To compare and analyse the accuracy of BERT (Bi-Directional Embedding Representation) and SVM (Support Vector Machine) NLP machine learning algorithms for sentiment analysis of online reviews.

**Subject**: Computer Science  
**Topic**: NLP sentiment analysis using BERT and SVM algorithms  
**Word Count**: 4005

**Contents**

[1. Introduction 3](#_Toc93837844)

[2. Background Information 4](#_Toc93837845)

[2.1 BERT NLP Algorithm: Bidirectional encoder representations from transformers 4](#_Toc93837846)

[2.2 Support Vector Machine (SVM): 8](#_Toc93837847)

[3. Data Cleaning, and Experimental Methodology 10](#_Toc93837848)

[3.1 Data Cleaning Process for Text Analytics 10](#_Toc93837849)

[3.2 Experimental Procedure 12](#_Toc93837850)

[3.2.1 The Dataset Used 12](#_Toc93837851)

[3.2.2 Pre-processing 13](#_Toc93837852)

[3.2.3 Experiment Details: 13](#_Toc93837853)

[4. Experimental Results & Evaluation Matrices 15](#_Toc93837854)

[4.1 Evaluation Matrices Explanation: 15](#_Toc93837855)

[4.2 Evaluation Matric Final Output Result Tab 16](#_Toc93837856)

[5. Analysis and Conclusion 16](#_Toc93837857)

[5.1 Results Analysis 16](#_Toc93837858)

[6. References: 17](#_Toc93837859)

[7. Appendix: 17](#_Toc93837860)

# **Introduction**

In the rapidly growing field within data science NLP(Natural Language Processing) has led to specialized machine learning models that are developed to perform tasks including summarization, analysis, and classification of text. These Machine Learning models are applicable to a variety of situations. For example, NLP can be used by a company which wants to know more about its customers, for sentiment analysis to get valuable insights by summarizing and classifying customer reviews.

My research work is to compare the accuracy of NLP BERT (Bi-Directional Embedding Representation) and SVM (Support Vector Machine) algorithms for sentiment analysis of online electronics product reviews. For example, we can use NLP to detect the sentiments of customers whom may be dissatisfied with the selection of products even without having a person reading all the product reviews. Moreover, customers can describe their sentiments through words in innumerable ways such as sarcasm, which makes it more difficult to gain an accurate insight by just relying on humans to interpret the reviews or other traditional algorithms (such as word frequency analysis) instead of using NLP algorithms.[[1]](#footnote-1)

Sentiment analysis of online reviews for Electronic Products on a leading Ecommerce Platform with emphasis on Data Cleaning, Experiment Methods and Accuracy (F1 Score) comparison. Ecommerce has become a prominent platform for retail and other industries due to the COVID imposed lockdown. Due to a greater number of commodity reviews or comments generated every day on ecommerce sites, it is more meaningful to analyse these reviews and evaluate data using NLP sentiment analysis that is an automated process to analyse a text and interpret the sentiments behind it. This would greatly help customers to buy the desired goods, and can also be used by businesses as a reference to improve their services.

In this study, we carefully analyse and compare the accuracy of the word representation model BERT (Bidirectional Encoder Representations from Transformers) and SVM (Support Vector machine) models on Amazon e-commerce reviews dataset. BERT has been able to provide excellent results in innumerable NLP researches and uses a similar structure across the different tasks, adapting BERT instead of SVM to online customer reviews on their products could potentially benefit various sentiment analysis tasks.

To analyse this research question, a theoretical background of BERT and SVM classifier is provided. Further detailed explanation of BERT and SVM design is added. Finally, an experiment is conducted using experimental data from Amazon online reviews dataset which is used to compare the accuracy of sentiment analysis using BERT and SVM NLP machine learning algorithms. To ensure the validity of the results, cross-validation technique is used to ensure accuracy.

The paper is organized in sections including: Section 2 which has a summary of related work. The detailed experiment approach is described in Section 3. The results of the experiment are discussed in Section 4. The conclusion of the paper is in the final section 5.

# **Background Information**

|  |  |
| --- | --- |
| https://upload.wikimedia.org/wikipedia/commons/thumb/b/bb/AI-ML-DL.svg/800px-AI-ML-DL.svg.png | DL (Deep Learning) is part of a broader family of machine learning methods based on artificial neural networks with representation learning. Learning can be supervised, semi-supervised or unsupervised. Higher-Level features are extracted from multiple layers in a set manner |

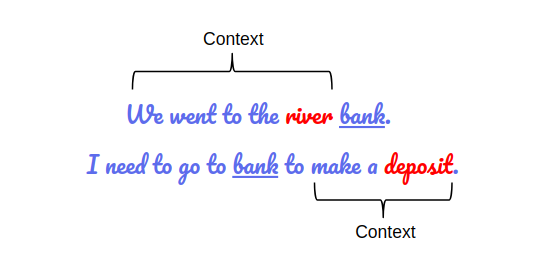
**Figure1 :** About Deep Learning

**Emerging Field of Natural Language Processing**

Natural language processing (NLP) is a part of computer science, artificial intelligence and linguistics that uses algorithms to understand, analyse, interpret and manipulate human language. NLP is one of the most broadly applied areas of machine learning and has been vital in analysing massive quantities of unstructured, text-heavy data effectively. NLP is used to build models that uncover contextual patterns, and produce insights from text as well as audio by analysing both speech and language.

