Sentiment Analysis for Amazon Review with Supervised Learning and Transfer Learning

EE 660 Course Project

Longhao Yang
longhaoy@usc.edu

December 9, 2022

1 Abstract

Sentiment analysis is always a trending topic in natural language processing fields. This project digs into this problem and predicts customers' sentiments as positive or negative using the "Multi-Domain Sentiment Dataset (version 2.0) [1]". This dataset contains amazon reviews in 4 domains: book, DVD, electronic, and kitchen. I used the traditional ML methods including Logistic Regression, Support Vector Machine, Random Forest, and XGBoost. I also developed this content into a transfer learning problem, and want to predict sentiments of DVD reviews as the target domain from books reviews as the source domain using the **Subspace Alignment** method and **TrAdaBoost** method. The best model for sentiment prediction is the **Random Forest Classifier**, which generated the most accurate prediction with 83.00% accuracy. The best method for the Transfer Learning model is **TrAdaBoost** which result in 76% accuracy.

2 Introduction

2.1 Problem Type, Statement and Goals

As the customers on the internet marketplace grow rapidly, we can obtain more feedback and reviews from customers and enhance their experience upon the reviews. One way to interpret millions of comments from the website is by conducting Sentiment Analysis using **Supervised Machine Learning** techniques. The purpose of this paper is to predict and classify the attitudes of Amazon reviews toward a product as **positive** or **negative**. Another goal of the project is to generate a sentiment word cloud from the reviews by interpreting the data sets.

As mentioned above, we can predict the reviews' sentiments (positive or negative) by training the reviews' data set. However, there are some categories of product with minor sales amount, so it doesn't doesn't have enough reviews to train the Supervised Learning Model. Thus, let us introduce the **Transfer Learning** (TL) Method which learned the relationship between these two categories. By learning the relationship between two datasets (books and DVDs), we can train the model using books and apply it to the other, which possibly generates better performance. This approach cuts down the size of datasets for model training and laborsaving, but since the dataset are similar, the performance of the approach is still stable.

2.2 Our Prior and Related Work (Mandatory)

Prior and Related Work - None

2.3 Overview of Approach

In this project, I will use the Logistic Regression Classifier as the baseline model for both the Supervised Learning (SL) and Transfer Learning (TL) Methods. Other machine-learning models are described in the section below. For the sentiment classification problem, the key metric I used is accuracy as a measure of model performance. Other metrics including confusion matrix, f-score, and ROC-AUC score are also measured for supplemental measurement.

2.3.1 Supervised Learning Methods

1. The trivial baseline: Randomly assign the class labels (positive and negative) to all the samples based on the class probability.
2. The non-trivial baseline: Logistic Regression Classifier to predict the positive and negative class.
3. Other systems:
    - Support Vector Machine
    - Random Forest Classifier
    - XGBoost

2.3.2 Transfer Learning Methods

The baseline system for TL is the Random Forest model learned from Supervised Learning section.

Other systems:

- Subspace Alignment
- TrAdaBoost

3 Implementation

3.1 Data Set

The dataset in the project comes from the "Multi-Domain Sentiment Dataset (version 2.0) "[1] and I used the dataset from *processed.acl.tar.gz* file. The dataset contains preprocessed and balanced review data. All the reviews come from 4 categories of products, including books, DVDs, electronics, and kitchen. In each category, there are equal numbers (1000) of positive and negative labeled reviews, and around 4000 unlabeled reviews. Specific dataset information is shown in the Table 1 below.

Table 1: Dataset Sample Size Summary

	Class	Books	DVD	Electronics	Kitchen
Labeled	Positive	1000	1000	1000	1000
	Negative	1000	1000	1000	1000
Unlabeled		4465	3586	5681	5945

All data samples in the dataset are tokenized words. That means, for each sample, it shows each distinct word in the comment and how many times it appears in the comment. The label of the comment is the last feature in the sample. An sample example are shown in the Table 2 below.

For example: "I love this book this book is fun"

