Group Coursework Submission Form

Specialist Masters Programme

Please list all names of group members: (Surname, first name) 1. Gopakumar, Kirtana 2. Sinkov, Evgeny 3. Ugale, Vaishnavi	4. Zahra, Hadeeka Tuj 5. 6. 7. **GROUP NUMBER:** **16**
MSc in: Business Analytics	
Module Code: SMM694	
Module Title: Applied Natural Language Processing (NLP)	
Lecturer: Dr Simone Santoni	**Submission Date:** 20-July-2022

Declaration:

By submitting this work, we declare that this work is entirely our own except those parts duly identified and referenced in my submission. It complies with any specified word limits and the requirements and regulations detailed in the coursework instructions and any other relevant programme and module documentation. In submitting this work we acknowledge that we have read and understood the regulations and code regarding academic misconduct, including that relating to plagiarism, as specified in the Programme Handbook. We also acknowledge that this work will be subject to a variety of checks for academic misconduct.

We acknowledge that work submitted late without a granted extension will be subject to penalties, as outlined in the Programme Handbook. Penalties will be applied for a maximum of five days lateness, after which a mark of zero will be awarded.

Marker's Comments (if not being marked on-line):

Deduction for Late Submission:

Final Mark: %

1. Introduction

IMDB is an online database of movies, with information on topics related to the same such as user-submitted movie reviews, trivia, top-10 lists, etc. The task at hand was to analyse 10755 reviews submitted by users, to train a machine learning model to predict if a movie review would be majorly upvoted (thus considered a “useful” review) or downvoted. This model is then harnessed to predict on 5071 unlabelled reviews. Thus the train-test data split is 68:32.

2. Data Exploration and Cleaning

To identify the steps for natural language processing of the IMDB movie review helpfulness dataset, the data was prima facie manually scrutinised for data inconsistencies to be cleaned.

The datasets were then cleaned through the following steps (See python notebook, section 2)-

1. Replacing unicode characters with ascii characters; the CSV file was riddled with unicode characters, which were removed using a combination of “ftfy” and “codecs” packages
2. Lowering the case of reviews
3. Identifying and removing website links within the reviews
4. Splitting contracted words such as “could’ve” to “could have”
5. Replacing slang words such as “gotta” to “got to” by scraping a popular slang database “no-slang”
6. An attempt was made through two language identification packages to identify and remove reviews which were not made in English. However, it was noted that certain poorly formatted reviews largely belonging to the “unhelpful” category were being mislabelled as non-English. Thus, this step was not adopted.

Primary data exploration of the train set showed a skewed data distribution of 24% unhelpful labels and 76% helpful labels (See appendix, Figure 1). Further, there was no apparent link between review length and its perceived helpfulness, since the average length of helpful reviews (188) and unhelpful reviews (134) were not drastically different.

It was noted that the reviews which were poorly formatted were somewhat more likely to be declared “unhelpful”. For instance, of the reviews starting with a lowercase letter, the following was noted-

Class	**Number of reviews**	**Percentage of total class**
Helpful	56	0.69%
Unhelpful	54	2.09%

Further, visual aids were utilised to better understand any underlying data patterns for this classification though a “scattertext” plot (See appendix, Figure 2). While this did not result in any deep insights, it was interesting to note that named entities with common movies seemed more likely to fall on the same side of helpfulness/unhelpfulness. For instance, “Kristen Stewart”, “Bella”, “Twilight”, “Vampires”, etc featured heavily in the unhelpful category. This helped further develop intuitions about the patterns within the dataset where reviews (and movies) revolving around a network of named entities such as the movie, its cast, director, etc performed similarly to each other.

3. Applying different NLP models

3.1 Machine Learning classification

A Logistic Regression model was selected first since it can be considered the baseline algorithm for classification tasks in NLP (Daniel and Martin, 2021). Using the Rule of thumb, the train dataset was split into train and test sets with an 80:20 ratio. Later, while training deep learning models the resulting train set was further split into train and validation sets with a ratio of 90:10. To verify that the train and test datasets were representative of the original dataset, data stratification was done while splitting. No reviews were imputed, synthesised, or cut to avoid losing knowledge or populating new reviews syntactically (Brownlee, 2020). Further, in terms of vectorization techniques, it was decided to begin with simple models and improve the same with sophisticated models.

