BAYES BUSINESS SCHOOL
CITY UNIVERSITY OF LONDON

Group Coursework Submission Form

Specialist Masters Programme

Please list all names of group members: **1. Kartikey Bhalla** **2. Soumya Ogoti** **3. Wenxu Tian**	**4. Preeti Pothireddy** **5. George Krivosheev** **GROUP 7**
MSc in: Business Analytics	
Module Code: SMM694	
Module Title: Applied Natural Language Processing	
Lecturer: Dr. Simone Santoni	**Submission Date: Deadline** **Friday, 21 July 2023, 4:00 PM**

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:** ☐ %

SMM 694

NATURAL LANGUAGE PROCESSING

INTRODUCTION:

We handled a real word dataset, consisting of beer reviews from the Beer Advocate community. The training data contains 21,057 labelled reviews (okay, good and excellent), with no significant class imbalance. The test data contains 8,943 unlabelled reviews. The primary goal was to classify the test data beer reviews using train data into one of the three categories: (a) 0 or okay (b) 1 or good (c) 2 or excellent. Broadly, this report has been divided into three subsections:

(a) Data Preprocessing

(b) Model Building (+ Final Model Selection)

(c) Conclusion

STEP 1: DATA PREPROCESSING:

In general, data preprocessing is the first step in the pipeline of any Natural Language Processing system. We preprocessed the train and test datasets by lemmatizing the text, removing stop words, removing punctuations and special characters, removing numbers (and dates), removing emails and URLs, and lowercasing the text. Using Gensim phrases, we accounted for bigrams and trigrams. As the next step, we broadly categorized our models into two types based on their underlying architectures: (a) Machine Learning Models and (b) Deep Learning Models.

STEP 2: MODEL BUILDING:

1. Machine Learning Models:

For text classification, machine learning models have drawn lots of attention in recent years. Classic machine learning based models follow the two-step procedure: In the first step, some hand-crafted features are extracted from the documents. In the second step, those features are fed to a classifier to make a prediction (Minaee et al). We followed this standard procedure, and below is a broad overview of our selected set of features, classifiers and their justification:

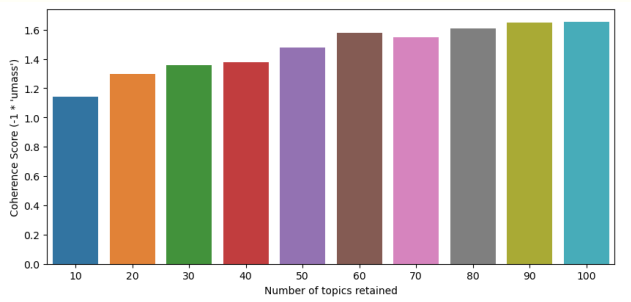
1.1 Features:

o. TFIDF:

TFIDF (Term Frequency and Inverse Document Frequency) is a statistical model which reflects how relevant a particular word is to a document. TFIDF penalizes frequently occurring words and assigns more weightage to words that are unique to a document (In BoW, every word is treated equally, and this can lead to high-dimensional sparse representations). We also incorporated n_grams and applied MinMaxScaler to reduce dimensionality.

o. Topic Modeling:

Topic modeling is a text-mining methodology for identifying topics in documents (Gupta, S. and Sharma, G, 2021). We used Latent Dirichlet allocation to fit the topic model. In an LDA topic model, each document is treated as a collection of topics, and each topic is treated as a collection of words. LDA provides a probabilistic framework for topic modelling, i.e. it estimates the probability of each word belonging to a particular topic and the probability of each document being associated with a specific topic. To choose the optimal number of topics, coherence score was calculated for k/topic_num ranging from 10 to 200 (at increments of 10). Higher the coherence score, the better. Although our maximum coherence was not achieved at k = 50, the difference was not significant. Also, this lowers the number of topics, leading to less sparse vectors and a more parsimonious representation.



o. Doc2Vec:

Doc2Vec is an extension or generalization of the Word2Vec algorithm. While Word2Vec generates numerical vector representations of words, Doc2Vec generates numerical vector representations of documents (irrespective of the length). Here, each document in the corpus is represented as a fixed-length vector in a continuous vector space. Doc2Vec captures semantic relationships and preserves contextual information.