## **2.1 BERT NLP Algorithm: Bidirectional encoder representations from transformers**

BERT stands for Bidirectional Encoder Representations from Transformers is a Natural Language Processing Model proposed by researchers at Google Research in 2018. Unlike other language representation models, BERT is designed to pre-train deep bidirectional representations from unlabelled text by jointly conditioning on both left and right context. Hence a wide range of tasks including question answering and language inference can be achieved by fine-tuning of just one additional output layer in the pre-trained BERT model to create state-of-the-art models. BERT is a “deeply bidirectional” model which is based on the Transformer model architecture, where the attention mechanism learns contextual relationships between words in a text. BERT also uses pre-trained models including the entire Wikipedia (that’s a large corpus of 2500 million words of unlabelled text) and a Book Corpus of 800 million words. Due to the Bidirectional nature of BERT, it learns information from both the left and the right side of a token’s context during the training phase.  
This is explained in the below example:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/09/sent_context.png)

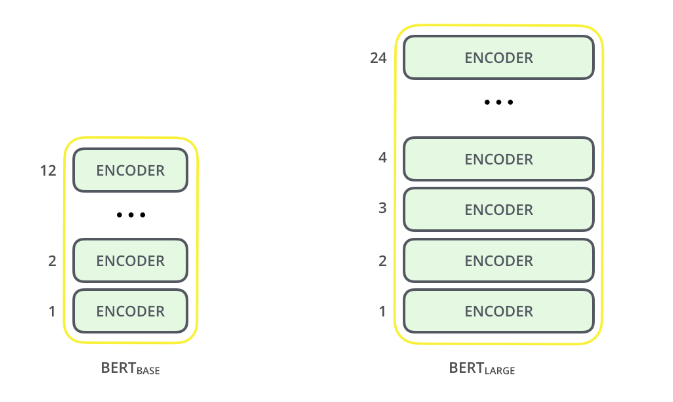
**Figure2** : BERT is bidirectional ( both left and right context is captured)

If the word “bank” is predicted by taking the left context or the right context, then there is going to be an error in one of the two examples given above .

However if we consider both the left and the right context before prediction we may be able to deal with this error. Since BERT is bidirectional we can see later in the document how it achieves this. Infact the most impressive aspect of BERT is its ability to fine-tune by adding just additional output layers to achieve state-of-the-art accurate models for a variety of NLP tasks.[[2]](#footnote-2)

**BERT Architecture:**

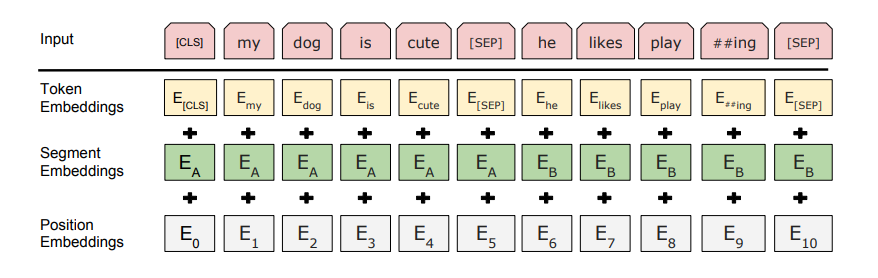
BERT is based on transformer architecture . There are two types of available BERT models. The first one is known as BERT Base and has 12 layers (transformer blocks), 12 attention heads, and 110 million parameters. The second one is BERT Large and has 24 layers (transformer blocks), 16 attention heads and, 340 million parameters. All of these are Transformer layers. In NLP ,transformer model is a novel architecture which aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease. Bert is essentially an Encoder stack of transformer architecture.

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/09/bert_encoder.png)

**Figure3**: BERT encoder blocking  
Source: *https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/*

**Text Pre-processing**

It is interesting how the developers of BERT models created a precise set of rules to classify the input text which make the BERT model better due to the multiple design choices. As seen below every input embedded is basically a combination of 3 different types of embeddings.

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/09/bert_emnedding.png)

**Disadvantages of SVM**:  
Although SVM is more efficient in some cases, it also has some fundamental problems. Firstly the SVM algorithm is not suitable for large data sets and does not perform very well when the data set target classes are overlapping because of the large amount of training time required to complete it. Secondly, in cases where the number of features for each data point exceeds the number of training data samples, the SVM will underperform. Lastly, As the support vector classifier works by putting data points, above and below the classifying hyperplane there is no probabilistic explanation for the classification.