Table 2: one sample of dataset

I	love	this	book	is	fun	Class
1	1	2	2	1	1	positive

3.2 Dataset Methodology

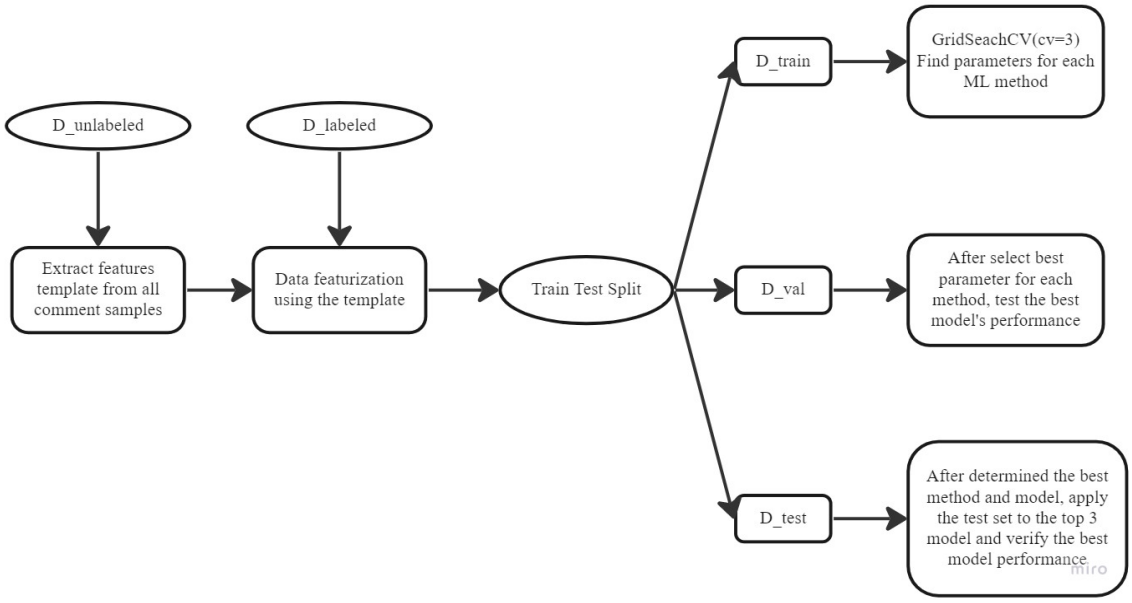
For the supervised learning section, I used all reviews from 4 categories' unlabeled data $D_{unlabeled}$ to build the "dictionary" used for featurization. During the training process, the 4 categories' labeled datasets are split into training D_{train}, validation D_{val}, and test set D_{test}. The proportion for each set is 60%, 20%, and 20%.

In order to choose the best parameter settings for each model I attempt to use, I used the grid search method to test each parameter pairs (*GridSearchCV* in python). This process involves the cross validation step and I set the number of folders to 3. The validation set I split intentionally is used for model selection among all best models selected from each method. During the method selection process, validation set is used to calculate the metrics and evaluate the method performance.

Test set is only used at last for the best model of each method, and validate that the selected model performs best on the test.

The flowchart Figure 1 below illustrates how I utilized the dataset from the beginning to the end.

Figure 1: Dataset Flowchart



3.3 Preprocessing, Feature Extraction, Dimensionality Adjustment

3.3.1 Data preprocessing and Cleaning

For natural language processing problems, extracting features is the most complicated and important step in the entire study. In the supervised learning section, the first step of data preprocessing is concatenating the words into sentences with the number of counts and replacing the "_" underscore characters with blank spaces. The sentences are "fake" sentences since they didn't preserve the word order of the original sentences. Thus, I did the text cleaning which includes removing punctuation and stopwords (e.g. I, am, he, she, the, a, etc.), and tokenizing the words. I also attempted to use

the tokenized words with stemming. However, this step didn't make a huge influence on the result (accuracy), so I didn't include it here.

3.3.2 Word Embedding

After cleaning the text content in the sample space, the next step is to convert the tokenized word into computer-readable features - Matrix. There are three methods I used for the word embedding step: Bag of Word, TF-IDF, and Word2Vec.

- **Bag of Word** (BOW): Collect all words appeared in the dataset (68562 distinct words) as features, and for each sample, count the number of each word to form a vector as one sample. I also reduced the feature size to 10000 since the original feature space is too large and might contain meaningless features. I used the python library *sklearn.feature_extraction.text.CountVectorizer* to implement the featurization method.

- **TF-IDF**: This featurization method is built based on the BOW method. The TF-IDF formula is declared below. [5]

$$\text{TF-IDF} = \frac{\text{\# of this word in the sample}}{\text{total \# of this words in the dataset}} \cdot log_{10}(\frac{\text{total \# of samples}}{\text{\# of samples contains this word}}) \quad (1)$$

  For each word (term) in the sample, it calculates the word's importance to the sample compared to the whole dataset. Thus, the TF-IDF features contain more information than a simple count of words. I also reduced the feature size to 10000 since the original feature space is too large and might contain meaningless features. I used the python library *sklearn.feature_extraction.text.CountVectorizer* to implement the featurization method.

- **Word2Vec**: The basis of the word2vec method is measuring the cosine similarity between words and quantifying the similarity with vectors. [3] The python library used for this method is *gensim.models.Word2Vec*. To generalize the vector for practical use, I also updated the original vector to the average of the word2vec vector, which is inspired by this article[4]. Due to time complexity issue, for the word2vec featurization, I only used the top 100 important features by setting the *vect_size* to 100. The result performance is influenced by the low feature size (low model complexity), which will be shown in the 3.5 section. I tested the word2vec method for 1000 features, but the model performance is not improved. Hence the 1000 features version is not included in this report.

This Table 3 below is a summary of the feature size for the dataset after 3 different featurization methods.

Table 3: Feature size summary

Featurization Method	num of features
Bag of Words (CountVectorizer)	10000
TF-IDF	10000
Avg Word2Vec	100

In order to cover the variety of words that appears in all possible comments (samples), I trained each of the three featurization models with all labeled datasets in all 4 product categories (books, DVDs, electronics, kitchen). During the featuring process, only$X_{labeled}$ are used for model training. For all machine learning methods I used, I always trained with all three pre-processed inputs.

