Sentiment Analysis of Amazon Customer Reviews

Shreyas Surendra
shreyas.sf92@knights.ucf.edu

Ramya Kondapi
rkondapi@knights.ucf.edu

Anupama Sinha
anupama.sinha@knights.ucf.edu

Yaswanth Santhanam Raghavan
yaswanthsr@knights.ucf.edu

1. MOTIVATION AND PROBLEM STATEMENT

Customer reviews have become a significant aspect before purchasing a product from any e-commerce website. Collecting feedback on a product that a consumer has already purchased is important for making a product judgment. Analyzing consumer feedback and knowing each customer's preferences would also significantly assist product owners/sellers with more product improvements.

2. RELATED WORK

Research papers on sentiment analysis of text data such as twitter tweets, film reviews and product reviews are published. Machine learning approaches such as Naïve Bayes, Support Vector Machines, Maximum Entropy, Decision trees etc. accomplish sentiment analysis, a classification task. Previous work demonstrates how accuracy can be enhanced using different preprocessing strategies such as tokenization, lemmatization, stop word elimination, etc. We would like to pursue a similar approach to creating a model that better suits the Amazon customer reviews dataset.

3. SENTIMENT ANALYSIS OF AMAZON CUSTOMER REVIEWS

Using classification algorithms such as Naïve Bayes, Support Vector Machines, Random Forest and Neural Networks like Multilayer Perceptron and Long Short-Term Memory, customer review in the form of text would be categorized into two classes i.e. positive or negative. To use these algorithms, the data must be pre-processed, which involves punctuation removal, stopword removal, lemmatization, tokenization, vectorization, dealing with conjunction rules, negation handling and feature selection. After the data is pre-processed, the Naïve Bayes algorithm will be used as a benchmark to train and check the accuracy for our dataset. Based on the progress of the project and the time available we would also try to design a model using SVM, Random Forest, Multilayer Perceptron and LSTM to test if a better accuracy score can be achieved.

4. EVALUATION

For this project, Amazon Customer Reviews Dataset will be .used which is maintained by Amazon. The dataset can be found in AWS (s3.amazonaws.com/amazonpds). This dataset contains 14 variables, among which variables such as Review Body and Star Rating are necessary for this project. Sample record of the dataset is shown below.

Review_body	Star_rating
As advertised. Everything works perfectly, I'm very happy with the camera. As a matter of fact, I'm going to buy another one for my 2nd car	5
Poor quality and low sound output for its size. A real disappointment.	1

Table 1: Sample attributes

There are many measurement criteria for evaluating the accuracy and consistency of the machine learning model. As the problem statement discussed in this project is related to classification, the most widely used metrics such as accuracy, precision, recall and F1- scores will be used to determine the model's output.

5. PREPROCESSING THE DATASET

The amazon customer review dataset needs to be preprocessed initially so that a classification algorithm can be used in order to classify the reviews into positive/negative data. Since the focus is on Mobile Electronics, we filtered the reviews using "Product title" attribute in the dataset which indicates what kind of an electronic device it is (For example: Mobile, headphones, cables, speakers, etc..). The features from the dataset that this project will focus on Product_title, Star_rating, Review_headline, Review_body.

The preprocessing pipeline of our project is shown below. In the first stage noisy entities of a text, i.e. punctuation and stopword, are removed from the review text. In the second stage the words are tokenized. The tokens are lemmatized to eliminate similar words from the vocabulary. In the final stage the tokens are vectorized.

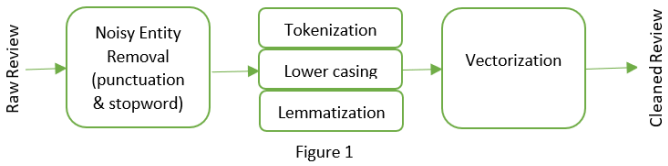

Fig 1: Pipeline

Vectorization: The process of encoding text as integers to create feature vectors, n-dimensional vectors of numeric features. There are multiple vectorization techniques, in this project, techniques like Count Vectorization, N-gram Vectorization, TF-IDF vectorization and Dense word vectors like GloVe are used.