TF IDF

Due to the anticipation that TF IDF would be helpful in identifying significant words/topics around which reviews would exist, which could then be used to determine whether a review is helpful or not, TF IDF vectorization was undertaken with max_features parameters equal to 100. The resulting F1 score on the set was 86.3%. (See python notebook, section 4.1)

LDA topic modelling

Next, each review was vectorized using topic modelling via Gensim LDA model, limiting the number of topics to 30. It was noticed that increasing topics to even 100 did not show a significant change in the model performance. The resulting coherence score was 0.547. The resulting F1 score after fitting logistic regression is the same as the one obtained by the TF-IDF model (86.3%). (See python notebook, section 3.3)

Word2vec vectorization

While the above mentioned models did not perform vectorization using any outside information about the words, utilising outside information through a pre-trained model was undertaken. To do so, the language model provided by Glove (huggingface.co, n.d.), which has been trained on 2 billion tweets with a vector length of 25 was used. By summing the vectors of each word, if the word was found in a dictionary, the model collected review-level vectors and used them to fit logistic regression. Yet again, the F1 score remained unchanged. (See python notebook, section 3.4)

3.2 Deep Learning classification

Neural network with 4 Dense layers

In an attempt to improve the F1 score past 86%, neural networks were enlisted to extract further insights from the reviews.

TF IDF vectors were created, and used as inputs to train the neural network with 4 Dense layers (See python notebook, section 4.1, Model 1). The resulting F1 score was 85%. (See Appendix, Figure 3)

Neural network with LSTM layer and pre-trained embedding_matrix

Up until this point, the order in which the words appear was ignored and treated text inputs as vectors. A neural network with an LSTM layer with a custom embedding matrix was designed in order to take advantage of both the sequential nature of texts and the meaning of individual words extracted from the glove twitter pre-trained model (see appendix, figure 4). To achieve it, two custom functions were created:

- get_train_vocabulary – to obtain the vocabulary out of pre-trained language model (Glove Twitter 25 in the case of our project)
- text_to_seq – similar to Keras tokenizer.texts_to_sequences. It converts text into a sequence of tokens based on the train vocabulary calculated using get_train_vocabulary function.

To ignore words that were left out as a result of padding, a layer called Masking was added. The F1 score was 86.6% after training the model on 20 epochs and predicting the test set labels. (See python notebook, section 4.1, Model 2)

Training a neural network using PyTotch

The TensorFlow Python library to train the classifiers was previously used and to make use of the PyTorch library, the code provided by Simone Santoni in the GitHub Repo was used (Simone Santoni, 2022). First, the class IMDB helpful reviews by inheriting PyTorch's Dataset class was created and the same code as Simone Santoni was implemented, except the learning rate was changed to 0.01. The accuracy score of 76.4 percent on the test set showed no improvement when compared to any previously implemented models. (See python notebook, section 4.2)

4. Choosing BERT as the final model:

After trying logistic regression and DL techniques such as Keras with various embedding techniques like TF-IDF and Keras embeddings, the accuracy of models does not see much of an improvement. The next best bet was the pretrained transformer model- BERT (Devlin, Chang, Lee and Toutanova, 2022), for the task. TF SavedModel (v3) is used as the pre-processing model to convert raw text to numeric input tensor that can be understood by the encoder. The first layer is the input layer that takes the preprocessed text as the input which will be encoded in the encoder layer and fed as input to the neural network layers. BERT is made of 12 hidden layers with a hidden size of 768 and dropout layers to avoid overfitting. Training the

model on our training dataset for 2 epochs gives a training accuracy of 75% , validation accuracy of 76% and F1-score of 0.865. The model is then tested on the testing dataset after removing unicodes and the output which has a probability of greater than 0.5 is classified as ‘helpful’ and less than 0.5 is classified as ‘unhelpful’.
Bert was chosen over LSTM even though it has a F1-score of 0.866 as it is a rather insignificant improvement, and also because BERT is an industrial standard. BERT combines bidirectional transformers and transfer learning which makes it much faster. With bidirectional transformers, the models learn from left to right and right to left simultaneously, unlike bi-drirectional LSTM which learn only in one direction at once. This enhances the ability of the model to understand the context of the text. Although the F1 score has not improved much, the model has much more scope of improvement if the reviews are more informative for feature extraction. Further, BerTopic can be then used to classify the reviews based on the topics (Angelov, 2022), i.e, the name of the movie or the cast. (See python notebook, section 4.1, Model 3)

