Sentiment Analysis of Amazon Alexa reviews
Tejesh Agrawal Anila Mathews Shriniket Revadekar

Introduction:

Sentiment analysis is the process of using natural language processing, text analysis, and statistics to analyze customer sentiment. In this project we are analysing Amazon Alexa Echo, Echo dot and Echo plus reviews to calculate positive and negative polarity.

Motivation:

Movie recommendation in the mobile environment is critically important for mobile users. It carries out comprehensive aggregation of user's preferences, reviews, and emotions to help them find suitable movies conveniently. Using sentiment analysis this can be achieved and it is much efficient way to do than other ways.

Technical Approach:

To achieve the sentiment analysis, we used below datasets for Alexa reviews from kaggle.com
https://www.kaggle.com/sid321axn/amazon-alexa-reviews

We used pandas for file reading, data manipulation and analysis. NLTK (The Natural Language Toolkit),Spacy for symbolic and statistical natural language processing for English written in the Python programming language. Scikit learn for model selection and training the data. Pattern for performing tasks such as tokenization, stemming, POS tagging, sentiment analysis. The pickle module implements binary protocols for serializing and de-serializing a Python object structure.

Analysis:

As Alexa echo plus having the highest price and echo dot as lowest, and the range is $50 to $150. As preliminary analysis gives us the idea that positive reviews are greater than negative reviews.

Data cleaning and rearranging:

We are using pandas for arranging and putting in data framing.

```
alexa_df = pd.read_csv('amazon_alexa.csv', sep = '\t')
alexa_df.head()
```

Then we normalized data,

```
alexa_df['rating'].value_counts(normalize=True)
```

To clean data, and separating new column for reviews:

```
alphanumeric = lambda x: re.sub('\w*\d\w*', ' ', x)
punc_lower = lambda x: re.sub('[%s]' % re.escape(string.punctuation), ' ',
x.lower())
alexa_df['reviews'] =
alexa_df['verified_reviews'].map(alphanumeric).map(punc_lower)
stop = set(stopwords.words('english'))
stop_words = ['-PRON-','s','be','"','""','''','u','--','','get','would' ]
```

```
def remove_stops(comment):
      stop_free = [i for i in comment.split() if i not in stop]
      got_stop_free = " ".join([i for i in stop_free if i not in
stop_words])
      return got_stop_free
alexa_df['reviews']= alexa_df['reviews'].apply(remove_stops)
```

Spacy is used for natural language processing.

```
nlp = spacy.load('en', disable=['parser', 'ner'])
def space(comment):
      doc = nlp(comment)
return " ".join([token.lemma_ for token in doc])
alexa_df['reviews']= alexa_df['reviews'].apply(space)
```

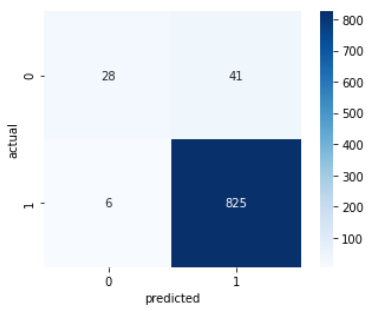
Using pickle we implement binary protocols for serializing and de-serializing a Python object structure. "Pickling" is the process whereby a Python object hierarchy is converted into a byte stream,

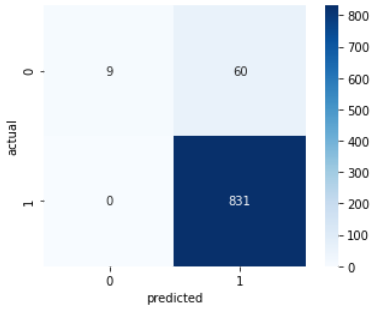
```
pickle.dump(alexa_df, open("alexa_cleaned.pkl", "wb" ))
```

Sentiment Analysis of Amazon Alexa reviews
Tejesh Agrawal Anila Mathews Shriniket Revadekar

Now to extract features, training and regression for prediction:
CountVectorizer is to convert a collection of text documents to a matrix of token counts

```
cv = CountVectorizer(max_df=0.90, min_df=2, max_features=1000)
X = cv.fit_transform(clean_corpus).toarray()
y = alexa_df['feedback'].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
log = LogisticRegression()
log.fit(X_train, y_train)
```





Train the LDA model

LDA: the latent Dirichlet allocation (LDA) is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar. For example, if observations are words collected into documents, it posits that each document is a mixture of a small number of topics and that each word's presence is attributable to one of the document's topics. LDA is an example of a topic model and belongs to the machine learning toolbox and in a wider sense to the artificial intelligence toolbox.

```
lda_model = LdaModel(corpus=corpus,
      id2word=dct,
      random_state=100,
      num_topics=3,
      passes=5,
      per_word_topics=True)
```

Sentiment Analysis of Amazon Alexa reviews

Tejesh Agrawal Anila Mathews Shriniket Revadekar

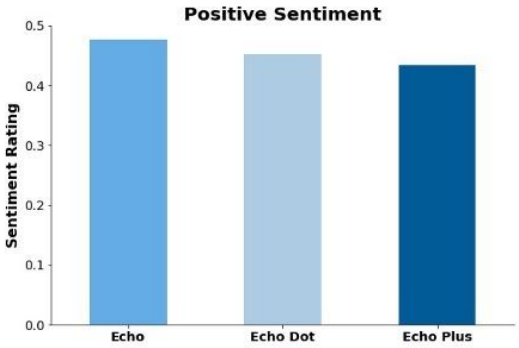
Word2Vec: Word2vec is a technique for natural language processing. The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence. As the name implies, word2vec represents each distinct word with a particular list of numbers called a vector. The vectors are chosen carefully such that a simple mathematical function (the cosine similarity between the vectors) indicates the level of semantic similarity between the words represented by those vectors.