3.4 Training Process

For all models I tested, I always fitted and tested the model with the dataset featured by all three methods. During each training process, I fitted the D_{train} dataset to the proposed model, and test

different parameters using the validation set data D_{val}, and find the best parameter setting using Grid Search with cross-validation method (GridSearchCV in sklearn) with 3 folders. I also tried to use the 5-folder method, but it is time-consuming with the repeated training processes. In the following section, I will present 3 machine learning models besides the trivial and non-trivial baseline models: **SVM**, **Random Forest**, **XGBoost**.

3.4.1 Trivial Baseline

For the trivial baseline, I chose to generate the prediction labels randomly following the training set labels' distribution ($Y_{labeled}$ class distribution in Table 4). After random assignment, the prediction accuracy on the validation set is 47.19%. It generated a 48.69% accuracy on the test set.

Table 4: Output class distribution

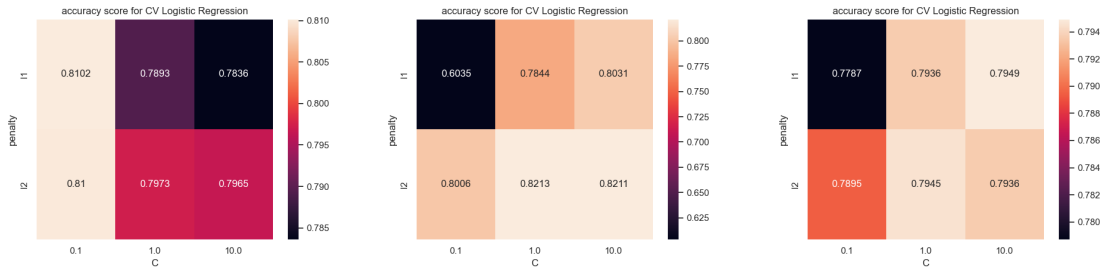
Class	P(Y = class)
positive (Y = 1)	0.5
negative (Y = 0)	0.5

3.4.2 Non-trivial Baseline

For the non-trivial baseline ML model, I selected the Logistic Regression Algorithm, since it is one of the most general classification models for binary classification problems. I used the python module *sklearn.linear_model.LogisticRegression* in this case. In this algorithm, the sigmoid function is used to calculate the probability that one sample belongs to a class. During the training process, the model's loss function is the Log Loss equation, which magnifies the influence of misclassification. Since this is the baseline model, I didn't apply any adjustments to the featured vectors.

In the parameter selection step, there are two main parameters I have considered: the regularization (penalty) term and the regularization strength. In the featurization vectors, there are 68562 features (distinct words) in the dataset. In order to reduce the model complexity, setting the regularization term can penalize too complex models or models with too many features and avoid the overfitting problem. I tested L1 and L2 regularization terms. For the regularization strength, I tried the strength equal to 0.1, 1, and 10. Thus, there are 6 pairs of regularization settings. By applying the 6 regularization setting to 3 featurization vectors, there are 18 different results. I tested the model accuracy using the validation set, and a summary is shown in Figure 2 below.

Figure 2: Accuracy - Logistic Regression with different featurization methods



(a) Count Vectorization (b) TF-IDF (c) Avg Word2vec

From the result above, we can see that the Logistic regression model provided the best prediction accuracy using the TF-IDF featurization vectors. The optimal Logistic Regression parameters for TF-IDF features is L2 regularization and with strength = 1. The baseline model accuracy is **82.13%**. I will use this accuracy as the measurement standard.

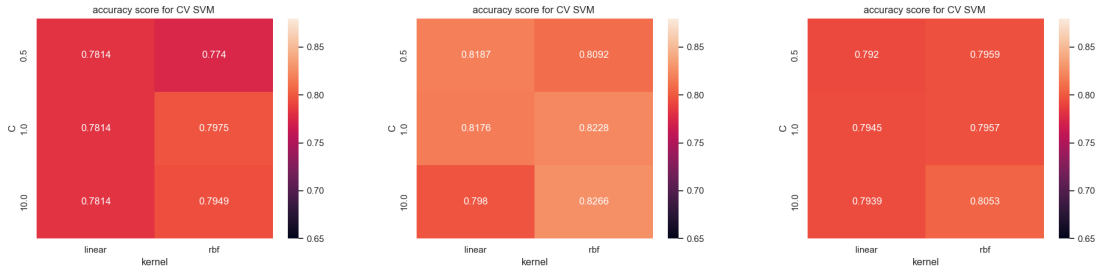
3.4.3 Support Vector Machine

The support Vector Machine Classifier (*sklearn.svm.SVC*) is another popular model for classification problems. It draws a hyperplane around the dataset which is transformed with different kernels[6]. The other important term in the SVM model is the margin size around the hyperplane. With different margin sizes, we can increase or decrease the margin based on our faith in the trend of the dataset.

I came up with this model since it aligns with my assumption of the feature's distribution. If some review contains the word "*good*", it unlikely includes the word "*terrible*". Thus, based on these two features (good and terrible), we can imagine two clusters on this feature dimension and I can draw a boundary in between to separate the positive and negative reviews.

