EE 660 Course Project

# Learning on Amazon Reviews

**Project Type:** (1) Design a system based on real-world data

**Number of student authors:** 2

Khalid Alharbi
khalidal@usc.edu

Yasssamin Neshatvar
neshatva@usc.edu

Date Dec 6th 2021

# Contents

# 1. Abstract

Machine Learning is a vast field which includes different types of Learning methods. In this project, we are interested in assessing the performance of the different types of ML methods namely: supervised learning, semi-supervised learning, unsupervised learning and transfer learning. First, we obtained two online available datasets (Amazon reviews) and used them in our learning procedure. In supervised learning, the following learning algorithms were implemented: Logistic regression, perceptron, support vector machine, decision tree, and Adaboost. For semi-supervised learning, we executed two algorithms: QNS3VM and label spreding. For unsupervised learning, all data labels were removed. Then expectation-maximization (EM) and kmeans algorithms were implemented. For transfer learning, the TrAdaboost algorithm was used to transfer the learning from our first dataset to our second dataset. We observed that all learning methods implemented above performed better than the trivial baseline system. Supervised learning had the highest performance due to the abundance of data. Other learning methods didn't reach to the highest performance obtained by supervised learning.

# 2. Introduction

## 2.1 Problem Type, Statement and Goals

In this project, we have chosen two publicly available datasets from Amazon. Each dataset is reviews for a particular product category. The first dataset is reviews of home improvement category. The second dataset is reviews of kitchen category. Both datasets are structured in the same way meaning that they all have the same feature names and differ only in the star rating values and reviews given.

Our goal in this project is to conduct binary classification for positive and negative reviews based on the star ratings given in each dataset. Our objective is to perform supervised learning, transfer learning, semi supervised learning and unsupervised learning on our datasets.

We have chosen these two datasets because we have never done anything in the field of product reviews or in general datasets that included mostly text features. In addition, the idea of transfer learning was very interesting to us and we wanted to implement it in our project. We believe that the major challenges of our project were preprocessing and dealing with large number of datapoints. The preprocessing took most of our time as we had to learn about different techniques of cleaning the reviews feature and eventually using google word2vec algorithm to convert the reviews into vectors. In addition, we had around 5 million and 3million datapoints in our kitchen and home improvement datasets, respectively. Dealing with this much data was tough and at some we had to cut down on our data to make the algorithms work (especially transfer learning) efficiently.

## 2.2 Our Prior and Related Work (Mandatory)

No prior related work

## 2.3 Overview of Our Approach

### 2.3.1 Supervised Learning (SL)

For supervised learning we trained on the home improvement dataset. We started with 2 baseline models which are described below. Then we performed 4 supervised leaning techniques namely Decision tree, Adaboost, Linear SVM and logistics regression. Hyperparameter optimization was performed to find the best set of parameters for each model. The models were then compared to each other based on the accuacy scores obtained.

The baseline systems are

- The trivial baseline: a classifier that randomly picks an output label, with probability given by the label's frequency of occurrence in the training data. Accuracy is 50%
- The non-trivial baseline: Perceptron was used which gave an accuracy of 78%.

#### 2.3.2 Transfer Learning (TL)

For transfer learning, we used the TrAdaboost algorithm and importance weights. It is good to mention that we had early stopping in our TrAdaboost algorithm.

The baseline system is:

- Decision Tree from the supervised learning algorithm was used as a baseline with accuracy score of 68%

#### 2.3.3 Semi-Supervised Learning (SSL)

For semi supervised learning, we have used Q3SVM and label spreading algorithms as our models. To compare the models for semi supervised learning, we have used the accuracy score.

The baseline system is

- Decision Tree from the supervised learning algorithm was used as a baseline with accuracy score of 68%

#### 2.3.4 Un-Supervised Learning (USL)

For unsupervised learning, we have used kmeans and EM algorithms as our models. To compare the models, we have used the accuracy score.