Figure4: BERT Token, Segment and Position Embeddings  
Source: *https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270*

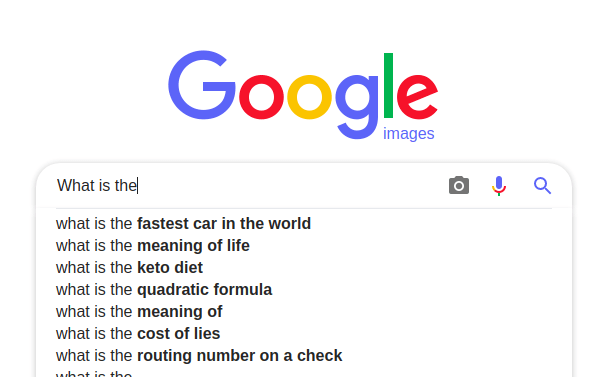
**Position Embeddings**. The positional embeddings are used by BERT to learn and capture the position and order of words in a sentence. They are also useful additions incase we have a transformer which is unable to capture “sequence” or “order” information.

**Segment Embeddings**: For Question-Answering using BERT it can also take sentence pairs as inputs. BERT is able to distinguish between these sentence pairs as it is able to learn a unique embedding for the first and the second sentences. Tokens in the above figure marked as EA belong to sentence and EB belongs to sentence B.

**Token Embeddings**: These are the embeddings learned for the specific token from the “WordPiece” token vocabulary.

Bert uses a comprehensive embedding scheme which makes it versatile as it provides useful information to the model by having multiple pre processing combinations. The input for a given token using BERT is constructed by adding the token embedding, segment embedding as well as position embedding thereby making it possible to use BERT on numerous NLP tasks without making any major changes in its architecture

The input representation for a given token using Bert is constructed by summing the corresponding token, segment, and position embeddings. This comprehensive embedding scheme contains a lot of useful information for the model and the combinations of these pre-processing steps make BERT versatile. Hence,We can easily train on multiple kinds of NLP tasks without making any major change in the model’s architecture,

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/09/language_model.png)  
**Figure5**: BERT real life use (Google), predicting word in a sequence

**Advantages of BERT:**

BERT is derived from a mask language model which represents context based word model with advantage of being pre-trained using bi-directional transformers .Its advantageous due to its bidirectional masked model over previous language models that had the limitation to use the combination of both unidirectional models that is left to right model and right to left model. BERT is also able to predict the sequence of masked words and can perform on most tasks without making major changes in the architecture.

