COMP9444 Project Summary

Sentiment Analysis on Amazon Reviews

Seoyeon(z5248219), Heeseob(z5375037), Raymond(z5240051),

Jae(z5160611), Neham(z5375484)

I. Introduction

Our project revolves around the issue of natural language processing (NLP) in computers. In particular, our aim is to create a neural network model that can analyse and understand product reviews in Amazon to determine its sentiment (positive or negative) by making use of NLP.

Online shopping services like Amazon have become very prevalent these past years and businesses compete daily to keep and attract customers. The implementation of sentiment analysis will allow businesses and users alike to get a better understanding of customer feedback and the factors influencing the positivity or negativity of their ratings. Thus, providing guidance for businesses on the types of strategies to implement for addressing customer feedback.**Motivation:** It is vitally important for customer obsessed companies to validate customer reviews to provide best services, thereby a need to form a model to process sentiments of Customers.

II. Experimental Setup

Our project is inspired by a Kaggle post that had previously done sentiment analysis for Amazon reviews from multiple categories (Bittlingmayer, 2022). Since this sample size was too large for the scope of this project, we collected the sample files for both training and testing purposes and created a smaller dataset with only 200,000 samples for training, testing and validation split in (80:10:10 ratio respectively) altogether.

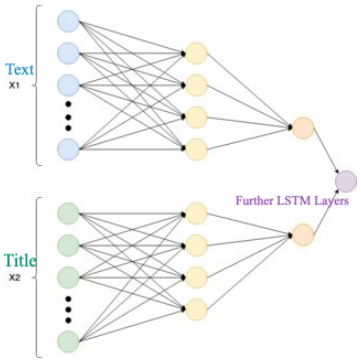
Key parameters in the mode included the text, title inputs, the learning rate, epochs, tokenisation padding length, optimizer, and loss function. We decided that accuracy and loss were the most important metrics and used them to evaluate our models. Loss was calculated using CrossEntropy for our 3-class model and BCE for our 2-class model. These parameters were chosen for their conventionality and it was decided to focus on data-preprocessing and model architecture to achieve great evaluation metric results.

In terms of the dataset, the key breakthrough made was that we chose to trim the words from the dataset with low density. This removed words that were most likely idiosyncratic words from individual reviews not useful for providing insight on sentiment. Given the nature of the dataset which would contain certain repeated emotional words, the words with relatively high density in the dataset were the words we decided to keep. Again, this proved to be the key in feeding our neural network with the critical words to determine accurate sentiment analysis and can clearly be seen in our results as an important factor in influencing model accuracy and loss.

III. Methods

Initially, we tried to set up a basic model to carry out 5-class classification from ratings of 1 to 5. We tried embedding techniques such as BERT, Word2Vec and Glove and using simple linear layers as used in assignment 1. However even the best of these iterations resulted in low accuracy topping 40% and we decided to first reduce the scope of the project to 3 classes.

The major breakthrough in our model architecture came from creating a multilayered LSTM. It involves preprocessing the Text and Title data and running them through separate layers which were then concatenated into a singular layer which was then fed into an output as seen in the diagram on the left. This greatly improved the accuracies of both the training, validation and test datasets and also had the benefit of reducing overfitting i.e the accuracy difference between datasets. We believe creating two layered channels helped train the neural network to recognise the key words from both Title and Text, hence the better results.



Furthermore we developed simple LSTM models which take individual Text and Title inputs each for comparisons. We then compared each of the three models using trimmed and non-trimmed datasets for a total of 6 scenarios.

IV. Results

Over the 6 scenarios we went through (trimmed/untrimmed) for three models (for each 2-class and 3-class model) we could clearly see that the two-input model using trimmed data was overall the best. It had the highest validation and test accuracy as well as the least overfitting. It should be noted for future reference that our 3 class models faced difficulty in classifying reviews with neutral sentiments. This was an issue that could be better resolved in the future. Otherwise, we engaged in hyper-parameter tuning with different epoch sizes, learning rates, layer depth and token padding length. Through trial and error we determined constant values that gave the best results which we included in our final model. The result for our 3 class model was a training accuracy of 78% with test and validation close to 65%. For our final 2 class model we saw a training accuracy of 90%, with test and validation at 87% apiece.

V. Conclusions

In the end we can draw conclusions that our model was challenged by a few non-trivial examples, for instance *“This product is quite good”* and *“Good product”*. Whilst both examples have the same meaning, one customer reviewer gave 3 stars while the other gave 4 stars, which made it difficult for the model to classify this as a Neutral or Positive sentiment. Further improvements can possibly be achieved with embedding GLOVE into the model, increasing the dataset size and enhancing data cleaning for neutral reviews. Finally, it is also better to show sentiments as a number between 0 to 1 (F1 score) rather than classifying it into 3 parts.

VI References/ Literature review

- Bittlingmayer, A. (2019). *Amazon Reviews for Sentiment Analysis*. Kaggle.com. from (https://www.kaggle.com/datasets/bittlingmayer/amazonreviews)
- Loye, G. (2019). *Long Short-Term Memory: From Zero to Hero with PyTorch*. FloydHub Blog. from(https://blog.floydhub.com/long-short-term-memory-from-zero-to-hero-with-pytorch/)
- Google Drive link for the dataset:
    - (https://drive.google.com/drive/folders/0Bz8a_Dbh9Qhbfll6bVpmNUtUcFdjYmF2SEpmZUZUcVNiMUw1TWN6RDV3a0JHT3kxLVhVR2M?resourcekey=0-TLwzfR2O-D2aPitmn5o9VQ) (Original Dataset)
    - https://drive.google.com/file/d/0Bz8a_Dbh9QhbZVhsUnRWRDhETzA/view?usp=sharing&resourcekey=0-Rp0ynafmZGZ5MflGmvwLGg (Resized Dataset)
- Glove
    - https://medium.com/mlearning-ai/load-pre-trained-glove-embeddings-in-torch-nn-embedding-layer-in-under-2-minutes-f5af8f57416a
    - https://edumunozsala.github.io/BlogEms/jupyter/nlp/classification/embeddings/python/2020/08/15/Intro_NLP_WordEmbeddings_Classification.html#About-the-data
    - https://coderzcolumn.com/tutorials/artificial-intelligence/how-to-use-glove-embeddings-with-pytorch#1
- PyTorch documentation
    - https://pytorch.org/docs/stable/data.html
    - https://pytorch.org/data/main/generated/torchdata.datapipes.iter.IterableWrapper.html
- Batch Size
    - https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/#:~:text=The%20batch%20size%20is%20a%20number%20of%20samples%20processed%20before,samples%20in%20the%20training%20dataset.
- Epoch tuning
    - https://machinelearningmastery.com/difference-between-a-batch-and-an-epoch/#:~:text=The%20batch%20size%20is%20a%20number%20of%20samples%20processed%20before,samples%20in%20the%20training%20dataset.