During the parameter selection process, I need to select the kernel for this problem since a mathematical kernel function can make the prediction easier. Intuitively, I chose the **linear** and **rdf** kernels as the kernel candidates. Linear boundary cuts the hyperspace of good and bad comments. RDF boundary draws a "*circle*" for one class, and all other feature spaces belong to the other class. There is the squared L2 regularization term used in SVM with different strengths (0.5, 1.0, 10). The summary accuracy Figure 3 is shown below.

Figure 3: Accuracy - SVM with different featurization methods



(a) Count Vectorization (b) TF-IDF (c) Avg Word2vec

From the result above, we can conclude that the best model parameter is using "rdf" kernel with regularization strength = 10, which generates an accuracy score of **82.65%**. This result beats the non-trivial baseline (Logistic Regression) by 0.52%, which is a minor improvement.

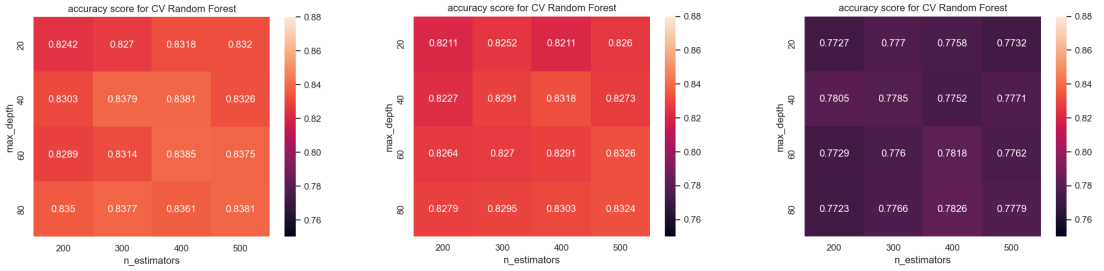
3.4.4 Random Forest

Since I am classifying the comments by text, the number of appearances of certain words (e.g. love, disappointed) determines the sentiments of the review content. The decision tree is another suitable method of dividing the feature space. The "*upgrade*" version – random forest *sklearn.ensemble.RandomForestClassifier* might be better since it is an ensemble version of the decision tree. It reduces the effect of overfitting since it adds more randomness during the feature selection and sample selection process. The decision tree improves the model by calculating the information gained through each split tree node.

There are several criteria for quality improvement from splitting nodes (leaves): Gini index, entropy, and log loss. By comparing the criterion, the entropy loss provides a better result in all three feature sets. Thus, I used the entropy loss in all Random Forest models. In terms of how to split the tree, there are many parameters to adjust. During the training process, I mainly test these two parameters: the number of trees in the forest (*n_estimators*) and the max depth of each tree (*max_depth*). The *n_estimators* helps generalize the result since it can build more trees in the "forest" and take the average. The *max_depth* parameter prevents the model from overfitting. If we don't set the max depth, the model will split every node to contain only one sample. For *n_estimators*, I selected 200, 300, 400, and 500 as testing values. For *max_depth*, the candidates are 20, 40, 60, and

80. All parameters are narrowed down to this range after many trials. The performance (accuracy) result is listed in Figure 4 below.

Figure 4: Accuracy - Random Forest with different featurization methods



(a) Count Vectorization (b) TF-IDF (c) Avg Word2vec

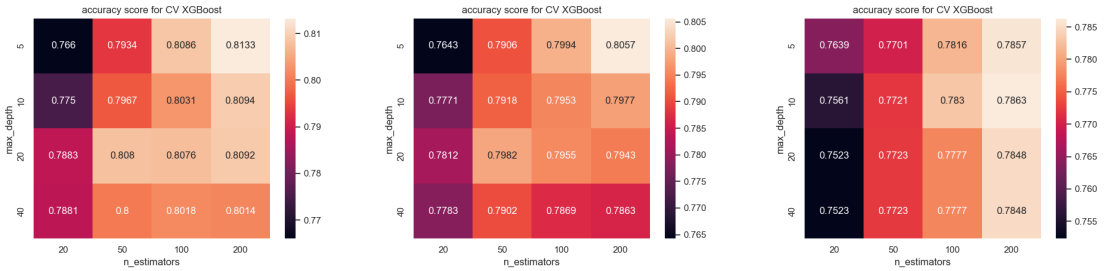
From the result above, we can conclude that the best RF model parameter is using the entropy loss criterion, set *n_estimators* = 400, *max_depth* = 60 and using the count vectorization feature space. This setting generates an accuracy score of **83.84%**. This result beats the non-trivial baseline (Logistic Regression) by 1.71%, which is better than the SVM model.

3.4.5 XGBoost

The previous tree-based model performs well, so I tried another tree-based model - XGBoost. In the XGBoost model *xgboost.XGBClassifier*, the trees are built on top of each other. "Each tree boosts the attributes that led to the misclassification of the previous tree. [2]" This tree is deeper and is the optimized version in general. However, there are some disadvantages that it is more likely to be an overfitting model. Thus, the parameter tuning process is important.

The booster I used is the *'gbtree'* booster by comparing the average accuracy with *'gblinear'* booster. I also did the GridSearch with the pair of *n_estimators* and *max_depth*. The *n_estimators* means the number of boosting rounds and the *max_depth* implies the depth of the base learning tree[8]. From the result in Figure 5, I observed that for all three kinds of features, when *n_estimators* = 200 and *max_depth* = 5, the model generates the best result.