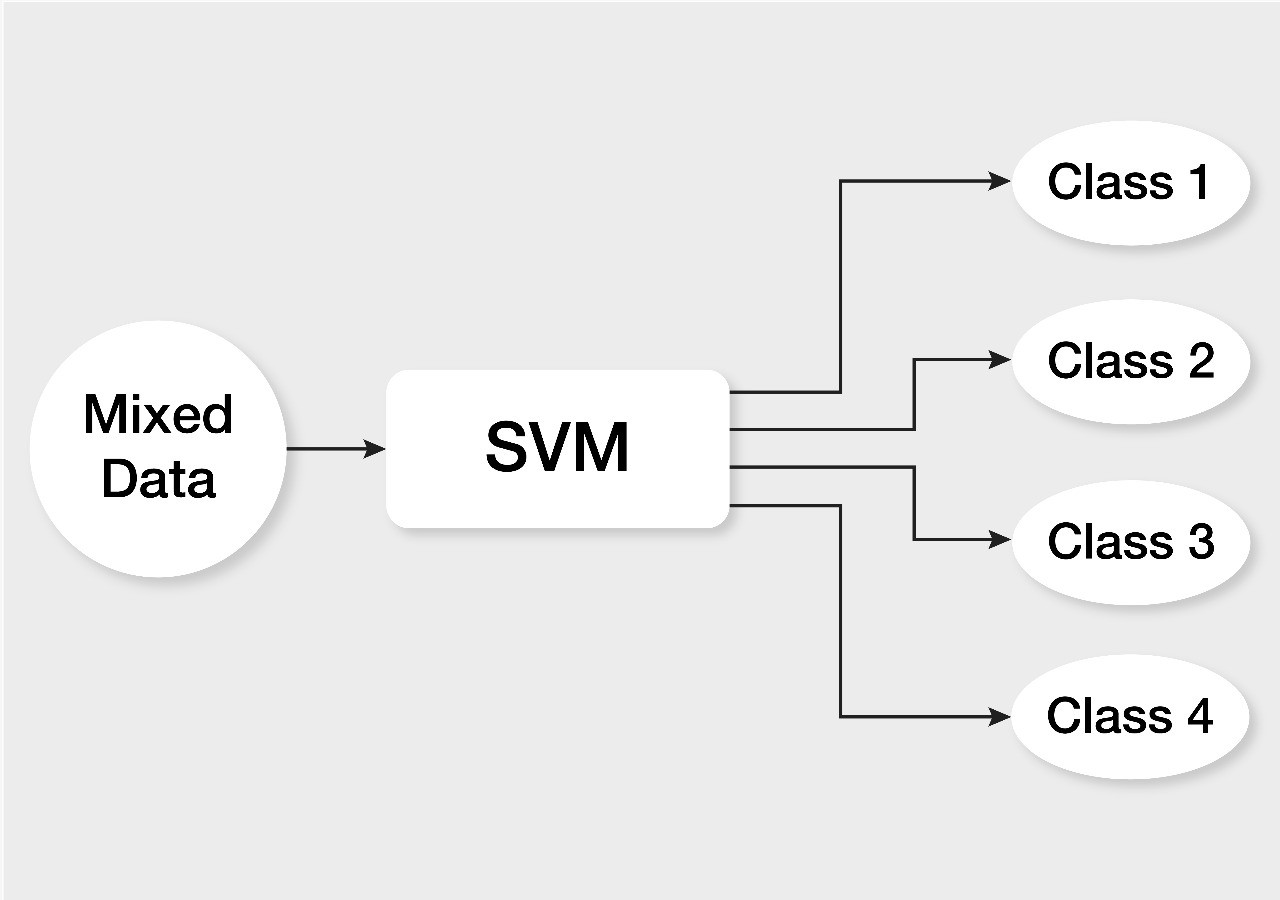
**Disadvantages of BERT:**

The major disadvantage of BERT is that it can become costly in its use for production at scale because of its compute intensive nature at inference time where it generates contextualized embeddings of word or vectors

Also while vectors like word2vec are precalculated and saved for retreaval whenever required in other models incase of Bert we need to compute or calculate vectors every time.

## **2.2 Support Vector Machine (SVM):**

SVM is a supervised machine learning algorithm that can be used for both classification and regression challenges. Classification is predicting a label/group and Regression is predicting a continuous value. SVM performs classification by finding the hyper-plane that differentiates the classes we plotted in n-dimensional space.



**Figure7**: SVM Classification after Data

Diagram

Description automatically generated  
**Figure8**: SVM hyperplane +ve and -ve segregation

SVM draws that hyperplane by transforming our data with the help of mathematical functions called “Kernels”. Types of Kernels are linear, sigmoid, RBF, non-linear, polynomial, etc.,

The tuning parameter Kernel — “RBF” is for non-linear problems and it is also a general-purpose kernel used when there is no prior knowledge about the data. Kernel —” linear” is for linear separable problems. Since our problem is linear(just positive and negative) here, we will go for “linear SVM”. **A hyperplane in an n-dimensional Euclidean space is a flat, n-1 dimensional subset of that space that divides the space into two disconnected parts. In this case the SVM algorithm will define the result based on the position of the point compared to the line (right or left)**[[3]](#footnote-3)

**Advantages of SVM:**Support Vector Machine (SVM) has been chosen for the classification in the experiments. The support-vector machine is used for two-group classification problems. It is used to classify the texts as positives or negatives. SVM works well for text classification due to its advantages such as its potential to handle large features but Using [SVM classifiers](https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/) for text classification tasks might be a really good idea, especially if the training data available is not on to bigger scale. SVM is also known to be more effective in high dimensional spaces or in cases where the number of dimensions is greater than the number of samples due to the model complexity of O (n-features \* n² samples). Moreover SVM is relatively memory efficient due to its program structure.

# **Data Cleaning, and Experimental Methodology**

## **Data Cleaning Process for Text Analytics**

Data cleaning is important process in text analytics and that’s why it is explained in detail. It’s primarily used in correction of corrupted incorrectly formatted duplicated or any kind of incomplete data in a dataset. When data is collected from multiple sources there could be many times it is duplicated or labelled incorrectly. The use of such incorrect data will lead to unreliable results using algorithms and we would require a data cleaning process which could be different for one dataset to another. But it is vital to establish a template for data cleaning process for text analytics. The following are some of the key steps to be performed prior to the model building process.

1. Remove Duplicate, Punctuation, and irrelevant observations
2. Fix Structural Errors and convert all the text to lower case
3. Tokenization of text and Filter Unwanted Outliers, Stop words
4. Handle missing and Non-alpha text
5. Word Lemmatization and load the date in Pandas Data frame.

### Remove duplicate or irrelevant observations

The unwanted observations from the dataset including duplicates, incorrect or irrelevant observations will generally happen when multiple datasets from multiple places scrape data, or maybe from multiple departments and clients are combined as part of data collection. Removing this duplication of data or irrelevant observations specific to the problem we are analysing is crucial. For example if a company wants to analyse data from a specific group, say millennial customers. But there dataset used has irrelevant observation from older generations as well the analysis done on such dataset will be inefficient and not fulfil the primary requirement as compared to when a higher performing dataset with only millennial customers observations is used.

### Fix structural errors and Convert all the text to lower case.

Structural errors can cause mislabelled categories when data transferred or analysed has strange naming conventions ,incorrect capitalization or typos. For instance, we may find both “N/A” and “Not Applicable”, however they should be in the same category for analysis. Perform unification of textual contents in complete lower case.

