Automate detection of different sentiments from paragraphs and predict overall sentiment

Synopsis:

Develop algorithms that process labelled datasets to learn the sentiments present within and then accurately predict the sentiment of similar data.

Solution Approach:

I implemented the project on 3 separate datasets:

1) Twitter
   https://data.world/crowdflower/sentiment-analysis-in-text
2) IMDB
   https://www.kaggle.com/utathya/imdb-review-dataset
   http://ai.stanford.edu/~amaas/data/sentiment/
3) Amazon
   https://www.kaggle.com/bittlingmayer/amazonreviews
   http://deepyeti.ucsd.edu/jianmo/amazon/index.html

I've used NLTK and Regular Expression libraries for text cleaning in each of the five methods I've implemented. I also tried using un-processed data in the methods 2-5 but it resulted in a minute difference in test accuracy (-0.2%) so I didn't include those in my final solutions.

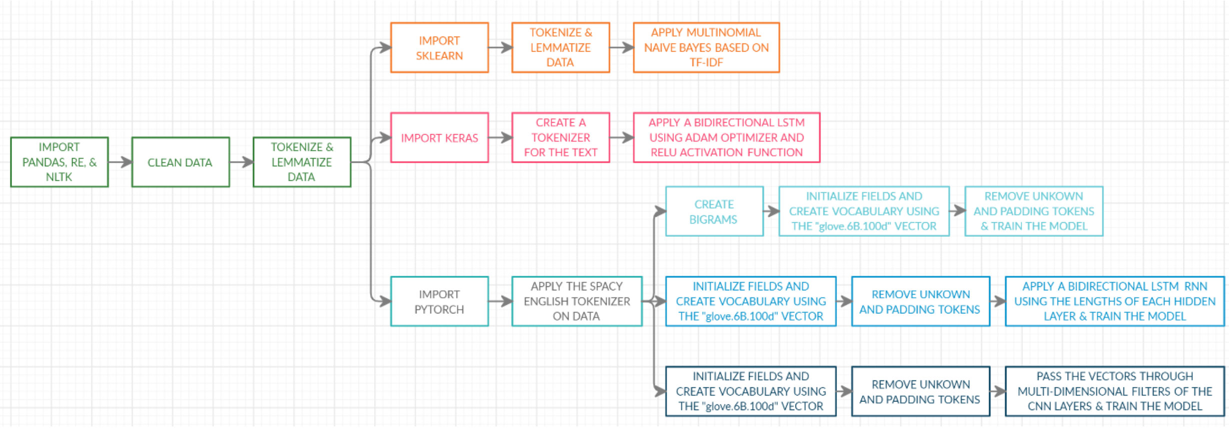
Methods:

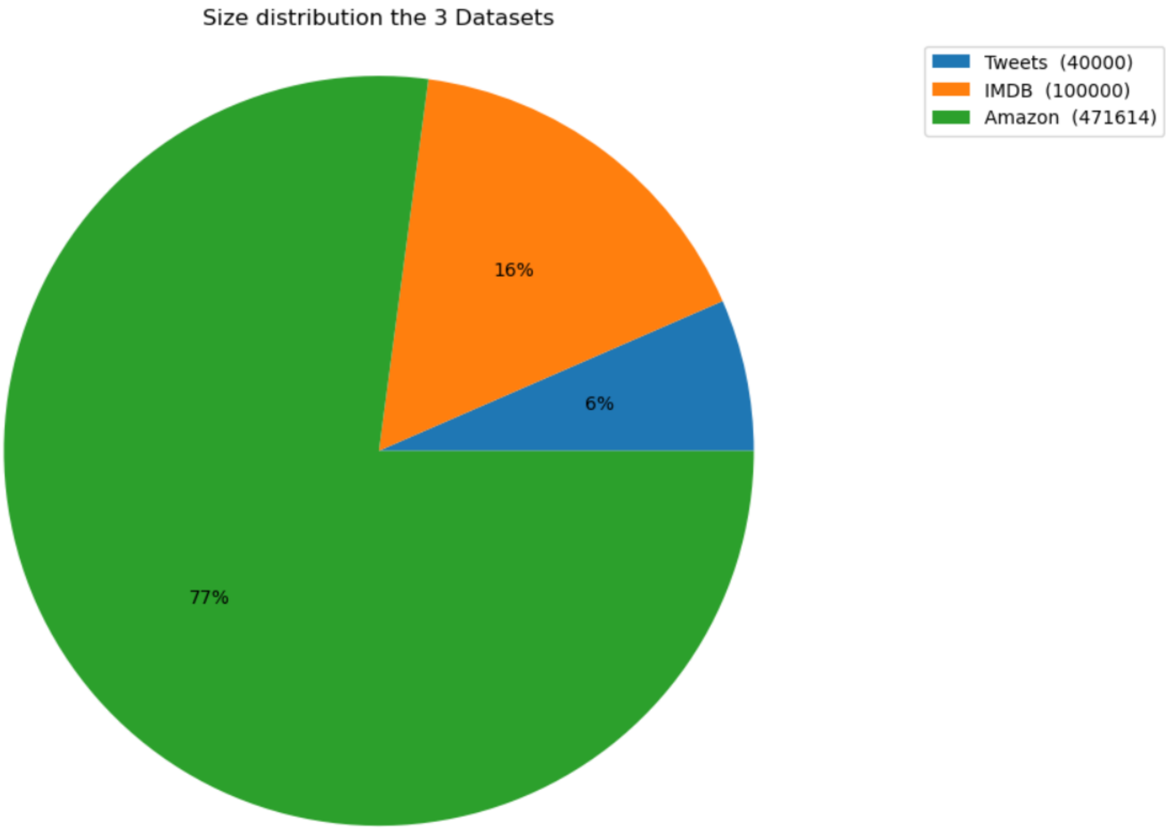
1) Multinomial Naïve Bayes based on tfidf using Sklearn
2) LSTM with relu and sigmoid activation functions using Keras
3) Bi-grams using Pytorch
4) 2D Bidirectional RNN using Pytorch
5) CNN layers of multi-dimensional filters using Pytorch