The baseline model is

- A model that randomly picked an output label with probability of 0.5 which is equivalent to a model that does a coin flip. Accuracy is 50%

# 3. Implementation of Supervised Learning

## 3.1. Data Set

The datasets chosen for this project are product reviews taken from Amazon. We have picked two datasets each from a different domain. The primary dataset is home improvement reviews as source domain and the secondary dataset (used for transfer learning) is from the kitchen reviews as a target

domain. There are 15 features in each domain that are identical in naming to each other. A summary of the features and their data types are below.

Kitchen Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4874890 entries, 0 to 4874889
Data columns (total 15 columns):
 #   Column             Dtype
---  ------             -----
 0   marketplace        object
 1   customer_id        int64
 2   review_id          object
 3   product_id         object
 4   product_parent     int64
 5   product_title      object
 6   product_category   object
 7   star_rating        float64
 8   helpful_votes      float64
 9   total_votes        float64
 10  vine               object
 11  verified_purchase  object
 12  review_headline    object
 13  review_body        object
 14  review_date        object
dtypes: float64(3), int64(2), object(10)
memory usage: 557.9+ MB
```

Home Dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2629867 entries, 0 to 2629866
Data columns (total 15 columns):
 #   Column             Dtype
---  ------             -----
 0   marketplace        object
 1   customer_id        int64
 2   review_id          object
 3   product_id         object
 4   product_parent     int64
 5   product_title      object
 6   product_category   object
 7   star_rating        object
 8   helpful_votes      float64
 9   total_votes        float64
 10  vine               object
 11  verified_purchase  object
 12  review_headline    object
 13  review_body        object
 14  review_date        object
dtypes: float64(2), int64(2), object(11)
memory usage: 301.0+ MB
```

The number of data points in the kitchen dataset is 4,874,890 and the number of data points in the home improvement dataset is 2,629,867. For our classification, we have chosen the star ratings as our labels. We have decided to perform a binary classification where star ratings of greater than 3 are positive reviews and star ratings of less than or equal to 3 are negative reviews.

## 3.2. Dataset Methodology

Initially, we started with all of our datapoints from both the home and kitchen datasets. We preprocessed all datasets and split our data into train and test set. We allocated 20% for test set and the remaining 80% was allocated for training set. This implies that 3,899, 912 datapoints for training in kitchen dataset and 2,103,893 datapoints in the home improvement. However, in running our base model, we faced a memory shortage and hence had to down sample our datasets. We further down sampled our data by removing reviews that had less than 30 words. This helped us to increase our model accuracy. The final datapoints area as follows: 262,472 datapoints for each class in the home improvement data. This means 524, 944 datapoints in total. Similarly, for the kitchen dataset, we down sampled to 293,126 datapoints for each class. Hence in total we ended up with 586,252 datapoints.

For our supervised learning section, we split the data into train and test sets. We allocated 20% for the test data. As for validation, we performed 5 fold

cross validation using the grid search from sklearn's module. This was done to obtain the best parameters for each supervised learning algorithm.

| | Data Set | Train | Test | Grid Search |
|---|---|---|---|---|
| Supervised Learning | Home | 469,001.6 | 117,250 | 5fold |

### 3.3. Preprocessing, Feature Extraction, Dimensionality Adjustment

Since the main feature of our dataset (the review_body) was in text format, we had to use data preprocessing techniques similar to those of natural language processing. To proceed, we decided to concatenate the review_headlines feature with the review_body feature into one feature called the reviews. After this step, we implemented different functions which are listed below to clean up the reviews and get it prepared for the implementation of the word2vec. First we removed all the html tags from the reviews. Next, we removed the URLs and applied contractions to the words. Later, we removed non alphabetical texts and stop words. Finally, we used the WordNetlemmatizer on the reviews. After we got the reviews feature cleaned up, we applied the google2vec to get the reviews into a vector form. This transformation resulted in 300 new features for each datapoint. As for the other features, we decided to remove 'customer_id', 'product_id', 'review_date', 'marketplace', 'product_category', 'review_id', 'product_parent', 'product_title' from our datasets as training on them had no impact on the performance of the model. In addition, for the remaining features we transfomed them into categorical features. For instance, the verified_purchase feature was 'Y', or 'N' and we made it into categorical of 1,2. After cleaning up of the data, we removed all the NAN values from the dataset. There were not a lot of them so removing them did not make any significant difference.