### Tokenization of words and Filter Unwanted Outliers, Stop words

Sometimes when we analyse data we observe one-off entries that do not appear to fit the dataset we are analysing. These are called outliers which should be removed in case of improper data entry however just because we see an outlier we cannot conclude that it is incorrect we would need further steps to determine its validity and weather it would be irrelevant for analysis as it may impact the performance of the data and it would be a mistake to remove it.

Stop words are words which are filtered out generally before processing a natural language. They could be common words or grammar including like conjunctions, pronouns, prepositions, articles, etc.  
Since the grammar and syntax are available for all human languages and they do not add much information, removing them the text helps give more focus to the important information. Another advantage of removing stop words is the dataset size is reduced which reduces the time and lesser number of tokens are involved.[[4]](#footnote-4)

|  |
| --- |
| **E.g.**  **Electronic review:** “The Phone Charger was not good at all.”  **After removing stop words:** “Phone Charger not good” |

### Handle missing and Non-Alpha Text

### Many algorithms do not accept missing values in the data hence it is important to deal with missing data in the following ways:

1. Firstly, we can choose to drop observations with missing values but we need to be careful before we remove them as we may lose information by doing so.
2. Secondly, the missing value should be analysed for inputs based on other data observation. This may also impact by loss of integrity of data because the entered value is not an actual observation but are assumptions
3. Finally, the data could be altered to be used to effectively navigate null values.

## **Experimental Procedure**

## **The Dataset Used**

The customer review dataset was downloaded from Amazon web site into the Google collab platform for processing. The dataset was imported through the python TensorFlow libraries to the system. The dataset was huge so truncated some rows and kept 50K rows for faster processing. For data to be properly used, added the sentiment feature with 1 (Positive) or 0 (Negative) for all reviews based on the star rating given by the customer.

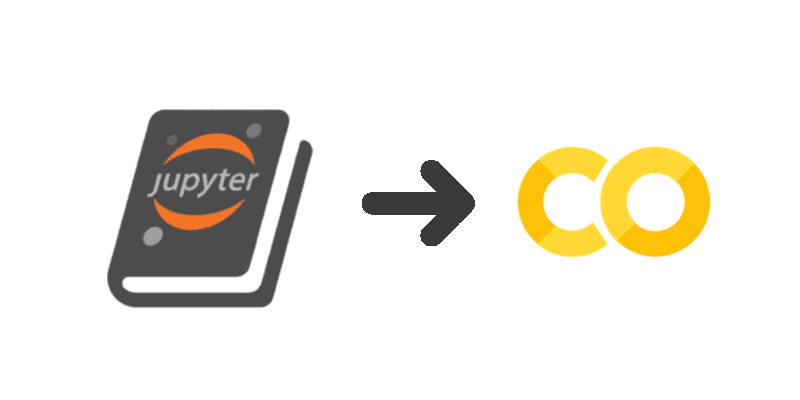
**Dataset Source:** <https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Mobile_Electronics_v1_00.tsv.gz>

## **Pre-processing**

The data was in TSV format and pre-processed in Python using various libraries (see appendix). For reviews with star ratings less than 3, the sentiment is qualified as negative and positive sentiment for start ratings 3 and above. The dataset was then split in train and test with 0.8 to 0.2 ratio. The dataset is then converted into truncated format to have data\review and sentiment features only for further processing. The detail steps to process the data and build and test BERT and SVM models for accuracy are in the next section.

## **Experiment Details:**

1. Setup the Experiment on Google Collaboratory Platform, Technology Stack.

  
**Figure9** : Jupiter-Notebooks and Google Collab coding Platforms

1. Install all the necessary Python Libraries for Natural Language Processing, Model Functions like SVM and BERT Transformer libraries along with TensorFlow.
2. Download the dataset of US amazon mobile electronic product review from Amazon web site into Google Collab platform for text analytics. The dataset was imported using tensor flow libraries and converted the dataset into pandas dataframe and then print the a few rows of the dataset. The following table shows some of the key features of the Amazon dataset.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **customer\_id** | **product\_category** | **product\_id** | **review\_body** | **review\_date** | **eview\_headline** | **star\_rating** |
| b'20980074' | b'Mobile\_Electronics' | b'B00D1847NE' | b'Does not work' | b'2015-01-09' | b'One Star' | 1 |
| b'779273' | b'Mobile\_Electronics' | b'B00KMO6DYG' | b'This is a great wiring kit i used it to set up my Pyle 2000 watt amp to 2 alpine subs and worked just fine. why i gave it 4 stars. great kit' | b'2015-08-06' | b'Great kit' | 4 |
| b'15410531' | b'Mobile\_Electronics' | b'B000GWLL0K' | b'It works great so much faster than USB charger..Buy it! You will be glad you did!' | b'2007-03-15' | b'A/C Charger for Creative Zen Vision M' | 5 |
| b'27389005' | b'Mobile\_Electronics' | b'B008L3JE6Y' | b'This product was purchased to hold a monitor on a desk, which is connected to a security camera at a door of our business. It serves our purpose perfectly.' | b'2013-07-30' | b'camera stand' | 5 |
| b'2663569' | b'Mobile\_Electronics' | b'B00GHZS4SC' | b"it works but it has really bad sound quality. the bass doesn't work almost at all" | b'2014-12-31' | b'bad sound quality' | 3 |