Figure 5: Accuracy - XGBoost (*n_est*) with different featurization methods

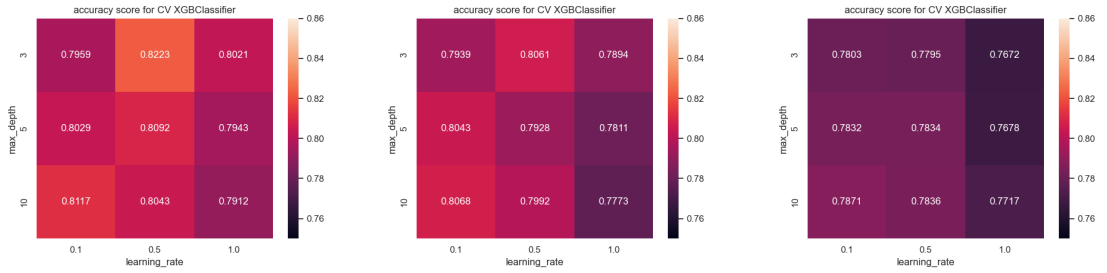


(a) Count Vectorization (b) TF-IDF (c) Avg Word2vec

Thus, I decided to fit the parameter *n_estimators* = 200, and tried various small *max_depth*. The final result is shown in Figure 6 below. From the result, We can conclude that the best XGBoost model parameter is using the *gbtree* booster, setting *n_estimators* = 200, *max_depth* = 3, *learning_rate* = 0.5, and using the count vectorization feature space. This setting generates an

accuracy score of **82.23%**. This model's performance beats the baseline model by 0.1% which is minuscule.

Figure 6: Accuracy - XGBoost with different featurization methods



(a) Count Vectorization (b) TF-IDF (c) Avg Word2vec

3.5 Model Selection and Comparison of Results

After the model training process described in section 3.4, I got the best model with certain parameters and features for each machine learning model.

The summary Table 5 below is an overview of the parameter, feature selection and metrics performance for each model. For featurization methods (word-ensemble methods), BOW is the abbreviation for Bag of Word methods (*CountVectorizer in sklearn*). In parameter selection for each model, the listed parameters are the optimal version in the training process. Other parameters not listed on the table use the default setting in *sklearn* library. The performance metrics I used is accuracy.

The entire dataset of 8000 reviews is divided into a training set of size 6400 (60%), a validation set of size 1600 (20%), and a test set of size 1600 (20%). Both validation set and test set accuracy is listed below.

Table 5: Model Performance Summary

ML Method	Featurization	Parameters	Val Acc.	Test Acc.
Trivial	None	None	49.69%	49.13%
Non-trival	TF-IDF	L2, C=1	82.87%	82.19%
SVM	TF-IDF	kernel=rbf C=10	83.81%	82.75%
Random Forest	BOW	max_depth=60 n_est=500	**86.12%**	**83.00%**
XGBoost	BOW	max_depth = 5 n_est = 200 eta = 0.5	81.87%	79.75%

From the summary table, we can see the SVM model beats the baseline model with 0.94% validation set accuracy and 0.56% in the test set. The improvement is minor. The XGBoost model is worse than the non-trivial baseline model and I think the reason is XGBoost is more likely to be overfitting because it developped an optimized deep tree based on the training dataset. The **Random Forest** model has the best performance among all models I tested. It outperforms all other models with a validation accuracy of 86.12% and test accuracy of 83%. Although the accuracy difference between the validation and test set indicates there is still an overfitting problem, it can be minimized with more parameter tuning.

An interesting part of the result is that, all models results in better metrics score with BOW transformed features or TF-IDF features. The averaged Word2Vector (avg w2v) is not the best features for all models. One interpretation is that the avg w2v features I have tested has only 100 features due to time and space complexity issue. It effected the performance when comparing to other features (BOW and TF-IDF) with 40047 dimension space.

4 Final Results and Interpretation

4.1 Final Results

The final system I adopted is the Random Forest Model, with the entropy loss criterion, the number of estimators = 400, and the max depth of the tree = 70. The best feature space that is suitable for the model is the Bag of Words featured vectors. I used prediction accuracy as the final performance metric. These set of parameters are trained The performance on the validation set is 85.00%, and the test set accuracy is 83.44%. Table 6 below is a comparison with the baseline model. We can see that in the test set, the prediction accuracy has improved by 1.56% compared with the non-trivial baseline model.

Table 6: Final Model Comparison

ML Method	Val Acc.	Test Acc.
Trivial	49.69%	49.13%
Non-trival	82.87%	82.19%
Random Forest	86.12%	83.00%

I also created the word cloud Figure 7 in the dataset based on the feature importance in the Random Forest model. This image is a comprehensive visualization of the top 20 important features in the Amazon Review Dataset. The size of each word indicates its feature importance. Some common positive words (great, excellent, best, perfect) are bold and big, while some obvious negative sentiment words (waste, bad, boring) are included in this picture as well.

