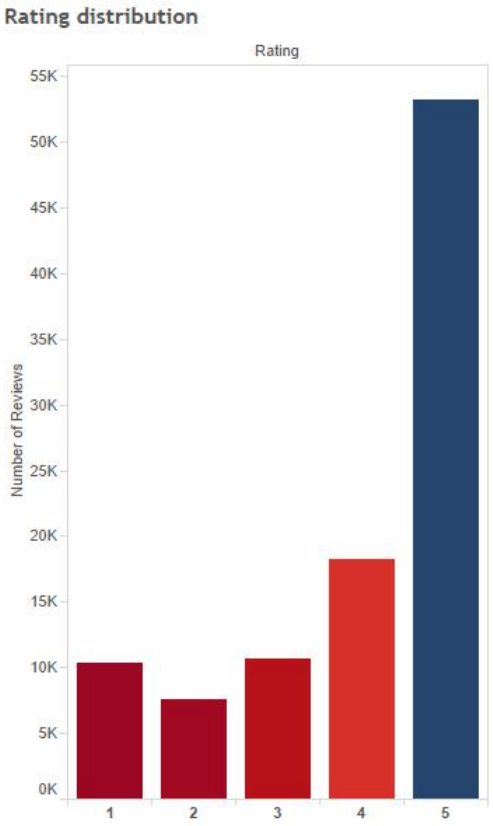

Figure 1: Rating Distribution

The model generates supervised learning tasks from the corpus. Continuous Bag of Words (CBOW) and Skip-Gram CBOW forces the neural net to predict current word by surrounding words, and Skip-Gram does the opposite, forces the neural net to predict surrounding words of the current word.Figure 4 shows the representation of both models.(Mikolov, Chen, Corrado, & Dean,2013)

The simplest version of the continuous bag-of-word model (CBOW) there is only one word considered per context, so the model will predict one target word given one context word, which is like a bigram model.

Doc2vec is a modification of the word2vec algorithm and is relatively similar. The main difference is in the model architecture. In the doc2vec model, the algorithms used

Figure 2: Wordcloud of reviews with rating = 5

Figure 3: Wordcloud of reviews with rating = 1

are distributed memory (dm) and distributed bag of words (dbow). Taddy et. al found that Word2Vec yields lower misclassification rates than both Doc2Vec-based classification They found that all methods are often outperformed by phrase-count logistic regression with rare-feature up-weighting and carefully chosen regularization.(Taddy,2013)

On the other hand, Qu et al. in the paper "The Bag-of-Opinions Method for Review Rating Prediction from Sparse Text Patterns" introduces a state of the art representation called "bag-of-opinions", where an opinion, within a review, consists of three components: a root word, a set of modifier words from the same sentence, and one or more negation words. They overcame the problems that unigram and n-gram representations present such as they cannot (or does not d it in a robust form) capture meaningful expressions, which are essential for prediction models of ratings.The model works in the following way: each opinion is assigned a numeric score which is learned from a corpus of reviews. For the actual test case of a domain-dependent review, the reviews rating is predicted by aggregating the scores of all opinions in the review and combining it with a domain dependent unigram model(Mikolov et al.,2013)

Methodology

In our approach, we try to predict the ratings of different amazon products based on the text review of the product. The Naive Bayes model that we learnt in class is the most appropriate model for this kind of prediction. We predict the ratings solely based on the text reviews. To implement the Naive Bayes model, we use the bag of words algorithm, where each review is broken up into a set of words and we assume that the words are not related to each other. The bag of words model is our baseline for our prediction. Apart

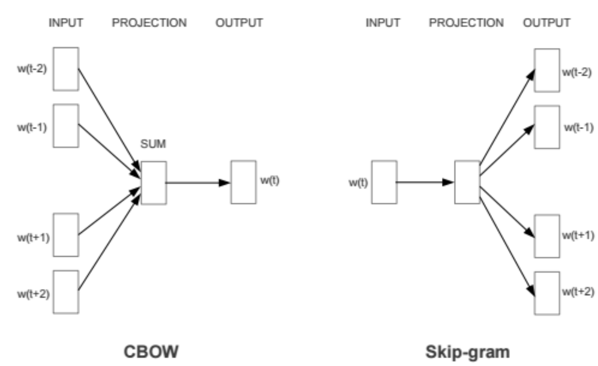

Figure 4: CBOW vs Skip-gram.

from that we use two flavors of the word2vec model. The Word2vec model is a neural network implementation that learns distributed representations for words. One major advantage of this model is that it learns quickly as compared to other models. It also does not need the data to be in a labeled format to make meaningful representations.

First we to get meaningful representation of word vectors, we clean up the reviews from our data by removing stop words. Stop words are word like a,is,the which do not carry much meaning by themselves. Next, we also remove any hyperlink link text and non alpha characters. We did all the pre-processing by using specialized python packages. After the pre-processing of the text, we used the scikit-learn python package to train our bag of words model.