1.2 Classifiers:

o. Logistic Regression: We used Logistic regression as the starter classification algorithm for beer review classification. This is because logistic regression is simple and easy to understand, and can be easily generalized to multiple classes (Susan Li, 2018).

o. Naive Bayes: Naive Bayes is a probabilistic learning method, and the variants of Naive Bayes are often used as baseline methods for text classification (Wang and Manning, 2012). They are faster than the linear model and are well suited for high-dimensional datasets (Naeem et al., 2022). For the purpose of our analysis, we chose Multinomial Naive Bayes, since the algorithm is suitable for multiple classes, and works well with text data that has a high number of features or is sparse.

o. Random Forest Classifier: Random Forest is a powerful ensemble learning method for text classification. RFs can cope with text-related challenges such as noise and sparsity, and show better performance for text classification (Jalal et al., 2022).

o. OneVsOne Classifier and OnevsRest Classifier: OvO and OvR reduce a multi-label classification into a binary classification problem. OvR compares each class against the rest, and on the other hand, OvO compares all plausible two-class combinations in the dataset.

o. Support Vector Machine: A Support Vector Machine finds an optimal hyperplane that best separates the data points of different classes in a high-dimensional feature space. SVM is the most resilient prediction model and is built on a statistical learning framework (Raza et al., 2021). SVMs use an overfitting protection, and are well suited for problems with sparse instances (Thorsten Joachims, 1998).

o. Voting Classifier: Finally, we also attempted a voting classifier, which uses a combination of multiple individual models. Voting ensembles are the simplest and effective method of combining individual classifiers output to improve classification performance (Raza et al., 2019).

Grid Search was used to arrive at the optimal set of hyperparameters. We built our machine learning models using a combination of the above mentioned features and classifiers, recording the validation macro-average F1 scores:

(However, it is important to note that we tried multiple other model combinations and model improvements, but for the purpose of simplicity and clarity, we narrowed down our focus to a few top models)

MACHINE LEARNING MODELS PERFORMANCE

CLASSIFIER	FEATURE	F1 VAL
Logistic Regression	TFIDF	0.546
Logistic Regression	Doc2Vec	0.396
Logistic Regression	Topic Modeling (TM)	0.472
Logistic Regression	TFIDF + Doc2Vec + TM	0.602
SVC	TFIDF	0.540
SVC	Doc2Vec	0.428
SVC	Topic Modeling (TM)	0.469
SVC	TFIDF + Doc2Vec + TM	0.163
Random Forest	TFIDF	0.508
Random Forest	Doc2Vec	0.397
Random Forest	Topic Modeling (TM)	0.462
Random Forest	TFIDF + Doc2Vec + TM	0.511

Multinomial NB	TFIDF	0.497
Multinomial NB	Doc2Vec	0.337
Multinomial NB	Topic Modeling (TM)	0.461
Multinomial NB	TFIDF + Doc2Vec + TM	0.490
OnevsOne Classifier	TFIDF + Doc2Vec + TM	0.606
OnevsRest Classifier	TFIDF + Doc2Vec + TM	0.594
Voting Classifier (LR,RF,Multinomial NB)	TFIDF + Doc2Vec + TM	0.596
Voting Classifier (LR,RF,Multinomial NB, OvO)	TFIDF + Doc2Vec + TM	0.581
Voting Classifier (LR,RF,Multinomial NB, OvO, OvR)	TFIDF + Doc2Vec + TM	0.601

2. Deep Learning Models:

For the text classification with neural networks, the following networks were trained; Bidirectional LSTM, FastText as a classifier, FastText word representations with BiLSTM classifier and BERT:

o. Bidirectional LSTM: An LSTM network has the ability to capture long-term relationships between words in a sentence. A Bidirectional LSTM (BiLSTM) was chosen as it can capture contextual information by processing the input sequence in both forward and backward directions, thereby enabling it to capture long-term dependencies and contextual nuances. It can handle sequential data well and is robust to varying sentence lengths. BiLSTM networks have been shown to outperform alternate architectures that use MLPs, CNNs, and LSTMs (Jang et al., 2020). We utilised the Bidirectional LSTM model with a Keras tokeniser and embedding layer for the classification of beer reviews. The input text was tokenized into individual words and pre-processed to remove stop words, numbers and punctuations, followed by lemmatisation. Reviews with only stop words, numbers or punctuations were removed.The Keras tokeniser tokenised the cleaned text into sequences of tokens. This was fed to the BiLSTM model with an embedding layer to return dense embedding vectors associated with the tokens after looking up the corresponding row in the embedding matrix for each token in the sequence. This enables the subsequent BiLSTM to process and learn the context and meaning of words better in beer reviews. With training and hyperparameter tuning (early callback set to avoid overfitting), the network had an f1 score of 53.93%.