5. **How the classifier could be improved:**

Through NLP modelling, it has been understood that additional data relating to the reviews, such as movie being reviewed, reviewer background, reviewer’s past reviews, etc could improve the performance of BERT model.

As such, inherent features of the review itself, such as its length, number of named entities contained, descriptive words such as “powerful”and “amazing”, or even its sentiment by itself could not easily be harnessed to predict review helpfulness to a high degree of accuracy. With most models performing with an f-1 score of around 86%, additional information which can be used to create a network of the underlying topics and entities about which the review has been made has the potential to push the models to perform better.

To test the intuition that reviews in itself were not generally able to be used to predict usefulness, a “Human Classifier” was created (See python notebook, section 6. A member of the group randomly pulled 80 reviews and removed their labels, and the group members attempted to predict the helpfulness of the review. Accuracy of this exercise was around 80%. Thus, without the context of other reviews of the same movie, it was difficult to gauge review helpfulness (Google Docs, 2022).

There is a great degree of homogeneity in the thought process of reviewers in relation to the movie being reviewed, its star cast, director, and setting. This is due to the powerful nature of social influence, since a consumer’s attitude toward a work of art is substantially influenced by that of others (Santoni, 2022). Simply put, people reviewing a movie were more likely to have a similar sentiment toward the movie, and the reviews that went against this collective sentiment were likely to be considered unhelpful. Thus, if the majority of reviewers had a positive sentiment toward the model, those reviews with a negative sentiment were more likely to be downvoted.

To validate this intuition, two popular movies “Titanic” and “The Godfather” were chosen. Sentiment toward the movie for each review was extracted through unsupervised machine learning. The “flair” package, which uses text embedding to predict sentiments was enlisted. Since flair has been pre-trained on IMDB reviews, this was an especially useful package. Due to the need for complete accuracy on this small dataset, sentiments were also manually labelled. It was noticed that the intuition held, majority reviewers expressed a positive sentiment for the movies, and such reviews were labelled ‘helpful’, while those going against this sentiment were labelled ‘unhelpful. (See appendix, Figure 3)

Appendix:

Figure 1. Category frequency histogram

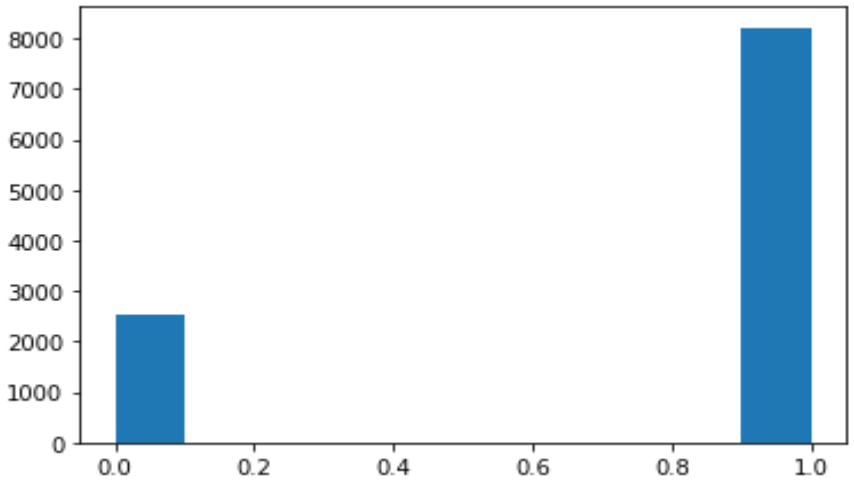


Figure 2. Frequently used words in each category

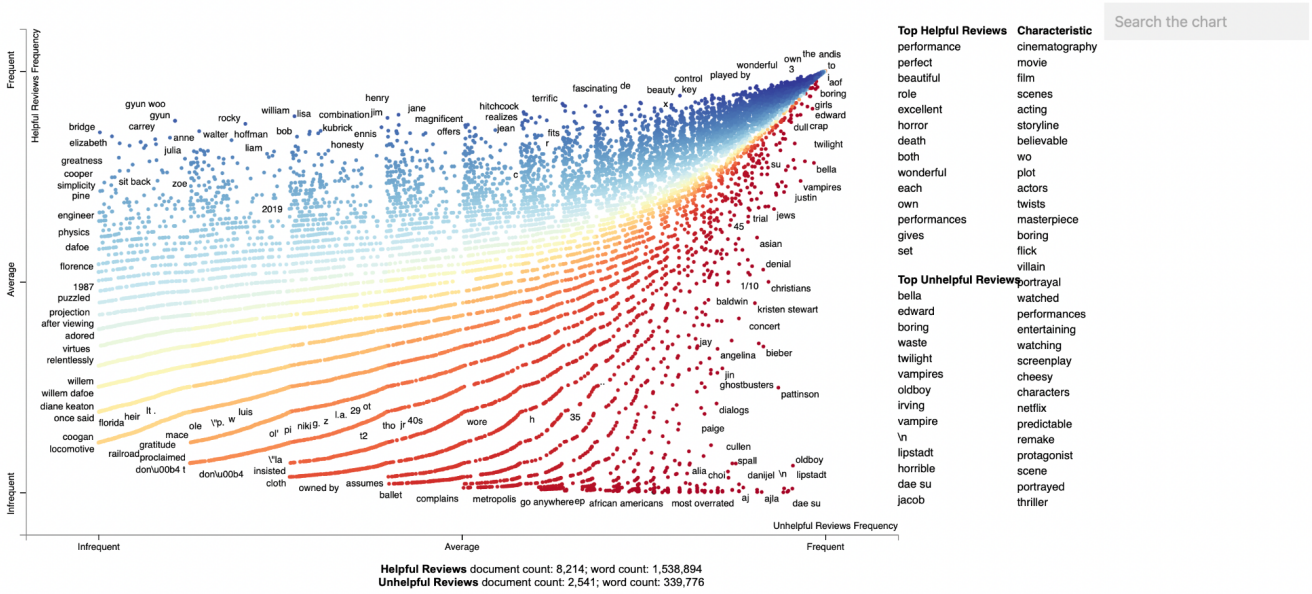


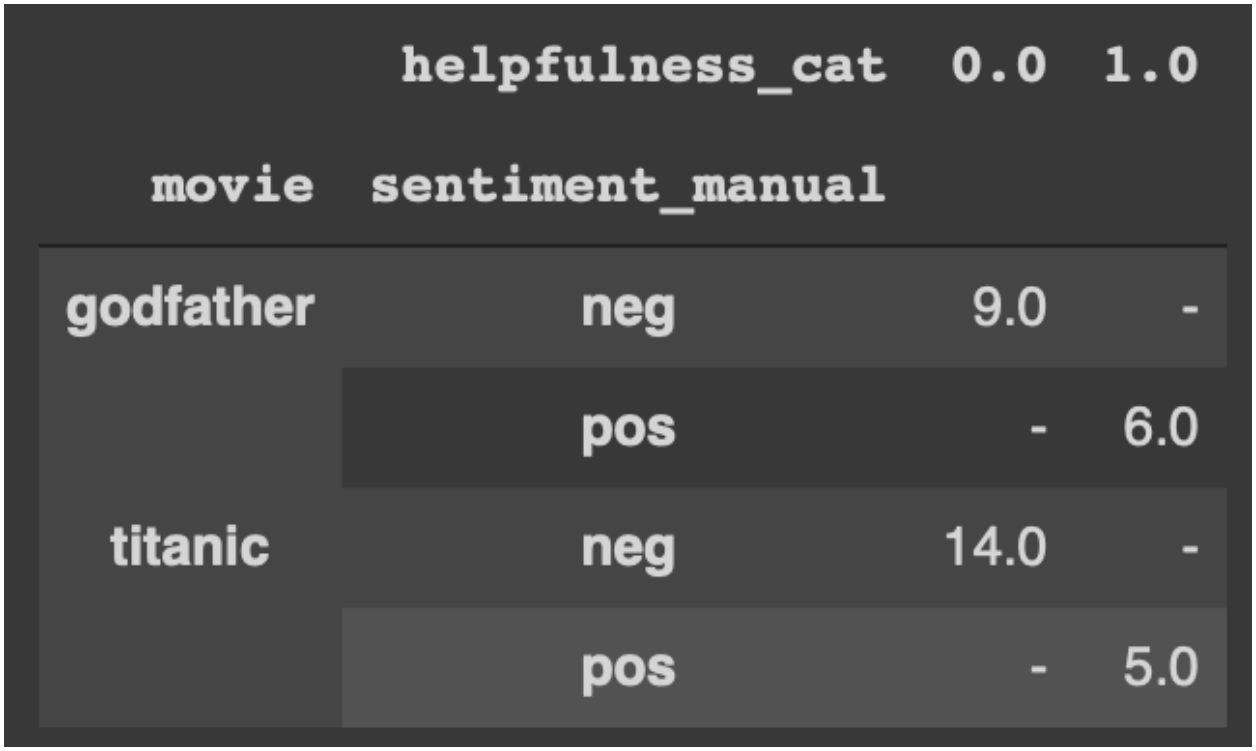
Figure 3. Deep learning model with TF IDF vectors as input