Figure 7: Word Cloud with Random Forest Feature Importance

4.2 Interpretation

The best model is Random Forest as I expected. The most intuitive way of classifying product reviews is using Decision trees. When we see a review, if any word in the word cloud above appears in the review, we can distinguish whether it is a positive or negative review immediately. The decision

tree does the same work and the random forest is an ensembled version of the decision tree. What is unexpected is that the BOW (Count Vectorization) features outperform the TF-IDF features. What I thought was the TF-IDF features contain more information such as the frequency and importance. With deeper investigation, I understood that the TF-IDF features may be suitable for regression and linear training methods, such as Logistic Regression and SVM, and not for tree-based models. This conclusion is confirmed by Table 5 result. Another discovery is that the XGBoost model doesn't perform as I expected. One possible reason is the overfitting problem. Since it performs well in the training set with nearly 100% prediction accuracy, the low accuracy in the validation set indicates it might need more parameter selection to avoid overfitting or it is not suitable for this problem.

There are several parts that can be improved further.

- The Random Forest model is still having the overfitting problem with high accuracy in the training set. We can further simplify the model by more feature selection and more parameter tuning steps. I didn't implement it further with the time restriction.
- Another method I want to implement is the n-gram model. We can calculate the conditional probability of a word appearing after n other words. I didn't apply it here since the dataset we use is "*fake sentence*" - the words in the sentence are not ordered as human speaking order, so the effect or n-gram feature might be not effective.

5 Implementation for Transfer Learning

5.1 Dataset

In the Transfer Learning section, the dataset is the same as the Supervised Learning section, which comes from the "Multi-Domain Sentiment Dataset (version 2.0) "[1]. This dataset contains the product reviews of books, DVDs, electronics, and kitchens. For the TL section, I only used the books and DVDs domain. I treated the books as the source domain and the DVDs as the target domain. Details are stated in the following section.

5.2 Dataset Methodology

The problem assumption for the Transfer Learning (TL) problem is different from the Supervised Learning section. The assumption is that we have a small sample size (200) of labeled review data for DVDs products, which is the target domain. If only using the DVDs dataset to predict itself, the result might be not generalizable and performs not as expected. Thus, we introduce the TL method. By learning the pattern from a large dataset (2000 samples) of reviews from books as the source domain, we can transfer the model learned to the target domain dataset. We predict the small dataset using the model with both Subspace Alignment and TrAdaBoost. Figure 8 is the dataset flow chart for subspace alignment. The TrAdaBoost only used the labeled data, so I didn't draw the dataset flow chart here.

Let us define some convention symbols for different domains.

- D_s is the source domain including books reviews. It contains labeled source domain D_{sl} and unlabeled source domain D_{sul}.
- D_t is the target domain including DVDs reviews. It contains labeled target domain D_{tl} and unlabeled target domain D_{tul}.

For the D_{tl} domain, I only extract 200 samples from the original 2000 samples, since I assume I don't have enough samples for effective model training. And based on the result from the paper "Boosting for transfer learning[7]", the effect of transfer learning is significant when the target domain sample size is less than 10% of the source domain. I summarized the number of samples in each domain in Table 7 below.

When measuring the model performance, I only care the performance on the target domain, so I do the train test split on the target domain. The target labeled dataset are split into training D_{tl_train}, validation D_{tl_val}, and test set D_{tl_test}. The proportion for each set is 60%, 20%, and 20%.