As for the data labels we picked star_ratings feature as our classification labels. The star ratings were from 1 to 5 and we decided on performing a binary classification. As such, star ratings above 3 were considered positive review and assigned a value of 1 and anything less than or equal to 3 was considered negative review and assigned a value of 0. Hence our labels were 1 and 0 for positive and negative reviews respectively.

After all the preprocessing was done, we tried some simple models on the data to check its performance, however, because of the large amount of data, we could not run the models and ran out of memory space. Hence, we down sampled our datasets to 524, 944 datapoints for home dataset and 586,252 datapoints for kitchen dataset. In addition, we trained the model on all our

features after preprocessing and again trained it using only the 300 features of the reviews. This proved to us that training on the 300 word2vec features of the reviews gave us better accuracy than training on all the features. As such, we dropped all our features and only kept the 300 wordvecfeatures generated from our reviews feature

## 3.4. Training Process

Perceptron

We implemented perceptron. We chose this model because we thought it's a simple model to use. We used this model with its default parameters from sklearn. The accuracy results are summarized in the table below.

| | Train | Test |
|---|---|---|
| Accuracy Score | 78% | 78% |

Decision Tree

We implemented decision trees. We chose this model because we thought its simple and could yield good results on our dataset. We did a grid search using 5 folds search to obtain the best max_depth for our model by trying different values of 2,3,4 and 6. We got the max_depth to be 6. The results of train and test accuracies are listed in the table below.

| | Best Parameter | Train | Test |
|---|---|---|---|
| Accuracy Score | 'max_depth': 6 | 68% | 68% |

Adaboost

We implemented Adaboost. We chose this model because we thought that it would give us a better result because of its adaptive basis function form. As for hyperparameter tuning, we performed 5fold grid search on number of estimatorsn and the learning rate. We tried values of 'n_estimators': [10,50,80] and 'learning_rate':[0.001,0.01,0.1]. We got our optimal values to be 80 and 0.1 respectively. Because our dataset was large, we expected Adaboost to take a long time to run. The results of train and test accuracies are listed in the table below.

| | Best Parameter | Train | Test |
|---|---|---|---|
| Accuracy Score | 'learning_rate': 0.1, 'n_estimators': 80 | 73% | 72% |

SVM

We implemented SVM. We chose this model because we had read in the blog posts that this model worked well on the amazon reviews. As for hyperparameter tuning, we decided to perform 5fold grid search on l1 and l2 norms. We got our optimal value to be the l2 norm. The results of train and test accuracies are listed in the table below.

| | Best Parameter | Train | Test |
|---|---|---|---|
| Accuracy Score | 'penalty': 'l2' | 80% | 80% |

Logistics Regression

We implemented Logistics Regression. We chose this model because it is one of the common techniques used for classification. As for hyperparameter tuning, we decided to perform 5fold grid search on l1 and l2 norms. We got our optimal value to be the l2. The results of train and test accuracies are listed in the table below.

| | Best Parameter | Train | Test |
|---|---|---|---|
| Accuracy Score | 'penalty': 'l2' | 80% | 80% |

As for the complexity of all the models above, we had 300 features in our dataset and the number of our datapoints both in training and test set were way more than 7-10 times our degrees of freedom. Hence, we were confident that we are not overfitting in our model.

### 3.5. Model Selection and Comparison of Results

For each model we performed a 5fold grid search to obtain the best parameters. The best model was then chosen based on the highest accuracy score obtained.