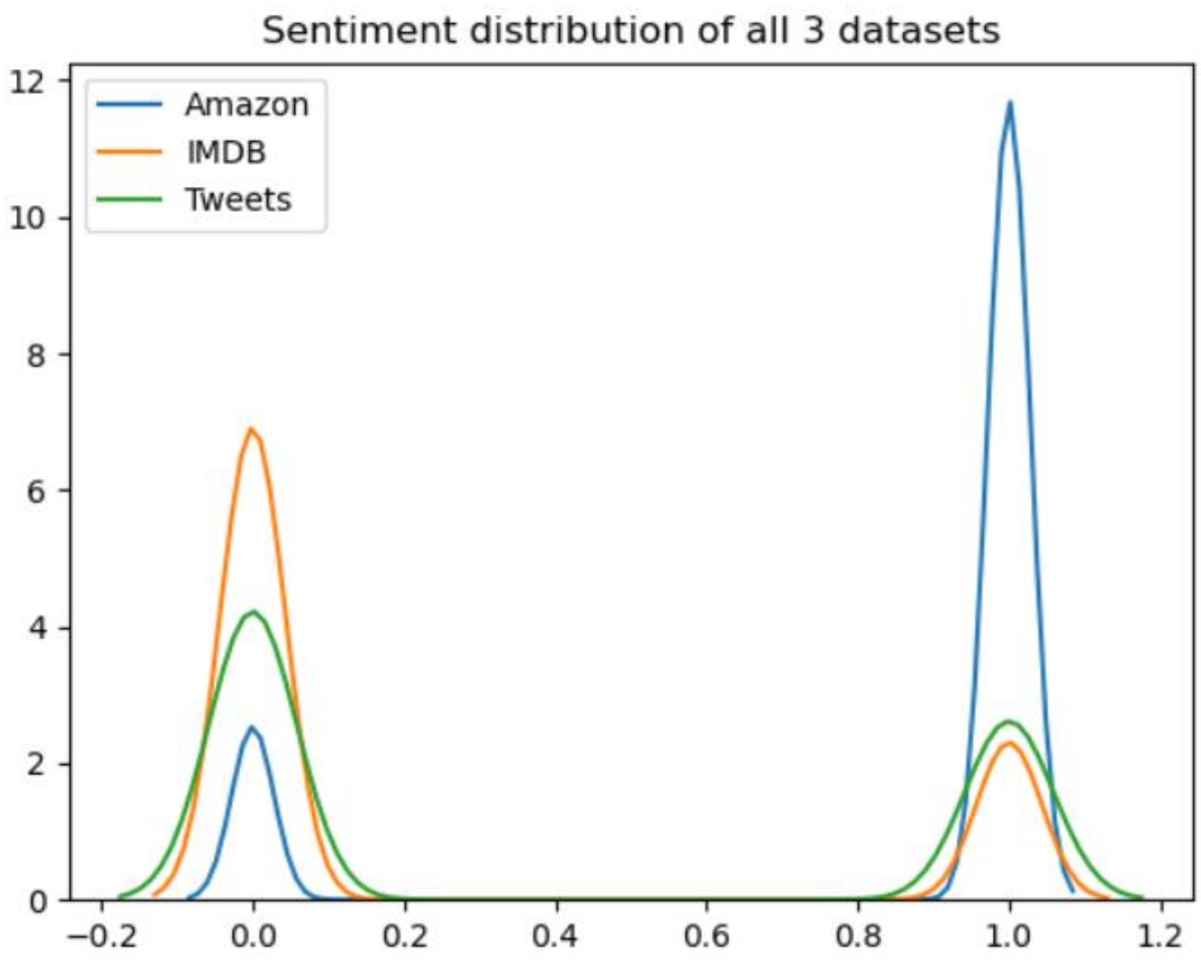
Assumptions:

1) The data is labelled correctly
2) Extremely short tweets/reviews like a single word or emoji can be used for training and subsequent prediction even though they might negatively affect results.

Diagrams & Charts:


IMPORT PANDAS, RE, & NLTK
CLEAN DATA
TOKENIZE & LEMMATIZE DATA
IMPORT SKLEARN
TOKENIZE & LEMMATIZE DATA
APPLY MULTINOMIAL NAIVE BAYES BASED ON TF-IDF
IMPORT KERAS
CREATE A TOKENIZER FOR THE TEXT
APPLY A BIDIRECTIONAL LSTM USING ADAM OPTIMIZER AND RELU ACTIVATION FUNCTION
IMPORT PYTORCH
APPLY THE SPACY ENGLISH TOKENIZER ON DATA
CREATE BIGRAMS
INITIALIZE FIELDS AND CREATE VOCABULARY USING THE "glove.6B.100d" VECTOR
REMOVE UNKOWN AND PADDING TOKENS & TRAIN THE MODEL
INITIALIZE FIELDS AND CREATE VOCABULARY USING THE "glove.6B.100d" VECTOR
REMOVE UNKOWN AND PADDING TOKENS
APPLY A BIDIRECTIONAL LSTM RNN USING THE LENGTHS OF EACH HIDDEN LAYER & TRAIN THE MODEL
INITIALIZE FIELDS AND CREATE VOCABULARY USING THE "glove.6B.100d" VECTOR
REMOVE UNKOWN AND PADDING TOKENS
PASS THE VECTORS THROUGH MULTI-DIMENSIONAL FILTERS OF THE CNN LAYERS & TRAIN THE MODEL


Size distribution the 3 Datasets
Tweets (40000)
IMDB (100000)
Amazon (471614)
16%
6%
77%



Algorithms:

1) Multinomial Naïve Bayes
2) Adam optimizer
3) Confusion matrices for F1 score

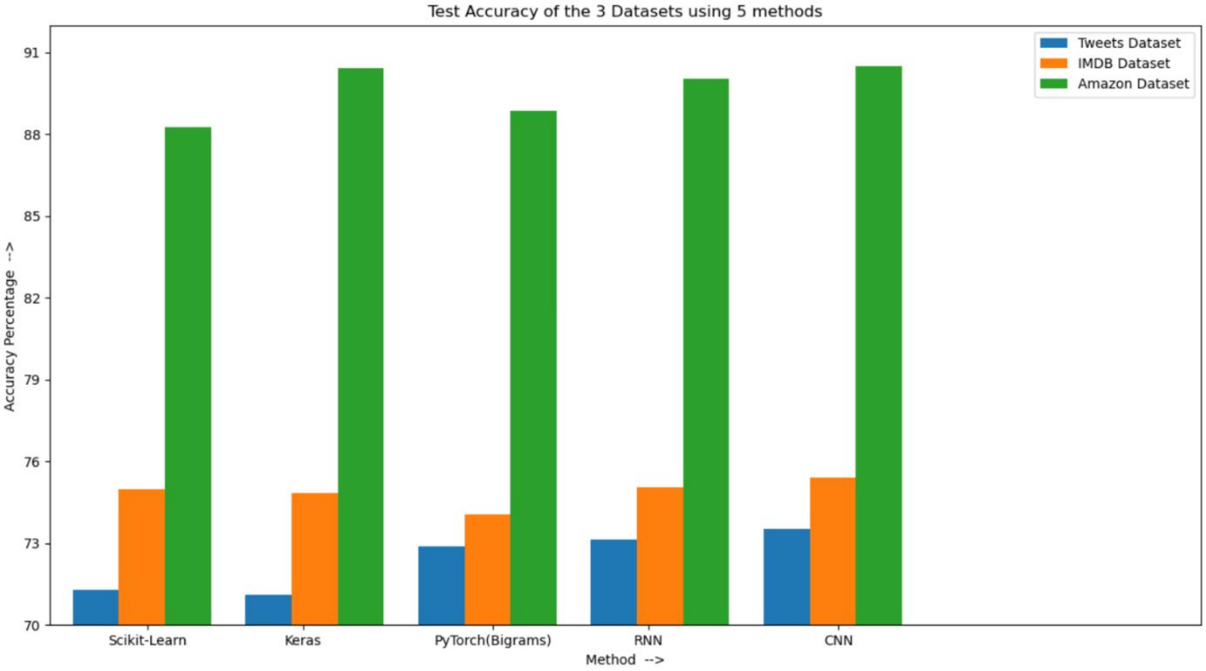
Outcome:

Testing Accuracy:

	IMDB	**TWEETS**	**AMAZON**
Multinomial Naïve Bayes - Sklearn	74.98%	71.31%	88.25%
LSTM - Keras	74.86%	71.10%	90.43%
Bigrams - Pytorch	74.06%	72.89%	88.86%
Bidirectional RNN - Pytorch	75.04%	73.14%	90.04%
CNN - Pytorch	75.42%	73.52%	90.50%

On comparing the results of the Amazon dataset with the others, it is evident that larger the amount of available data for training, higher the accuracy. But this could also be due to overfitting even though precautions were taken to avoid it.

Among the 5 methods used, the method of passing the vectors through multi-dimensional filters of CNN layers achieved the highest accuracy across all three datasets.



Exceptions considered:

The twitter dataset doesn't allow the algorithms to achieve high accuracy due to the use of slangs, different short forms for the same word and high amounts of sarcasm. I've included the dataset without removing such tweets for training the model but each such tweet is an exception in its own different way.

Also, the IMDB and Twitter datasets have a larger amount of negative sentiments which is directly in contrast to

Enhancement Scope:

Since it is clear that a larger amount of data for training results in higher accuracy, the algorithms on the IMDB and Tweets can be further enhanced to attain much higher accuracy by training them with more similar data.

I've included alternate datasets as well for this purpose.

Link to Code and executable file:

1)Multinomial Naïve Bayes using Sklearn:

IMDB:

https://colab.research.google.com/drive/1NY5nRT8Ja28A3wzO03fj88lDTm_J0QZ3?usp=sharing

Tweets:

https://colab.research.google.com/drive/1jB9Hh8dQqcMv7N0ldF5UvDu6_Qwo7P0a?usp=sharing

Amazon:

https://colab.research.google.com/drive/1yl0wmCzhulIk_pkh0FF7Lswi261bMwDL?usp=sharing

2)LSTM using Keras:

IMDB:

https://colab.research.google.com/drive/1VDPVgp_Gca3go3FKDq_1BOtdCPYQ8Bsi?usp=sharing

Tweets:

https://colab.research.google.com/drive/1MgSUTYgVE1e_X0ds3WchmYZL6KCYlZ0Z?usp=sharing

Amazon:

https://colab.research.google.com/drive/1d3d0pZlx-GdgYMBvOseYCU3C-eWvPFre?usp=sharing

3)Bigrams using Pytorch:

IMDB:

https://colab.research.google.com/drive/1g7rrlSa2LlZ4DfGk7eMAeWpXJk9CTeL9?usp=sharing

Tweets:

https://colab.research.google.com/drive/1LKAjb5zZF_P1coihE-Nhd92TBbV2RbMM?usp=sharing

Amazon:

https://colab.research.google.com/drive/1zFPL3_h2_2u4pumZHNU06TaN_N52Lg_z?usp=sharing

4)Bidirectional RNN using Pytorch:

IMDB:

https://colab.research.google.com/drive/1UW7qpjJ30jtxQ4VXLnBsgCg3aZYj3UUI?usp=sharing

Tweets:

https://colab.research.google.com/drive/1oAgJlBeY_LaZI46lShnX6T2IABulH_kN?usp=sharing

Amazon:

https://colab.research.google.com/drive/1X_L98YwvPO6S_mJexRdTsRQgXCXJEEp8?usp=sharing

5)CNN using Pytorch:

IMDB:

https://colab.research.google.com/drive/1NA0YFjWvS1ucf2z7fmUvBdprXdeT4AWs?usp=sharing

Tweets:

https://colab.research.google.com/drive/1j4-ciVdjg-ZfNF_G0bJGLbRT0vDY0c8Z?usp=sharing

Amazon:

https://colab.research.google.com/drive/1gxu81K4zoHY023GIOEAISzp2RuI786SN?usp=sharing