Figure10: Amazon Mobile Review Dataset Key Features

1. Pre-process and convert the data in Pandas Data frame. The dataset consists of several features such as Customer ID, Product ID, Product Category, review body, date and star Rating. We want to keep start rating and review features provided by the customer so we have to drop all other columns.
2. Enable sentiment score based on short review and star rating, add a sentiment column with positive (1) and negative (0) categories. The star rating given by the customer is on a scale of 1 to 5 where 5 is the best rating and 1 being the lowest rating. We want to build classification models for this experiment so will translate the star rating into two categories, “1” and “0”. The star rating 3 and above will be labelled as positive sentiment with “1” and star rating less than 3 will be negative sentiment and labelled as “0”. The following table shows the output of last 5 rows after the sentiment labels are created.

|  |  |  |
| --- | --- | --- |
| **Index** | **short\_review** | **Sentiment** |
| 49995 | Awesome | 1 |
| 49996 | I got this cover for my daughter's ipod. It is so sweet and all her friends loved it. | 1 |
| 49997 | This battery does not fit the T-mobile G2 is it too BIG, and by trying to fit it in one of the metal pieces for the battery almost broke. DO NOT BUY FOR G2! | 0 |
| 49998 | It has good sound but , the charge only last 1 to 1 and a half hours. | 1 |
| 49999 | What I imaginaed getting from this machine was a portable, wet/dry vac with a scrubber and spray. | 1 |

**Figure11**: Dataset after processing

1. Split the data in train and test datasets in 80-20 Ratio, 80% for training and 20% for testing and validations. We are using scikit learn class to split the data in test and training dataset.
2. We have to convert our review column into numerical values to build BERT and SVM machine learning models. The text vector conversion can be achieved through tokenizer class. The words gets converted into token based on the frequency of word appearance in the review feature column.
3. Set up the model using the Python Tensorflow NLP machine learning libraries for BERT and SVM.
4. Build and train the BERT and SVM models on the training data i.e. on 80% split from the initial dataset. We can save the model and load it to evaluate the accuracy of predictions on testing data.
5. Run the trained model on the testing data (a 20% split from the initial dataset).
6. Record the Accuracy, F1 score and Loss of BERT and SVM models and compare the results of BERT and SVM algorithms.

# **Experimental Results & Evaluation Matrices**

## 4.1 **Evaluation Matrices Explanation:**

The following evaluation parameters are used to analyse the accuracy of BERT and SVM models :  
 **F1 Score:** The F1 score recorded through this experiment is interpreted as a weighted average of the precision and recall values, where an F1 score reaches its best value at 1 and worst value at 0. **Accuracy:** Accuracy of this experiment is defined as the ratio of Sum of True Positives and True Negatives to the Total Number of Predictions. Accuracy is important when the True Positives and True negatives are more important while F1-score is used when the False Negatives and False Positives are crucial. Accuracy can be used when the class distribution is similar while F1-score is a better metric when there are imbalanced classes as in the above case.

**Loss:** Loss value is the penalty for a bad predictions occurred throughout this experiment. Loss is a number indicating how bad the model's prediction was on the dataset. If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater.

## 4.2 **Evaluation Matric Final Output Result Tab**

|  |  |  |  |
| --- | --- | --- | --- |
| Model | F1 Score | Accuracy | Loss |
| Support Vector Machine Classifier (Supervised Learning) | **0.9010** | **0.9013 | 90%** | **0.0987** |
| BERT (Bidirectional Encoder Representations from Transformers) | **0.9899** | **0.9809 | 98%** | **0.0548** |

**Table1**: BERT and SVM Result Table

# **Analysis and Conclusion**

## **Results Analysis**

Based on the experiment results, BERT algorithm has better accuracy and F1score as compared to SVM algorithm. The results recorded with 98% accurate predictions when using BERT algorithm as compared to 90% accurate predictions using SVM algorithm for sentiment analysis of Amazon customer reviews for Mobile device feedback. Hence, this leads to the conclusion that BERT has better accuracy for sentiment analysis on the same set of data. Amazon dataset was used to perform sentiment analysis since it provided varied amounts of online reviews for 50K reviews, hereby more chance to explore varied sentiment analysis.

F1 score or F-measure is a measure of a weighted average of precision and recall and it’s best value is ‘1’ and worst model score is ‘0’. The experiment shows that BERT is with F1 score of 0.98 and SVM’s F1 score is 0.90. F1-score is calculated from the [precision](https://en.wikipedia.org/wiki/Precision_(information_retrieval)) and [recall](https://en.wikipedia.org/wiki/Recall_(information_retrieval)) of the test, where the precision is the number of true positive results divided by the number of all positive results, including those not identified correctly, and recall is the number of true positive results divided by the number of all samples that should have been identified as positive.