Count Vectorizer: It creates a document-term matrix where there is one row per review and columns represent unique words or terms. Each cell in the matrix represents the frequency of occurrences of terms. For example, let us suppose that we have following three reviews- "Great Product", "Great Deal", "Great Experience", matrix of size 3 x 4 is generated. This is because we have three reviews and four different features.

Review	Great	Product	Deal	Experience
Great Product	1	1	0	0
Great Deal	1	0	1	0
Great Experience	1	0	0	1

Table 2: Count vectorization example

N-Gram Vectorizer: It creates a document-term matrix where there is one row per review and each cell represents the frequency. But, instead of columns representing single term, they represent N combination of adjacent words in the review. For the above example, the features would be Great, Deal, Great Product, Great Deal etc.

TF-IDF Vectorizer: It creates a document-term matrix where there is one row per review and columns represent unique words or terms. Each cell in the matrix represents a weighting that represents how important a word is to individual review. The weight is almost proportional to the frequency of word in a document. For e.g., let us suppose that a review in mobile electronics dataset contains 80 words where the word Great appears 4 times. The TF (Term Frequency) for Great will be (4/80) = 0.05. Again, suppose that there are 106 million reviews in the entire corpus and the word great appears 103 times in the corpus. The IDF (Inverse Document Frequency) will be log (106/103) = 3. Hence, TF-IDF will be 0.05*3 = 0.15.

GloVe: GloVe is dense vector representation of words. It combines the benefits of the Word2Vec skip-gram model which exploits local statistical information and Matrix Factorization which exploits global statistical information. Using GloVe method, we can obtain sematic relationships between words from the co-occurrence matrix. Suppose we are given a corpus containing V words, the co-occurrence matrix X will be a V x X matrix. X_ij represents how many times word i has occurred with word j, where i and j denotes the row and the column respectively. Consider the probability entity P_ik/P_jk where P_ik = X_ik/X_i. Here P_ik represents the probability of seeing words i and k together, which is given by the number of times words i and k appeared together (X_ik) divided by the total number of times word i appeared in the corpus (X_i). By incorporating P_ik/P_jk to computing words, the goal of exploiting global statistics is achieved when the word vectors are learned.

In this project, pretrained word vectors are used. Specifically, GloVe word embedding of 100 dimension is used which is available on Stanford's official website (link : https://nlp.stanford.edu/projects/glove/)

5.1 Method A

In the next step, the attributes "Review headline" and "Review body" is preprocessed using Spacy. The preprocessing steps includes Sentence Segmentation, Tokenization, lowercasing, lemmatization and stop word removals which is being performed using Spacy library. With the help of preprocessing, all the review data attribute is converted to "Tokens" (words that appear in that product review) and these words can be compared with a word database like "SenticNet" which provides a polarity between positive and negative meanings of words.

Features in Sentient Word Database	Details
Number of unique word concept	1,00,000
Polarities	Positive, Negative
Intensity range	-1 (extremely Negative) to 1 (extremely positive)

Table 3: Sentinet word database details

Now, the reviews in the dataset is split into positive and negative reviews by comparing the reviews with the word databases. This split is achieved by comparing each word in the preprocessed sentence of a product review with the SenticNet word database and each comparison gives us a value between -1 to 1, which helps us identify if a review has a lot of positive or negative words and the intensity of the words used in the review(for example: as previously stated "-1" indicating extreme negativity). Now, after obtaining this review type of the attribute "Review body", the review types which has the value "0" (i.e. either the review body is empty or the sentient does not contain any words used in the review) are compared with the "Star Ratings" to identify the type of review since they also tend to indicate if it's a good (or) bad product. Now, the review type will be used as the Class (positive/ Negative) for this dataset.

A simple example of how a Review body's class is obtained using Sentinet:

Input Sentence: Harry Potter is a fabulous book.