o. FastText: The next model chosen was FastText because of its ability to capture subword information, which was not so in the BiLSTM model. This helps in extracting morphological features and handling out-of-vocabulary words (Bojanowski et al., 2017). Internally, a skip-gram model is used to predict word embeddings for the sub-words in a sentence which are then averaged to get a fixed-sized representation of the sentence. This is followed by a linear classifier that classifies the sentence. FastText can handle a wide range of data and is robust to noise and rare words. Hence only basic preprocessing was done to remove rows with empty or only stop words, numbers or punctuations. We first trained the model with its default configuration as a starting point and then employed the autotune feature to select the optimal parameters. FastText with default config achieved an f1 score of 53.75 while with auto tuner, it was 55.85%.

o. BiLSTM with FastText word representations: We then explored the effectiveness of using FastText embeddings on a BiLSTM model to combine the strengths of each (Khasanah, 2021). Here we additionally trained the FastText model on our training data to learn domain-specific word embeddings. Sentence embeddings (average of sub-word embeddings) were extracted from lemmatized sequences to obtain dense vector representations that contained semantic information of reviews. These were fed to a BiLSTM model and trained with the early callback to avoid overfitting. This model gave an f1 score of 56.4%. This combination worked well than using fasttext as a classifier.

o. BERT: The next network that we used to capture contextualised word embedding was a BERT (Bidirectional Encoder Representations from Transformers) model (Devlin et al., 2018). It is a start-of-the-art model that has shown good performance for various NLP tasks. BERT models, based on the Transformer architecture, leverage attention mechanisms to consider the entire context of a word when generating its representation. Prior research shows that BERT uses stop words and punctuation to learn nuances and intricacies of language (Qiao et al., 2019). Hence, we only apply basic preprocessing, removing rows which were empty. A pre-trained BERT tokeniser based on WordPiece (Song et al., 2020) was used to tokenise a review into a sequence of subword units. This generates contextualised embeddings which is fed into the classification layer. First, a model with fixed padding is used which resulted in long training times. This resulted in an f1 score of 60.32% after 2 epochs. To improve computational efficiency, we adopted the smart padding (McCormick, 2020) technique that involves grouping input sequences of similar lengths together, minimising the need for excessive padding. This resulted in quicker training times. The model was fine-tuned using an appropriate optimiser and learning rate scheduler on our training data, this is crucial to capture the nuances/intricacies of the target task as BERT is trained on a large, generic corpus. We trained it on 4 epochs (suggested 2-4 for BERT) (Devlin et al., 2018) and stopped here as the validation loss (0.83) just started to increase to avoid overfitting. An f1 score of 61.3% was achieved.

DEEP LEARNING MODELS PERFORMANCE

DL Classifier	Embedding	F1 Val
Bidirectional LSTM	Keras Tokeniser + Embedding	0.5393
Fast Text Classifier	Fast Text Embedding	0.558
Bidirectional LSTM	FastText Embedding	0.564
BERT Optimised (Smart Padding)	Wordpiece Tokeniser + BERT Embedding	0.613
BERT with Fixed Padding	Wordpiece Tokeniser + BERT Embedding	0.603

Note: For the purpose of fair evaluation and optimal model selection, care has been taken to ensure that the models are comparable:

o. Common preprocessing step: Applied the same preprocessing step across all the models. Consistency in preprocessing helps eliminate variations that might impact model performance.

o. Common Evaluation Metric: Macro-average F1 score has been used for evaluation across all models. Here, our underlying assumption is that all the classes have equal weight.

o. Common Train Validation Split: We followed a consistent data splitting strategy across all models (80:20)

o. Random seed: Where applicable, to eliminate the impact of randomization, the same random seed has been used.