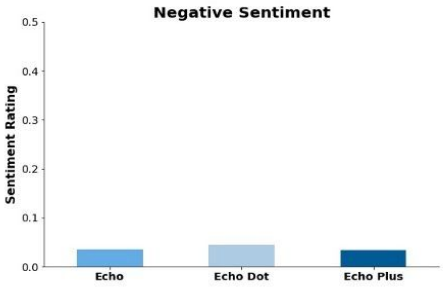
```
# Train Word2Vec model. Defaults result vector size = 100
model = Word2Vec(clean_corpus, min_count = 0, workers=cpu_count())
# Get the word vector for given word
Model['sound']
model.most_similar('sound')
```

Vader: VADER (Valence Aware Dictionary and sEntiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed in social media. VADER, a lexicon and rule-based sentiment analysis tool that is specifically attuned to sentiments expressed on social media.
We used VADER(Valence Aware Dictionary and sEntiment Reasoner) for sentiment analysis, and matched the rating with reviews.

```
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyser = SentimentIntensityAnalyzer()
def sentimentScore(sentences):
    analyzer = SentimentIntensityAnalyzer()
    results = []
    for sentence in sentences:
        vs = analyzer.polarity_scores(sentence)
        print(str(vs))
        results.append(vs)
    return results
```

Sentiment Analysis of Amazon Alexa reviews
Tejesh Agrawal Anila Mathews Shriniket Revadekar

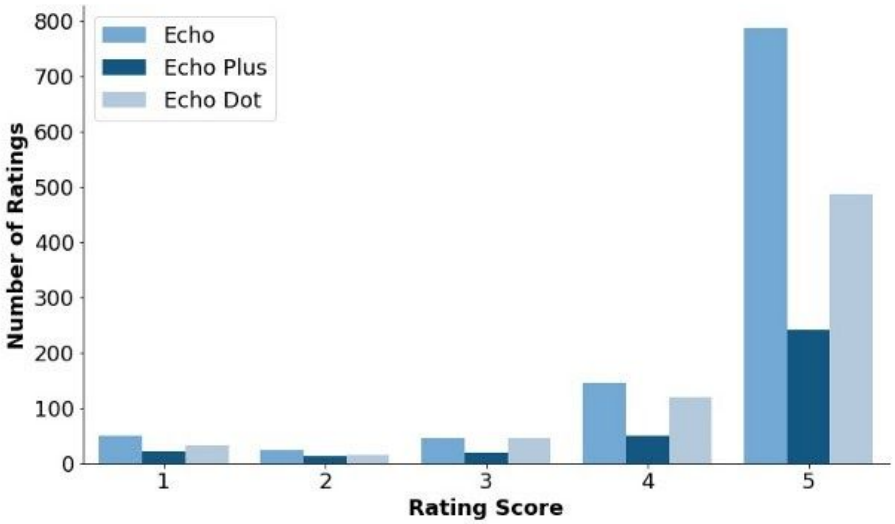




Github repository of working code:
https://github.com/TEJESH/Sentiment-Analysis

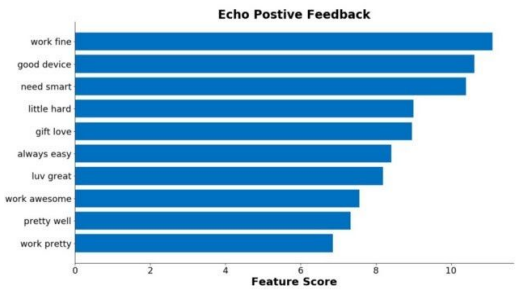
Tool used to compile:
Jupyter Notebook.

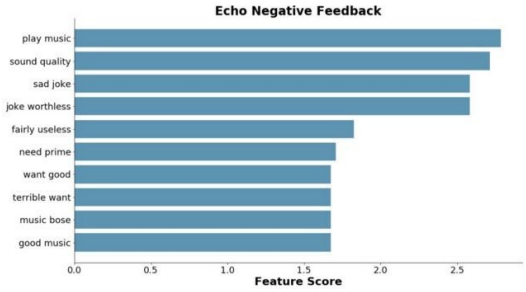
Results and Discussion:



Now, to find out which features were commonly reviewed, performing topic modelling using LDA (Latent Dirichlet Allocation). Using an iterative process, LDA maps documents to a distribution of topics. The distribution of words in the topics build up over the iterative process. The most common topics seem to be: users commenting on how much they love it, the ease of use and sound quality.

Using a count vectorizer (TFIDF), we analysed what the users love and hate about the Amazon Echo aka the words that contributed to positive and negative sentiments.





Conclusions and Future Work:

With this we conclude that sentiment analysis using the above tools and libraries are a much more efficient way to predict accurate and training models.

With this finding we can use the above analysis for systems such as recommendation systems, such as Netflix, Amazon etc.

We were planning to make our own recommendation system, and in future we will make a recommendation system using the above algorithms and libraries.

References:

https://www.kaggle.com/sid321axn/amazon-alexa-reviews
https://algorithmia.com/blog/introduction-sentiment-analysis
https://www.hindawi.com/journals/wcmc/2018/8263704/
https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-latent-dirichlet-allocation-437c81220158
https://medium.com/analytics-vidhya/using-nlp-on-amazon-echo-reviews-efb5078bf0d3