Figure 8: Transfer Learning Flow Chart

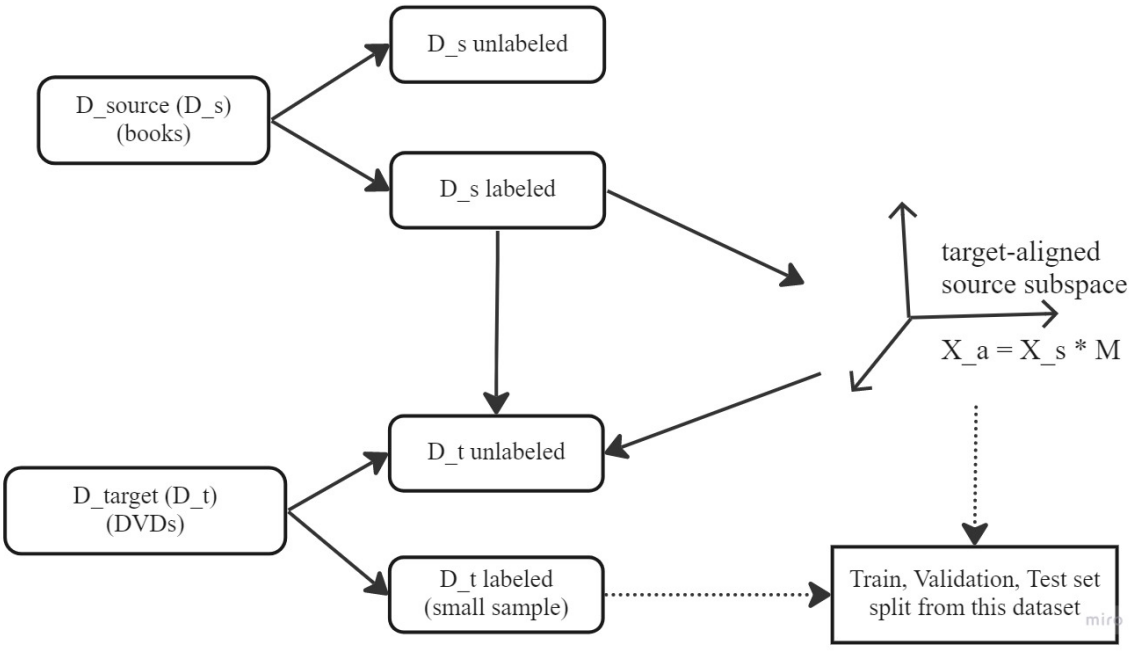


Table 7: Transfer Learning Dataset Size

Domain	Sample size
D_{sl}	2000
D_{sul}	4465
D_{tl}	200
D_{tul}	3586

5.3 Preprocessing, Feature Extraction, Dimensionality Adjustment

During the featurization (word embedding) step, I selected the **Bag of Words** (CountVectorizer) methods, since this method is the most effective shown in the supervised learning section. I also convert the feature space of each method to contain only the top 1000 and 5000 frequent features (words). After the word embedding step, it is important to *standardize* the dataset in the Subspace Alignment method. Thus, I standardized the input features of X_{sl}, X_{tul}, and the training set of X_{tl}, and use the mean and standard deviation of X_{tl} to standardize the D_{tl} testing set for SA method. For the TrAdaBoost method, I used the non-standardized dataset. Table 8 is the summary of all techniques we used.

Table 8: Featurization in TL Models

	Subspace Alignment		TrAdaboost	
Word-Embedding	CountVectorizer		CountVectorizer	
Standardize	Yes		No	
# of features	1000	5000	1000	5000

5.4 Training Process

In the training process, for both the SA method and the TrAdaBoost method, I trained the model with both 1000 features and 5000 feature versions.

5.4.1 Baseline

Since our main goal is to predict the DVDs' reviews sentiments, I only use the DVDs' reviews (Target Domain Labeled dataset) as the train, validation, and test set in the baseline model, without any other domain's support. The baseline model for the TL section is the Random Forest Model with n_estimators = 400 and max_depth = 70 and uses the entropy criterion (the best ML model in the Supervised Learning Section). I fitted the RF model to the target domain training dataset and evaluated the performance on the validation and test set. The overall performance on summarized in Table 9 below.

Table 9: Transfer Learning Baseline Result

ML Model	# features	Val Acc.	Test Acc.
Random Forest	1000	80.00%	64.00%
Random Forest	5000	68.00%	60.00%

We can see the baseline model achieves an accuracy score of 80% on the validation set and 64% accuracy on the test set for 1000 features. With 5000 features, the validation dataset accuracy is 68% and 60% in the test set. I will treat the baseline separately for 1000 features' result and 5000 features' result in the following sections.

5.4.2 Subspace Alignment

In subspace alignment, I used the books dataset as the source domain, which consisted of unlabeled and labeled datasets. I also used DVD reviews as the target domain, which consists of labeled and unlabeled datasets. In order to calculate the similarity matrix between the source domain and the target domain, I used labeled data in the source domain and unlabeled data in the target domain to get this matrix. Then I multiplied this projection matrix with features of the source domain to get new features X_a. Finally, I applied X_a to train labeled data in the target domain using Random Forest. This target domain dataset was also separated into validation and test dataset. Finally, I compared the accuracy of each dataset and applied the best parameters to get the test result shown in Table 10 and Table 11.

Table 10: Subspace Alignment Model Summary (1000 features)

Estimator	PCA Dimension	Val Acc.
Random Forest	80	68.0 %
Random Forest	100	68.0 %

Table 11: Subspace Alignment Model Summary (5000 features)

Estimator	PCA Dimension	Val Acc.
Random Forest	80	48.0 %
Random Forest	100	48.0 %

From the result above, we can see that the Subspace Alignment method didn't perform well (48% accuracy) and was even worse than the baseline model (68% accuracy) in the 5000 features space. It's similar to the 1000 feature space dataset. One reason is that the Subspace Alignment is assuming we want to transfer some feature representations from the source to the target domain.

However, this assumption cannot hold in my case, since the source and target domain features are similar contents (product reviews). The result indicates that the subspace alignment method might not be suitable in this case.

5.4.3 TrAdaBoost

The main issue is the insufficient data sample in the target domain. Thus, this might be an instance-based transfer learning problem. The target domain should learn some extra knowledge from the source domain. That is the reason why I chose the TrAdaBoost method. For the TrAdaBoost model, I used the random forest model as the base estimator. The parameters I trained is n_estimators which implies the number of boosting iterations and I tried different learning rates as well. All the result is presented in Table 12 and Table 13.

Table 12: TrAdaBoost Model Summary (1000 features)

Estimator	$n\_estimators$	learning rate	Val Acc.
Random Forest	8	0.1	76.0 %
Random Forest	10	0.1	72.0 %
Random Forest	8	0.5	72.0 %
Random Forest	10	0.5	72.0 %

Table 13: TrAdaBoost Model Summary (5000 features)

Estimator	$n\_estimators$	learning rate	Val Acc.
Random Forest	8	0.1	76.0 %
Random Forest	10	0.1	80.0 %
Random Forest	8	0.5	80.0 %
Random Forest	10	0.5	84.0 %

From the table above, we can see that the best parameter setting for the TrAdaBoost model is $n\_estimators = 10$ and learning rate = 0.5, with 5000 features. It results in 84% accuracy which is better than the baseline model.

