The task was to implement a Binary Perceptron and apply to it Sentiment Analysis for Film Reviews. The Binary Perceptron with Bag-of-Words representation has been successfully implemented along with a Bigram and Trigram as further feature types as required.

Code Implementation Explanation

Execution

The file is executed in Terminal using, *python3 lab1.py review_polarity* where review_polarity is the data folder. It is assumed that the data folder is in the same directory as the executable python file. To access the individual film reviews in the dataset, the path to the reviews is found via the CommandLine class. This returns the paths to both the positive review dataset and negative review dataset.

Dataset and Model Creation

The reviews are downloaded using the returned paths and split into the training dataset and test dataset by *splitFullDataset* method in the *Dataset* class. The first 800 positive and negative reviews makeup the training dataset and the last 200 positive and negative reviews makeup the training dataset. During creation of the datasets the individual reviews are split into terms by the regular expression *"[\W]", " "*, this splits the review by whole words and blank spaces. Each term is then screened to remove any terms that may be stop words. Stop words are words unlikely to be beneficial in determining sentiments and so are removed to avoid corrupting results. The remaining terms are then used to create the feature types as required.

For the Unigram bag of words model the final structure is:
- *{(Review Sentiment Label, File Name): {Term 1: Count, Term 2: Count}}*

For the Bigram model:
- *{(Review Sentiment Label, File Name): {(Term 1, Term 2: Count), (Term 2, Term 3: Count)}}*

For the Trigram model:
- *{(Review Sentiment Label, File Name): {(Term 1, Term 2, Term 3: Count), (Term 2, Term 3, Term 4: Count)}}*

Weight Vector Generation

The blank weights vector is generated by the *Weights* class using the *generateInitialWeights* method. By iterating through the training dataset and adding previously unseen terms to the vector, the weights vector is created, terms in the vector are given an initial weight of 0.

Training

Using the blank weights vector and training dataset for each model, the weights are updated each training iteration with a value dependant on the sentiment prediction given by the perceptron algorithm. Training continues until the model is said to have converged, convergence in this case is defined as when the change in prediction error between two iterations is less than 0.005. The prediction errors for each model is shown in figure 1. During a training iteration the inputted dataset is randomised using a random seed equal to the iteration count. The randomisation stops the weights becoming generalised to a fixed order of reviews. The perceptron calculates the dot product between the occurrences of terms in the review and the corresponding weights for those terms. The sign step input is applied to the dot product, to produce either a positive or negative sentiment prediction for the review. If the dot product is less than 0 the prediction is negative, else the prediction is positive. If this prediction is not equal to the actual review sentiment then the weights are updated accordingly. After each iteration the weights are averaged by the iteration counter returning the average of all weight vectors for each class.

Other Feature Types Implemented

The two other feature types implemented are Bigrams and Trigrams. Bigrams and Trigrams include information about sentence structure. Unigrams only have information on individual words such as (“good”: 5) all structure is lost, bigrams include information about the immediate neighbouring term (“very”, “good”: 2), and trigrams include even more information about the structure of the sentence including the immediate two next neighbours. (“not”, “very”, “good”). There is a trade-off between the amount of sentence structure information retained and prediction accuracy, too much information and it’s likely that the term will only be relative to a specific reviews. The weights model will therefore be unable to generalise to new data. The more words included in the term the higher the accuracy on training datasets but the lower the prediction accuracy on unseen training data.

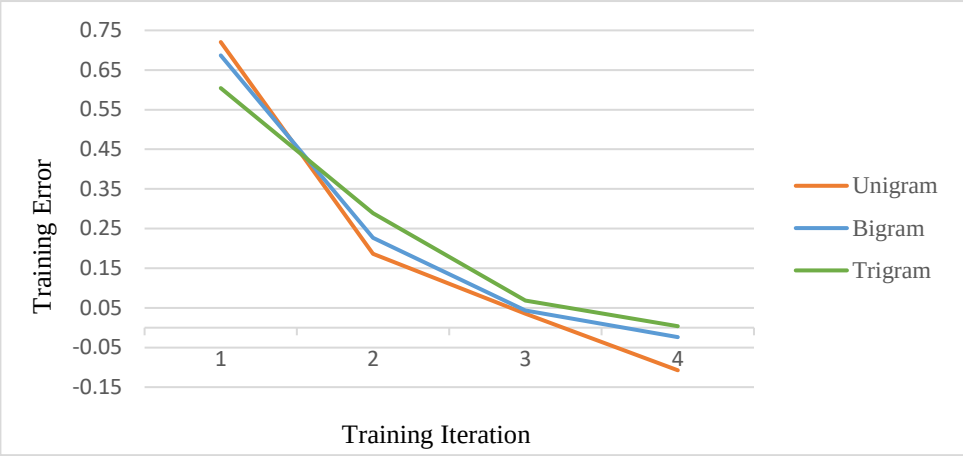

Figure 1: Graph showing the change in Prediction Error for each training iteration when updating the Weights Vector for the Unigram, Bigram and Trigram models. It is likely that over more iterations that the uni-gram model would oscilate around convergence, taking longer to converge than the bigram or trigram models.

Results

Computer specs: MacBook mid 2010, macOS Version 10.13.6,, 2.4GHz Intel Core 2 Duo, 16GB 1067MHz DDR3 RAM.

Feature Type	Test Data Accuracy	Most Positive Features
Unigram	79.5%	sometimes – performances – memorable – release – people – night – picture – hilarious – playing -- fun
Bigram	76.75%	feel good -- film takes -- very good -- best performance -- seven years -- throughout film -- one best -- movie goes -- very funny -- nothing short
Trigram	64.5%	shows us modern -- very good movie -- saving private ryan -- five minutes film -- well worth time -- done very well -- robert downey jr -- robert de niro -- film takes place -- first feature film

The top 10 positive features for each model do make sense for example, the unigram contains “hilarious”, “memorable” and “fun”. Bigram contains “very good” and “very funny” and Trigram includes “very good movie”, “well worth time”. All these terms we can see are positive terms, however; it is surprising that positive terms such as “good” do not feature more highly, this is because many positive terms can be used in negative ways such as “good for nothing”.

If the Trigram classifier was applied to reviews in other domains, the classifier would be able to predict sentiments but with low accuracy. Many of the terms found in other domains would not be included in this classifiers weight vector and many of the terms found in this classifiers weight vector would not be found in other domains. “Saving Private Ryan” is very unlikely to appear in restaurant reviews so the Trigram classifier does not generalise well. However; the Unigram and Bigram classifiers generalise much better terms such as ”very good” appear in all domains, therefore the sentiment predictions in other domains using these classifiers would still not be ideal but better than the trigram.

A better feature would be to use a human derived lexicon of words with defined weights as to how positive or negative the terns are. Terms could have further rules such if the term is in capitals then increase the magnitude of the polarity for that term by 2. The lexicon would contain terms valid in all domains, not domain specific terms like the classifiers produced in this lab. This would produce consistently accurate sentiment analysis over a wide domain of reviews.