3. Choosing Final Model:

BERT Optimised (smart padding) performed well compared to the models and is our final model. One reason for this could be due to the contextual understanding capability by leveraging its self-attention mechanism, which allows it to capture subtle meanings and nuances better. It can also handle out-of-vocabulary words better (Song et al., 2021). Also, because of BERT's pre-training on a large and diverse corpus, it helps generalise well to various domains, here beer reviews. BERT's pre-training followed by fine-tuning on the train data allowed it to adapt to the task at hand and align its representations with specific characteristics of beer reviews. Of the other methods, the ones that used a combination of hand-crafted features with ensemble classifiers came close to the performance of BERT. Along with it having the highest f1 score, the BERT model's better generalization capability motivates us to choose it as our final model.

To access our saved BERT Optimised (Smart Padding) Model:

https://drive.google.com/drive/folders/1n--UhRrl_ZvLUfJssD7LUSywOlC_aw_c?usp=sharing

CONCLUSION:

Overall, while we achieved a decent F1 score, there is always scope for improvement. A larger dataset of beer reviews has the potential to improve the performance of BERT as it would allow the model to learn domain-specific vocabulary and context.

REFERENCES:

- Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. (2020). Deep Learning Based Text Classification: A Comprehensive Review. 1, 1 43 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

- Wang, S. and Manning, C. (2012). Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. [online] Association for Computational Linguistics, pp.90–94. Available at: https://aclanthology.org/P12-2018.pdf.

- Xu, Shuo & Li, Yan & Zheng, Wang. (2017). Bayesian Multinomial Naïve Bayes Classifier to Text Classification. 347-352. 10.1007/978-981-10-5041-1_57.

- Raza, A., Faiz-Ur-Rehman, Bilal, M. and Rauf, M. (2021). COMPARATIVE ANALYSIS OF MACHINE LEARNING ALGORITHMS FOR FAKE REVIEW DETECTION. International Journal of Computational Intelligence in Control Copyrights @Muk Publications, [online] 13(1). Available at: https://www.mukpublications.com/resources/19.%20Asif%20Raza1_pagenumber.pdf.

- Joachims, T. (1998). Text categorization with Support Vector Machines: Learning with many relevant features. Machine Learning: ECML-98, [online] pp.137–142. doi:https://doi.org/10.1007/bfb0026683.

- Jalal, N., Mehmood, A., Choi, G.S. and Ashraf, I. (2022). A novel improved random forest for text classification using feature ranking and optimal number of trees. Journal of King Saud University - Computer and Information Sciences, 34(6), pp.2733–2742. doi:https://doi.org/10.1016/j.jksuci.2022.03.012.

- Li, S. (2018). Multi-Class Text Classification Model Comparison and Selection. [online] Towards Data Science. Available at: https://towardsdatascience.com/multi-class-text-classification-model-comparison-and-selection-5eb066197568.

- Joseph SJ, Robbins KR, Zhang W, Rekaya R (2010) Comparison of two output-coding strategies for multi-class tumor classification using gene expression data and Latent Variable Model as binary classifier. Cancer Inform. doi: 10.4137/cin.s3827.

- Gupta, S. and Sharma, G. (2021). Topic Modeling in Natural Language Processing. International Journal of Engineering Research & Technology, [online] 10(6). doi:https://doi.org/10.17577/IJERTV10IS060250

- Bojanowski, P., Grave, E., Joulin, A. and Mikolov, T., 2017. Enriching word vectors with subword information. Transactions of the association for computational linguistics, 5, pp.135-146.

- Song, X., Salcianu, A., Song, Y., Dopson, D. and Zhou, D., 2020. Fast wordpiece tokenization. arXiv preprint arXiv:2012.15524.

- Jang, B., Kim, M., Harerimana, G., Kang, S.U. and Kim, J.W., 2020. Bi-LSTM model to increase accuracy in text classification: Combining Word2vec CNN and attention mechanism. Applied Sciences, 10(17), p.5841.
- Qiao, Y., Xiong, C., Liu, Z. and Liu, Z., 2019. Understanding the Behaviors of BERT in Ranking. arXiv preprint arXiv:1904.07531.
- Devlin, J., Chang, M.W., Lee, K. and Toutanova, K., 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
- McCormick C., 2020. Smart Batching Tutorial – Speed Up BERT Training. URL (accessed on 20-7-2023): https://mccormickml.com/2020/07/29/smart-batching-tutorial/
- Khasanah, I.N., 2021. Sentiment classification using fasttext embedding and deep learning model. Procedia Computer Science, 189, pp.343-350.