## 4. Final Results and Interpretation SL

Based on the supervised learning models described above, the best model was chosen based on the accuracy score. The table below provides a summary of all the models accuracies on train and test data with their optimal hyperparameter.

| SL models | Hyperparameter | Train | Test |
|---|---|---|---|
| Perceptron | 'alpha':0.0001, 'penalty': 'l2' | 78% | 78% |
| Decision Tree | Max_depth = 6 | 68% | 68% |
| Adaboost | 'learning_rate': 0.1, 'n_estimators': 80 | 73% | 72% |
| SVM | 'penalty': 'l2' | 80% | 80% |
| Logistic Regression | 'penalty': 'l2' | 80% | 80% |

In all of the models above, the best performance was clearly SVM model. Decision tree and Adaboost use simple algorithm to classify the data. The reason behind this is that our data is not categorical otherwise decision tree will perform better. It is true that when the number of data is large, we will have both in-sample and out-of-sample errors equal to each other. We verify this by looking at the epsilon in Hofdling inequality, as we can see that it's inversely proportional to N (number of datapoints).

# 5. Implementation of Transfer Learning

## 5.1. Data Set

We used the home improvement data as our source domain and used kitchen data as our target domain.

## 5.2. Dataset Methodology

For the transfer learning method, we had to cut down our data further because we could not run the Tradaboost as we kept on getting early stop after the first iteration. As such we cut down our data for home to 600 datapoints for test set and 3000 for training set. Similarly, we reduced the kitchen data (target domain) train to 400 and test to 200 datapoints.

| | | Train | Test | |
|---|---|---|---|---|
| Transfer Learning | Home (source) | 3000 | 600 | 5-fold |
| | Kitchen (target) | 400 | 200 | |

## 5.3. Preprocessing, Feature Extraction, Dimensionality Adjustment

The same preprocessing techniques described in section 3.3 are applied here.

## 5.4. Training Process

TrAdaboost

Our implementation of TrAdaboost was very challenging. We kept on hitting the early stop point of the algorithm. Even with our reduced size dataset, we ended up stopping after 4 iterations only. Nonetheless, we still trained it on our test set and got the following results.

| | Train | Test |
|---|---|---|
| Accuracy Score | 78% | 74% |

As for the complexity of the model, we had 300 features in our dataset and the number of our datapoints both in training and test set were around 10 times the features in the source and target domains.

# 6. Final Results and Interpretation for TL

The table below provides a summary of the accuracies on train and test data on the TrAdaboost algorithm.

| | Train | Test |
|---|---|---|
| Accuracy Score | 78% | 74% |

Since our baseline model for TL was decision tree with an accuracy of 68%, we can say that our TL model performed better. We can see that the SL has better accuracy on its own domain. But when transferring this to a new domain it performed quit well. In addition to TL, we also performed importance weights model which did not work well on our dataset and resulted in numerous implementation errors. For convergence in TrAdaboost, as we increase the number of iterations we face an early stop at the third iteration, hence we can't say that the system converged as the convergence requires more iterations (Wenyuan Dai, 2007).

# 7. Implementation of Semi-Supervised Learning

## 7.1. Data Set

The home improvement data is used for this part of the learning.

## 7.2. Dataset Methodology

For semi supervised learning method, we modified our dataset to include 1000 train datapoints and 300 test datapoints. We used 8 labeled datapoints and 992 unlabeled datapoints. The information about datapoints is summarized in the table below.

| Semi Supervised Learning | Train labeled | Train unlabeled | Test |
|---|---|---|---|
| | 8 | 992 | 300 |

## 7.3. Preprocessing, Feature Extraction, Dimensionality Adjustment

The same preprocessing techniques described in section 3.3 are applied here.

## 7.4. Training Process

QNS3VM

We implemented QNS3VM. We chose this model because it was one of the algorithms taught in class. The results of train and test accuracies are listed in the table below.

| | Train | Test |
|---|---|---|
| Accuracy Score | - | 61% |

Label Spreading

We implemented the label spreading algorithm from the sklearn library. This model is similar to the basic Label Propagation algorithm. The results of train and test accuracies are listed in the table below.

| | Train | Test |
|---|---|---|
| Accuracy Score | 98% | 68% |

As for the complexity of all the models above, we had 300 features in our dataset and the number of our datapoints both in training and test set were about 3 times our degrees of freedom. Hence, our model may overfit.