Output words after Spacy preprocessing: Harry, potter, fabulous (3 words)

Now, these three words are compared with Sentinet and the score obtained for this sentence is: "0.793" for the word "fabulous". Since 0.793 is greater than zero. The class

"Positive" is assigned for this review. A similar process will be followed for class "Negative" if the score of a sentence is less than zero.
Based on the given example the output from spacy and Sentinet score can be summarized as:

Review_body	Sentinet Score	Class
Harry, potter, fabulous	0.793	Positive

Table 4: Sample data after Spacy preprocessing

In case, if a sentence has a score "zero", then the star rating of the sentence is taken into consideration to determine the Class label.
After creating the classes for each review in this dataset, the data set is stratified based on the class type of the reviews. This stratification is done to ensure that the training and test set contains approximately the same percentage of samples from each target classes. After stratification, the dataset is split into training and testing data in the ratio of 80:20.
In order to vectorize the data, Count vectorizer and TfIdf Vectorizer were used separately and the training and testing set is vectorized. After vectorization, the ML algorithms are trained in order to predict the test set data and calculate the accuracy of the predicted test set.

5.2 Method B

In the first method, preprocessing using SpaCy library, it was observed that a lot of words, which we thought was important in terms of representing the sentiment of the customer, were being removed in second stage of the preprocessing pipeline. We also observed that word removal was not consistent, and it was dependent on the sentence. To have little more control on the kind of words that could be used for sentiment analysis, we also performed preprocessing using NLTK library.
NLTK's lemmatization was found to be consistent and sentiment bearing words were retained. The stopword list of NLTK was also updated to remove negation words such as NOT, DON'T, etc. from the list. The number of unique words, i.e. the vocabulary size, increased significantly compared to vocabulary size obtained using SpaCy. In order to reduce the number of words, we performed exploratory data analysis to gain insights about the type of words and their frequency. From the analysis it was found that a lot of misspelled, and words of no use were retained. Another interesting observation was that most of the words that appeared in product title also appeared in the review. It is quite natural to use the product title in the review. For example, the word 'macbook' appears twice in the review "tried pair time macbook pro kept disconnecting connecting continuously wasn't usable tried like look nice price usable macbook". To remove product title from the review another block was added in the preprocessing pipeline that removed a list of words that appeared in the product title. The word frequency threshold was set to 20 (eliminated words whose frequency was less than 20).
In this method the vocabulary size was reduced to 3700 words. After preprocessing, the dataset consists of two features Review_body (cleaned data) and Star_rating. The label column, named Class, is generated using Star_rating, if the rating is 3 and above the review is tagged as positive, if the rating is below 3 the review is tagged as negative.
The preprocessed dataset is shown below.

Review_body	Star_rating	Class
['advertised', 'work', 'perfectly', 'very', 'happy', 'matter', 'fact', 'going', 'buy', 'another']	5	1
['poor', 'disappointment']	1	0

Table 5: Sample data after NLTK preprocessing

6. MACHINE LEARNING ALGORITHMS

6.1 Naïve Bayes

Naïve Bayes Classifier is a Machine learning classification algorithm that is widely used in NLP tasks. It helps to classify the most probable Class (positive/ negative review) given an input data. Mathematically Naïve Bayes can be represented as, $P(C/X) = P(X/C)P(C)\ P(X)$ Here, P(C/X) represented the probability of "X" belonging to class "C". In this work, "X" will represent our customer's "review body" attribute words and "C" will represent either Positive/Negative class. P(X/C) is the likelihood value which represents the probability of "X" given "C" and this a normal distribution P(C) represents the Prior value of the class. And P(X) can be dropped because we need to maximize the value of the equation when we use "Argmax" function. Therefore, after calculating the P(C/X), we need to choose the value of the Class that maximizes the probability value, and this is achieved using the above mentioned Argmax function. During this process there are high chances of incurring sparsity problem. But this can be managed by smoothing technique, which involves adding a small constant value to all the probabilities. In training phase, the preprocess review body is given as input to the classifier and Naïve Bayes calculates the likelihood value and Prior value of class. For this dataset, prior value is obtained by dividing "total number of reviews in the dataset" by "Number of reviews belonging to each class". The likelihood value is obtained by dividing the "count of occurrence of that particular word in the class" by "count of total number of words in that class (positive/ negative)". After training the data, we try to classify the test data by calculating the prior and likelihood values and the probability value for each class (positive or negative) is obtained and the class that has the maximum probability value is chosen to be correct class. Finally, the accuracy results can be calculated based on the obtained results from test set results.