* 1. **Conclusion**

This investigation aimed at assessing the accuracy of BERT and SVM NLP algorithms for sentiment analysis of online reviews. After carrying out an experiment and assessing the results, I’m led to conclude that BERT has a better accuracy as compared to SVM for sentiment analysis.

BERT is a really powerful language representation model that has been a big milestone in the field of NLP. It has greatly increased our capacity to do transfer learning in NLP. BERT models perform fairly well in comparison to SVM model. However, BERT require high computational power and a large time to train on a model. Thus, this model is primarily used in complex dataset and applications which require high accuracy but we can also use SVM which is a simpler model and it is faster to train SVM with less computational requirement.

# **References:**

*1.4. Support Vector Machines*. scikit. (n.d.). Retrieved January 16, 2022, from <http://scikit-learn.org/stable/modules/svm.html>

Bedi, G. (2020, July 13). *Simple guide to text classification(nlp) using SVM and Naive Bayes with python*. Medium. Retrieved January 16, 2022, from <https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34>

Horev, R. (2018, November 17). *Bert explained: State of the art language model for NLP*. Medium. Retrieved January 16, 2022, from <https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270?gi=5422e6b2e031>

*SVM: Support Vector Machine Algorithm in machine learning*. Analytics Vidhya. (2021, August 26). Retrieved January 16, 2022, from <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code>

*What is Bert: Bert for text classification*. Analytics Vidhya. (2020, June 14). Retrieved January 16, 2022, from <https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/>

Tigue, J. (n.d.). *Jupyter book to colab*. John Tigue's Tech Writings. Retrieved January 16, 2022, from <https://tigue.com/jupyter_book_to_colab/>

# **Appendix:**

**Dataset Source:** <https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Mobile_Electronics_v1_00.tsv.gz>

**Code Repository:**

* 1. **BERT Implementation**

|  |
| --- |
| # -\*- coding: utf-8 -\*-  """NLP-Sentiment Analysis-BERT-Amazon Reviews.ipynb  Automatically generated by Colaboratory.  Original file is located at  https://colab.research.google.com/drive/1QbJfE0XL24\_ydAA3GJaUnu71N8HfhmLm  """  import tensorflow as tf  #num\_gpus\_available = len(tf.config.experimental.list\_physical\_devices('GPU'))  #print("Num GPUs Available: ", num\_gpus\_available)  #assert num\_gpus\_available > 0  !pip install transformers  from transformers import DistilBertTokenizerFast  from transformers import TFDistilBertForSequenceClassification  import pandas as pd  import numpy as np  from sklearn.model\_selection import train\_test\_split  import nltk,re  nltk.download('stopwords')  from nltk.corpus import stopwords  from nltk.stem.porter import PorterStemmer  import tensorflow\_datasets as tfds  ds = tfds.load('amazonreviews/MobileElectronics\_v1\_00', split='train', shuffle\_files=True)  assert isinstance(ds, tf.data.Dataset)  df = tfds.as\_dataframe(ds)  df.head()  df["Sentiment"] = df["data/star\_rating"].apply(lambda score: "positive" if score >= 3 else "negative")  df['Sentiment'] = df['Sentiment'].map({'positive':1, 'negative':0})  df['short\_review'] =df['data/review\_body'].str.decode("utf-8")  df = df[["short\_review", "Sentiment"]]  # Dropping last n rows using drop  n = 54587  df.drop(df.tail(n).index,  inplace = True)  index = df.index  number\_of\_rows = len(index)  print(number\_of\_rows)  df.tail()  reviews = df['short\_review'].values.tolist()  labels = df['Sentiment'].tolist()  #print(reviews[:2])  #print(labels[:2])  training\_sentences, validation\_sentences, training\_labels, validation\_labels = train\_test\_split(reviews, labels, test\_size=.2)  tokenizer = DistilBertTokenizerFast.from\_pretrained('distilbert-base-uncased')  tokenizer([training\_sentences[0]], truncation=True,  padding=True, max\_length=128)  train\_encodings = tokenizer(training\_sentences,  truncation=True,  padding=True)  val\_encodings = tokenizer(validation\_sentences,  truncation=True,  padding=True)  train\_dataset = tf.data.Dataset.from\_tensor\_slices((  dict(train\_encodings),  training\_labels  ))  val\_dataset = tf.data.Dataset.from\_tensor\_slices((  dict(val\_encodings),  validation\_labels  ))  model = TFDistilBertForSequenceClassification.from\_pretrained('distilbert-base-uncased',num\_labels=2)  optimizer = tf.keras.optimizers.Adam(learning\_rate=5e-4, epsilon=1e-09)  model.compile(optimizer=optimizer, loss=model.compute\_loss, metrics=['accuracy'])  model.fit(train\_dataset.shuffle(100).batch(16),  epochs=5,  batch\_size=8,  validation\_data=val\_dataset.shuffle(100).batch(8))  optimizer = tf.keras.optimizers.Adam(learning\_rate=5e-5, epsilon=1e-08)  model.compile(optimizer=optimizer, loss=model.compute\_loss, metrics=['accuracy'])  model.fit(train\_dataset.shuffle(100).batch(16),  epochs=2,  batch\_size=16,  validation\_data=val\_dataset.shuffle(100).batch(16))  model.save\_pretrained("./sentiment")  loaded\_model = TFDistilBertForSequenceClassification.from\_pretrained("./sentiment")  test\_sentence = "This is a really good product. I love it"  predict\_input = tokenizer.encode(test\_sentence,  truncation=True,  padding=True,  return\_tensors="tf")  tf\_output = loaded\_model.predict(predict\_input)[0]  tf\_prediction = tf.nn.softmax(tf\_output, axis=1)  labels = ['Negative','Positive']  label = tf.argmax(tf\_prediction, axis=1)  label = label.numpy()  print(labels[label[0]]) |