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 10)                3010

 dense_1 (Dense)             (None, 10)                110

 dense_2 (Dense)             (None, 10)                110

 dense_3 (Dense)             (None, 1)                 11

=================================================================
Total params: 3,241
Trainable params: 3,241
Non-trainable params: 0
_________________________________________________________________
```

Figure 4. Neural network with LSTM layer and pre-trained embedding_matrix

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 10)                1010

 dense_1 (Dense)             (None, 10)                110

 dense_2 (Dense)             (None, 10)                110

 dense_3 (Dense)             (None, 1)                 11

=================================================================
Total params: 1,241
Trainable params: 1,241
Non-trainable params: 0
_________________________________________________________________
```

Figure 5. Comparison of manual sentiment classification and flair model output



movie	sentiment_manual	helpfulness_cat 0.0	1.0
godfather	neg	9.0	-
	pos	-	6.0
titanic	neg	14.0	-
	pos	-	5.0

References:

Devlin, J., Chang, M., Lee, K. and Toutanova, K., 2022. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. [online] arXiv.org. Available at: <https://arxiv.org/abs/1810.04805> [Accessed 20 July 2022].

Angelov, D., 2022. Top2Vec: Distributed Representations of Topics. [online] arXiv.org. Available at: <https://arxiv.org/abs/2008.09470> [Accessed 20 July 2022].

Daniel, J. and Martin, J. (2021). Speech and Language Processing. [online] Available at: https://web.stanford.edu/~jurafsky/slp3/5.pdf.

Brownlee, J. (2020). Train-Test Split for Evaluating Machine Learning Algorithms. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/#:~:text=Stratified%20Train%2DTest%20Splits [Accessed 20 Jul. 2022].

huggingface.co. (n.d.). Gensim/glove-twitter-25 · Hugging Face. [online] Available at: https://huggingface.co/Gensim/glove-twitter-25 [Accessed 20 Jul. 2022].

Simone Santoni (2022) NLP-orgs-markets/textClassification/pytorch.py Available at: https://github.com/simoneSantoni/NLP-orgs-markets/blob/master/textClassification/pytorch.py

Google Docs. (2022). Human_classifier. [online] Available at: https://docs.google.com/spreadsheets/d/1eua9auvfp4DipMd2OQJQu5rUBwViypg6kR7J2YY2Cdw/edit#gid=1277495839 [Accessed 20 Jul. 2022].

Santoni, S. (2022). README. [online] GitHub. Available at: https://github.com/simoneSantoni/NLP-orgs-markets/blob/master/caseStudies/_1.md [Accessed 20 Jul. 2022].