# 8. Final Results and Interpretation for SSL

The table below provides a summary of all the models accuracies on train and test data. Based on the semi supervised learning models described above, the best model was chosen based on the accuracy score.

| | Train | Test |
|---|---|---|
| QNS3VM | - | 61% |
| Label Spreading | 98% | 68% |

Here in SSL, we aimed to reach the same accuracy as the baseline. For label spreading, we performed the same as the baseline model. Implementing QNS3VM with different numbers of labeled datapoints gave us a different result each time. Using small amount of data was always predicting one label. The best accuracy we got was using 8 data points as labeled and the rest as unlabeled datapoints. We think that 8 datapoints were enough to establish a discriminant function that can learn to separate the two classes using QNS3VM algorithm. QNS3VM algorithm uses schemes that involve non-linearity optimization. (F. Gieseke, 2019)

# 9. Implementation of Un-Supervised Learning

## 9.1. Data Set

The home improvement data was used.

## 9.2. Dataset Methodology

For unsupervised learning method we used the same dataset as the one we used in semi supervised learning with one difference that we used the train data as unlabeled data. We predicted on the test set and then compared our prediction to our test labels which we had kept aside as in the table below.

| unsupervised Learning | Train unlabeled | Test |
|---|---|---|
| | 1000 | 300 |

### 9.3. Preprocessing, Feature Extraction, Dimensionality Adjustment

The same preprocessing techniques described in section 3.3 are applied here.

### 9.4. Training Process

Kmeans

We implemented Kmeans. We chose this model because it was one of the algorithms taught in class. The results of train and test accuracies are listed in the table below.

| | Train | Test |
|---|---|---|
| Accuracy Score | 62% | 61% |

EM Algorithm

We implemented the EM algorithm because it was taught to us in class for unsupervised learning. The results of train and test accuracies are listed in the table below.

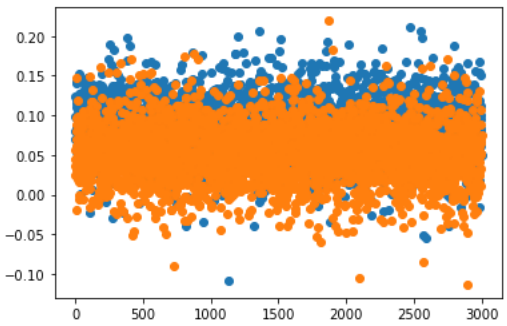
| | Train | Test |
|---|---|---|
| Accuracy Score | 62% | 58% |

## 10. Final Results and Interpretation for USL

Based on unsupervised learning models described above, the best model was chosen based on the accuracy score. Kmeans performed better.

| | Train | Test |
|---|---|---|
| Kmeans | 62% | 61% |
| EM | 62% | 58% |

For our dataset, kmeans proved to perform better than EM algorithm. The reason behind that could be that GMMs differ from k-means clustering in that they provide posterior probabilities based on information about the center (mean) and variability (variance) of each cluster. Moreover, our data showed an overlap between some features as shown in the fig below. Separating the two classes by hard threshold is better in this case than soft threshold. (Leon Bottou, 1995)



## 11. Contributions of each team member

Both team members worked together from planning the project to writing the report.

## 12. Summary and conclusions

We observed that supervised learning always performed better for our dataset. Semi supervised and unsupervised learning lacked the ability to give the same results as supervised learning. As for transfer learning, it performed better than both semi supervised and unsupervised learning.

# 13. References

F. Gieseke, A. A. (2019, Mar 25). *QN-S3VM BFGS optimizer for semi-supervised support vector machines.* Retrieved from Github: https://github.com/NekoYIQI/QNS3VM/blob/master/qns3vm.py

Leon Bottou, Y. B. (1995). *Convergence Properties of the K-Means Algorithms.* Retrieved from proceedings.neurips: https://proceedings.neurips.cc/paper/1994/file/a1140a3d0df1c81e24ae954d935e8926-Paper.pdf

Wenyuan Dai, Q. Y.-R. (2007). *Boosting for Transfer Learning.* Retrieved from D2L: https://cse.hkust.edu.hk/~qyang/Docs/2007/tradaboost.pdf