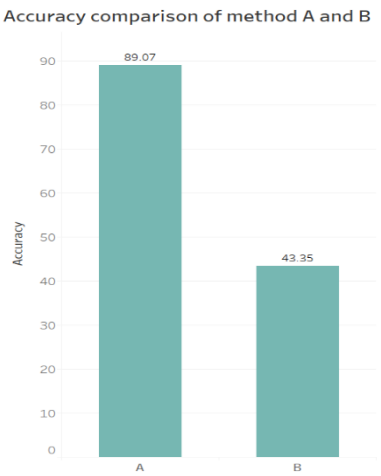

Fig 2 : Results obtained from Naïve Bayes algorithm

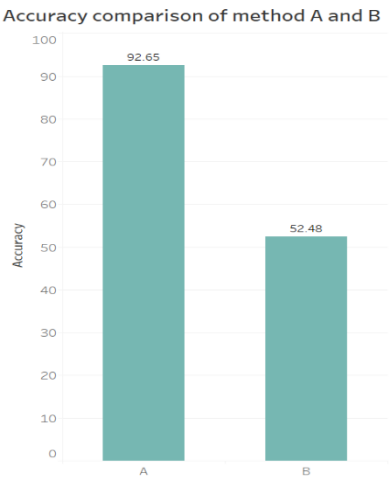

Fig 3: Results obtained from SVM algorithm

6.2. Support Vector Machine

SVM is one of those techniques which finds the finest hyperplane that separates the data between two classes "positive" and "negative". Each side of the plane has different class and we can easily classify any new data point based on which class it belongs to. For making predictions that are accurate, SVM plays an important role. Constructing SVM can be divided into two phases: In the first phase we train our machine by providing a classified dataset. In the second phase, we will be using our trained model to make predictions about the classification of the new data. The most difficult part here is to find the best hyperplane that separates the data. After completing the preprocessing stage, the feature words from the reviews are converted to numerical vectors to represent each review as a numerical data.

Next, these numerical vectors are provided as an input to SVM. SVM then defines precision boundaries by hyperplanes by plotting all the numeric vectors in space. This hyperplane divides the vectors into two different categories in a way that the distance from each category's closest point to the hyperplane is maximum. After the model is trained, the number of positive and negative reviews that are predicted correctly and the number of positive and negative reviews that are predicted incorrectly are shown by the confusion matrix. Prediction accuracy is calculated on each fold using the confusion matrix and the final accuracy is calculated by taking the mean of all separate accuracies of 10 folds. Performance evaluation parameters like values of precision, recall and F-measures are evaluated. We implement K-fold cross validation SVM algorithm where single fold is examined for testing and the rest folds are examined for training.

6.3. Random Forest

A Random Forest is a type of an ensemble classifier that estimates based on the combination of different decision trees. We can use random forests in case for both classification and regression. The more the number of decision trees, the more robust a random forest is. It makes decision trees based on randomly collected data samples and gets prediction from each decision tree and then selects the best solution. In our implementation of random forest, the number of trees in forest i.e., n_estimators is 10, which is a default number. We have taken max_depth=100 which is the number of levels in each decision tree.

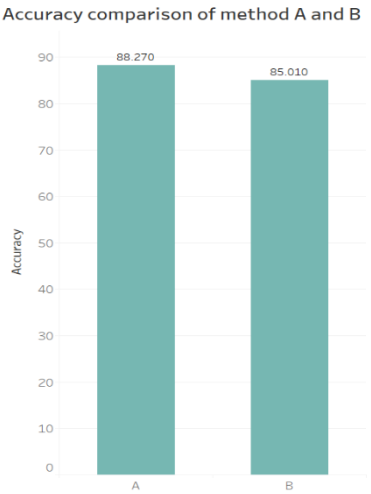

Fig. 4: Results obtained from Random Forest algorithm

7. NEURAL NETWORKS:

7.1 Artificial Neural Networks (ANN)

ANN is Machine Learning Neural Net method and has multiple layers which are fully connected. This network contains an input layer, multiple hidden layer and Output layer. Every node in a layer is connected to every node I the next layer.