* 1. **SVM Implementation**

|  |
| --- |
| # -\*- coding: utf-8 -\*-  """NLP-Sentiment Analysis-Multiple Models-Amazon Reviews.ipynb  Automatically generated by Colaboratory.  Original file is located at  https://colab.research.google.com/drive/114NJ78a4kDdY-EZcoZkm7RjnEGxZQuHh  """  import tensorflow as tf  !pip install transformers  from transformers import DistilBertTokenizerFast  from transformers import TFDistilBertForSequenceClassification  import pandas as pd  import numpy as np  import nltk  import re  nltk.download('stopwords')  from nltk.corpus import stopwords  from nltk.stem.porter import PorterStemmer  from sklearn.model\_selection import train\_test\_split  import tensorflow\_datasets as tfds  from sklearn.naive\_bayes import MultinomialNB  from sklearn.pipeline import Pipeline  ds = tfds.load('amazonreviews/MobileElectronics\_v1\_00', split='train', shuffle\_files=True)  assert isinstance(ds, tf.data.Dataset)  df = tfds.as\_dataframe(ds)  df.head()  df["Sentiment"] = df["data/star\_rating"].apply(lambda score: "positive" if score >= 3 else "negative")  df['Sentiment'] = df['Sentiment'].map({'positive':1, 'negative':0})  df['short\_review'] =df['data/review\_body'].str.decode("utf-8")  df = df[["short\_review", "Sentiment"]]  # Dropping last n rows using drop  n = 54445  df.drop(df.tail(n).index,  inplace = True)  index = df.index  number\_of\_rows = len(index)  print(number\_of\_rows)  df.tail()  reviews = df['short\_review'].values.tolist()  labels = df['Sentiment'].tolist()  print(reviews[:2])  print(labels[:2])  training\_sentences, validation\_sentences, training\_labels, validation\_labels = train\_test\_split(reviews, labels, test\_size=.2)  # Replace "nan" with space  X\_train = training\_sentences  X\_test = validation\_sentences  X\_train\_targetSentiment = training\_labels  X\_test\_targetSentiment = validation\_labels  # Text preprocessing and occurance counting  from sklearn.feature\_extraction.text import CountVectorizer  count\_vect = CountVectorizer()  X\_train\_counts = count\_vect.fit\_transform(X\_train)  X\_train\_counts.shape  tfidf\_transformer = TfidfTransformer(use\_idf=False)  X\_train\_tfidf = tfidf\_transformer.fit\_transform(X\_train\_counts)  X\_train\_tfidf.shape  ~~"""\*\*Support Vector Machine Classifier\*\*"""~~  from sklearn.svm import LinearSVC  clf\_linearSVC\_pipe = Pipeline([("vect", CountVectorizer()), ("tfidf", TfidfTransformer()), ("clf\_linearSVC", LinearSVC())])  clf\_linearSVC\_pipe.fit(X\_train, X\_train\_targetSentiment)  predictedLinearSVC = clf\_linearSVC\_pipe.predict(X\_test)  np.mean(predictedLinearSVC == X\_test\_targetSentiment) |

1. <https://www.ibm.com/cloud/learn/natural-language-processing#:~:text=Natural%20language%20processing%20(NLP)%20refers,same%20way%20human%20beings%20can>. [↑](#footnote-ref-1)
2. *What is Bert: Bert for text classification*. Analytics Vidhya. (2020, June 14). Retrieved January 16, 2022, from <https://www.analyticsvidhya.com/blog/2019/09/demystifying-bert-groundbreaking-nlp-framework/> [↑](#footnote-ref-2)
3. *SVM: Support Vector Machine Algorithm in machine learning*. Analytics Vidhya. (2021, August 26). Retrieved January 16, 2022, from <https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code> [↑](#footnote-ref-3)
4. Bedi, G. (2020, July 13). *Simple guide to text classification(nlp) using SVM and Naive Bayes with python*. Medium. Retrieved January 16, 2022, from <https://medium.com/@bedigunjit/simple-guide-to-text-classification-nlp-using-svm-and-naive-bayes-with-python-421db3a72d34> [↑](#footnote-ref-4)