For the Word2Vec models, there were different parameters we fitted our model to get the most optimal results. This included the choosing between hierarchical softmax and negative sampling for the training algorithm, the number of context words the training algorithm takes into account, downsampling of frequent words, number of parallel process and the dimensionality of the word vectors. After doing several iteration and checking our predictions for the MSE, we fitted values which gave us the least MSE.

One challenge with the Amazon dataset is the variable-length reviews. We took individual word vectors and transformed them into a feature set that is the same length for every review. Our very first approach was to take the average of the sum of all the word vectors in a review. After we got our average word vectors for each review, we used it to fit a random forest model. Our model did a little worse than the bag of words model. This might be because the reviews vary a lot in the vocabulary the use and so a Naive bayes model with no relationship will work better if the training set is not big enough (¿ 10 million reviews).

After this, we used clustering to cluster semantically similar words. Grouping words this way is called vector quantization. K Means clustering algorithm is used to cluster the words. After this, we classify the reviews into bag of centroids with semantically similar words clustered together. After this, we train our model again using random forest and get a slightly better result than the bag of words.

Conclusions

Results are presented in Table 2 and in the graphical representations fig. 5 and 6

Future Direction

With better computing power, we would like to increase the size of our training set and check if we get similar results for the three algorithms we used. Googles improvement on the results using Word2Vec is on a much larger corpus, more than a billion words. It would also be a good to compare the performance of models that preserve the word ordering and meaning of sentences.

Challenges Faced

While training our bag of words model, we initially started out with a training set of 500,000 data points. However, after an hour and half of running our model, we decided to it would be extremely time consuming and CPU intensive to compute a model on a training set of this size and so we scaled down our training set to 100,000 data points. We were running our code in an iPython notebook and while its convenient to execute snippets of code, we were running into the issue of our kernel dying quite often. For this reason, we would lose our fitted models and had to recompute our models every single time our kernel died. This was really time consuming.

Table 2: Results.

Algorithm	Execution Time	Accuracy
Bag of words	1509.30851102	60.764%
Word2vec (random forest)	530.36894836	58.932%
words2vec (knn)	91.5718159676	63.809%

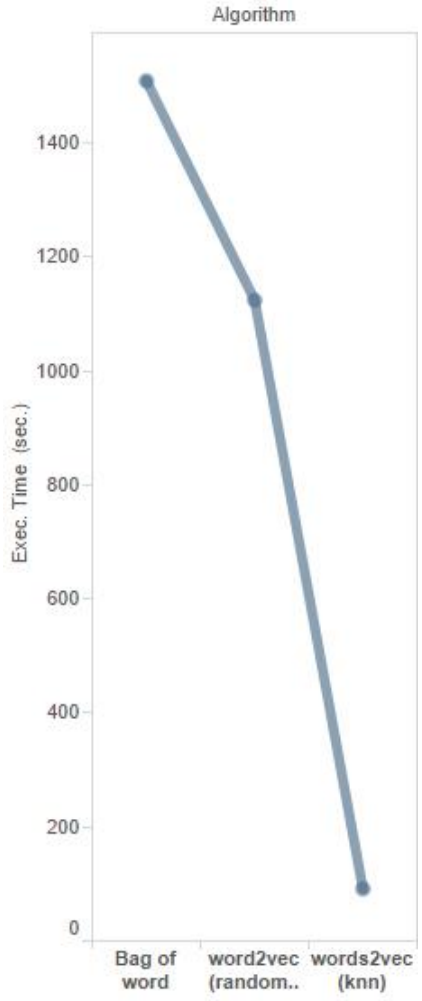

Figure 5: Execution time

[h]

References

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

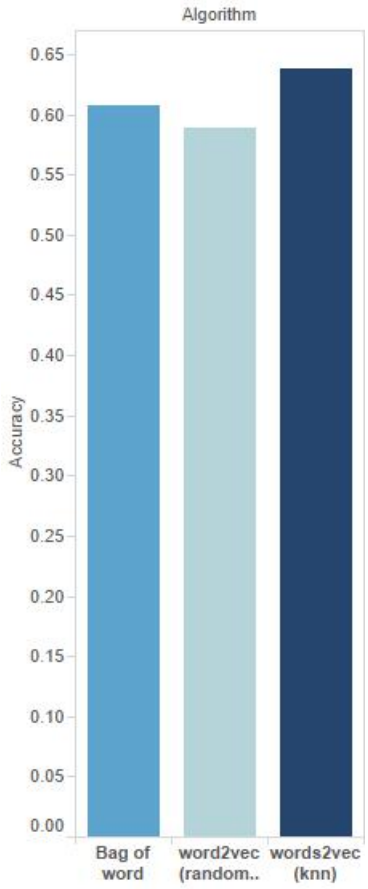

Figure 6: Accuracy

Snap web data: Amazon reviews. (n.d.). http://snap.stanford.edu/data/webAmazon-links.html.

Taddy, M. (2013). Document classification by inversion of distributed language representations. *arXiv preprint arXiv:1504.07295*.