For Method 1, ANN was used in order to evaluate the accuracy of the model and the results were obtained for various certain parameters in this model. The activation function used in input and hidden layers are "relu" activation and the output layer contains a "sigmoid" activation function.
The accuracies are obtained for "nb_epoch" parameter values 10 and 20.
For "nb_epoch" parameter value 10, three different results were obtained by ranging the number of hidden layers between two to four and in each of these hidden layers, 10 neurons were used. Now for the same "nb_epoch" value, 20 neurons were used instead of 10 and the results were obtained.
Now the "nb_epoch" parameter is updated to 20 and the same evaluation was performed.
The highest accuracy obtained for ANN was from 2 Hidden layers, 20 Epochs, 100 Neurons.

7.2 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network capable of learning order dependence in sequence prediction problems. Since it maintains order dependence it can capture negation words and how they impact the sentence meaning, hence it is a good choice for sentiment classification task.

7.2.1. Word Embedding

The word vectors need to be developed so that the vectors can be passed to the LSTM NN as the input. We decided to use pretrained word embeddings for our model. There were two options Word2Vec and GloVe embeddings. GloVe embeddings were chosen because of the size feasibility; particularly GloVe 100-dimension word embeddings were chosen.
The word embeddings were downloaded, from Stanford's official website, as a text file. The text file was loaded as a dictionary with key as the word and the value as the 100-dimension embedding of the word. Using this dictionary of embeddings, an embedding layer of our vocabulary was created. Method B's vocabulary was selected for this model. Since our vocabulary size is 3700 and the embedding dimension is 100, by
restricting the input sequence size to 100 words we obtained an embedding layer of dimension 3700 x 100.

7.2.2. LSTM Architecture

The NN is implemented using TensorFlow and Keras library. Keras sequential model was used to build the NN. Tensor of dimension 3700 x 100 is used at the embedding layer, Followed by a layer of LSTM cell with 250 units. The final layer is a single neuron sigmoid classifier. Loss function used is cross entropy with Adam optimizer. The output is a scalar value between 0 to 1, if the value is equal to or greater than 0.5 the review is classified as positive, The value is less than 0.5 the review is classified as negative. The model was trained for 15 epochs.
The metric we used to evaluate this model is accuracy. The accuracy of 86.80 is achieved. Tested the model with a couple of review including negation words and observed
that the model handled negation words in the sentence very well. Review "I like it" was classified as positive and review "I do not like it" was classified as negative.