5.5 Model Selection and Comparison of Results

The summary Table 14 below is an overview of the parameter, feature selection, and metrics performance for each model. In parameter selection for each model, the listed parameters are the optimal version in the training process. Other parameters that are not listed on the table use the default setting in *sklearn* library. The performance metrics I used is accuracy. The entire target domain of 200 reviews is divided into a training set of size 120 (60%), a validation set of size 40 (20%), and a test set of size 40 (20%). Both validation set and test set accuracy is listed below.

Table 14: Transfer Learning Model Selection

TL Method	Estimator	# features	Param Selection	Val Acc.	Test Acc.
None (Baseline)	RF	1000	–	80%	64%
Subspace Alignment	RF	1000	PCA dimension = 80	68%	55%
TrAdaBoost	RF	5000	Boost iter. = 10, lr = 0.5	84%	72%

From the result above, I conclude that the TrAdaBoost method performance is the best among all three models.

6 Final Results and Interpretation for the extension

6.1 Final Results

The best result from the transfer learning comes from the TrAdaBoost method. The TrAdaBoost method I used has 10 boosting iterations and the learning rate equals 0.5. The best feature dimension is 5000. With these settings, the performance on the validation set is 84% and the test set accuracy is 72%. Table 15 below is a summary table for the baseline and TrAdaBoost model. We can conclude that the TrAdaBoost Model improved the baseline performance by 8% which is significant.

Table 15: Transfer Learning Final Model Summary

TL Method	Estimator	Val Acc.	Test Acc.
None (Baseline)	RF	68%	64%
TrAdaBoost	RF	76%	76%

6.2 Interpretation

It's interesting to see that the TrAdaBoost method performs well with a large feature space than the Subspace Alignment method. The reason might be that the boosting iterations can fully utilize the features, while, the Subspace Alignment method does not require large feature space since the PCA dimension reduction process will reduce the feature space eventually. This transfer learning study provides me with much new information and knowledge. With the variety of types of transfer learning problems, we need to find a suitable transfer learning method for different problems. In my case, the problem is an instance-based transfer learning problem with different X distributions in books and DVDs products reviews. Both books and DVDs are subcategories of product reviews. Their dataset distribution is similar to each other. Thus, the TrAdaBoost method can learn more information than the Subspace Alignment method.

7 Summary and conclusions

In conclusion, the prediction of positive or negative sentiment on product reviews is successful with good performance accuracy. I think there is still space for performance improvement if I can get more information from the dataset. For example, with the raw reviews dataset that contains complete sentences, I can use the n-gram featurization method to get the relationship in a sentence, etc. Also, I can dig into this problem with a 5-stars rating dataset. This model can be applied to many sentiment prediction problems, such as the sentiment of Youtube video comments, and customer support ticket analysis, and it is a long journey for us to perfect the model and apply it to more circumstances.

References

[1] John Blitzer, Mark Dredze, Fernando Pereira. Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification. Association of Computational Linguistics (ACL), 2007.

[2] Kane, Frank. "XGBoost: How It Works, with an Example." YouTube, 11-Feb-2020. [Online] Available: www.youtube.com/watch?v=OQKQHNCVf5k.[Accessed: 20-Dec-2021]

[3] Prabhu, "Understanding NLP word embeddings-text vectorization," Medium, 21-Nov-2019. [Online]. Available: https://towardsdatascience.com/understanding-nlp-word-embeddings-text-vectorization-1a23744f7223. [Accessed: 04-Dec-2022].

[4] S. D. N, "Amazon fine food reviews featurization with Natural Language Processing," Medium, 01-Dec-2020. [Online]. Available: https://medium.com/analytics-vidhya/amazon-fine-food-reviews-featurization-with-natural-language-processing-a386b0317f56. [Accessed: 04-Dec-2022].

[5] V. Karbhari, "What is TF-IDF in feature engineering?," Medium, 26-Feb-2020. [Online]. Available: https://medium.com/acing-ai/what-is-tf-idf-in-feature-engineering-7f1ba81982bd. [Accessed: 04-Dec-2022].

[6] V. Reddy, "Sentiment analysis using SVM," Medium, 23-Jun-2020. [Online]. Available: https://medium.com/@vasista/sentiment-analysis-using-svm-338d418e3ff1. [Accessed: 05-Dec-2022].

[7] W Dai, Q. Yang, G. Xue, and Y. Yu. 2007. "Boosting for transfer learning". In Proceedings of the 24th international conference on Machine learning (ICML '07). Association for Computing Machinery, New York, NY, USA, 193–200. [Online]. Available: https://doi-org.libproxy2.usc.edu/10.1145/1273496.1273521 [Accessed: 20-Jun-2007]

[8] Y. Jiaming. "Python API reference" Python API Reference - xgboost 1.7.1 documentation. [Online]. Available: https://xgboost.readthedocs.io/en/stable/python/python_api.html#module-xgboost.sklearn. [Accessed: 06-Dec-2022].