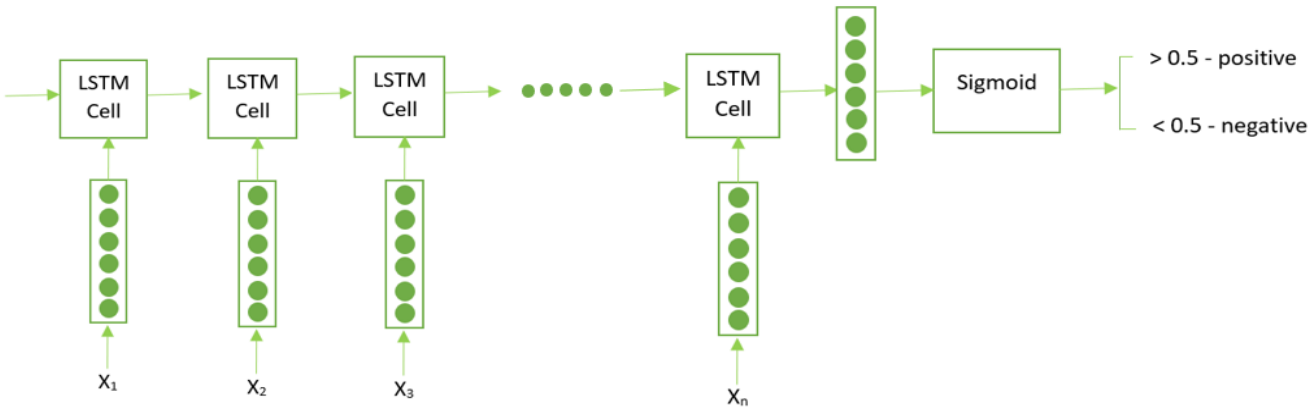

Fig 5. LSTM Architecture

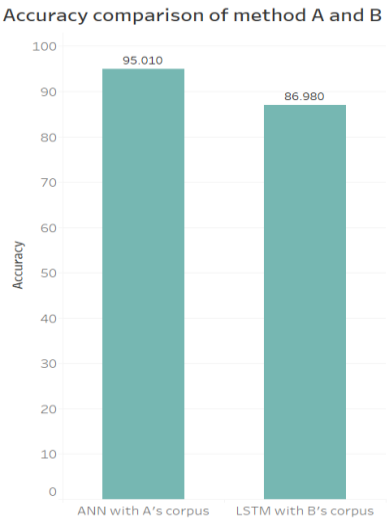

Fig 6: Results obtained from ANN (method A) and LSTM (method B

8. CONCLUSION

Based on the above obtained results, for this Amazon Customer Reviews Dataset, Spacy was better since it gave a reduced vocabulary size after preprocessing and producing good accuracy results. NLTK was better when handling sentiment bearing words which has negation contents in it. Best accuracy was achieved using ANN with spacy preprocessing at 95.01%. But taking into considering of analyzing negation bearing words, the best combination was using LSTM after NLTK preprocessing.

Note: Link to the Source code:
https://github.com/sryaswanth/NLP-Project

9. PLAN AND ROLE OF COLLABORATORS

Shreyas Surendra - Data Extraction and Preprocessing (30%), Model Implementation (50%), Documentation (20%).
Yaswanth Santhanam Raghavan - Data Extraction and Preprocessing (40%), Model Implementation (30%), Documentation (30%).
Ramya Kondapi - Data Extraction and Preprocessing (40%), Model Implementation (30%), Documentation (30%).
Anupama Sinha - Data Extraction and Preprocessing (50%), Model Implementation (20%), Documentation (30%).

10. REFERENCES

[1] Dataset: https://s3.amazonaws.com/amazon-reviewspds/tsv/amazon_reviews_us_Mobile_Electronics_v1_00.tsv.gz
[2]https://medium.com/@datamonsters/sentiment-analysis-tools-overview-part-1-positive-and-negative-words-databases-ae35431a470c
[3] https://en.wikipedia.org/wiki/SpaCy
[4]https://medium.com/datadriveninvestor/implementing-naive-bayes-for-sentiment-analysis-in-python-951fa8dcd928
[5] https://becominghuman.ai/naive-bayes-theorem-d8854a41ea08
[6] http://cs229.stanford.edu/proj2018/report/122.pdf
[7] https://towardsdatascience.com/creating-the-twitter-sentiment-analysis-program-in-python-with-naive-bayes-classification-672e5589a7ed
[8] https://sentic.net/
[9] https://machinelearningmastery.com/time-series-prediction-lstm-recurrent-neural-networks-python-keras/
[10] https://towardsdatascience.com/sentiment-analysis-using-rnns-lstm-60871fa6aeba
[11] https://www.kaggle.com/ngyptr/lstm-sentiment-analysis-keras
[12] https://towardsdatascience.com/sentiment-analysis-using-lstm-step-by-step-50d074f09948
[13] https://core.ac.uk/download/pdf/82667502.pdf
[14]https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html
[15]https://towardsdatascience.com/random-forest-in-python-24d0893d51c0
[16] https://sentic.net/
[17] https://nlp.stanford.edu/projects/glove/