Sentiment Classification on Movie Reviews

s2333722 **s2447566** **s2449242**

Abstract

With the growth of data quantities and computational power, machine learning alongside natural language processing has risen as one of the most dominant tool for sentiment analysis. Allowing for more efficient and accurate information extraction on how people reacts to certain entities. In this report, we explore several techniques of data preparation, feature extraction, and machine learning model to build an optimal sentiment analysis model for movie reviews. We first apply it to a simple binary sentiment classification to observe how it perform and then scale it to a multiple levels of sentiment. We found that based on our results, a pretrained Transformers model RoBERTa without any cleaning on the text used as a feature extractor significantly outperforms other methods on all models with best model for binary sentiment classification using logistic regression achieves 87.06% accuracy and 87.06% macro-averaged precision and in multiclass sentiment classification it achieves 52.58% accuracy and 52.53% macro-averaged precision.

1 Introduction

Sentiment analysis (SA) is a study of people's sentiments or emotions towards entities such as products, topics, etc. It allows us to extract people's mood on certain entities to allow for more actionable knowledge which can be used to understand, predict, and explain certain social phenomena. This is especially beneficial to businesses or industries since they can make better decisions on how to improve their services or products on how the public react to it. Due to the exponential growth of data and computational power, machine learning (ML) combined with natural language processing (NLP) has risen as one of the most dominant method to do SA due to it enabling more efficient and advanced analytics [11, 24].

1.1 Related Work

There have been a lot of researches done to apply ML-based techniques for SA. In this section we will review two studies related from application using conventional ML to a deep learning-based (DL) model to provide context on why we have selected these techniques in this project. The impact of feature extraction method is explored by [1]. They evaluate several techniques for feature extraction such as Term Frequency-Inverse Document Frequency (TF-IDF) and N-gram or Bag of Words (BoW) models where it is a language model that is based on the frequency of N number of words grouped together on a Twitter SA dataset. They tested these features extraction methods on several classification algorithms including Logistic Regression (LR), Naive Bayes (NB), Support Vector Machines (SVM), Decision Tree (DT), and k-Nearest Neighbor (kNN). Their findings show that overall, TF-IDF provides the best result with 3-4% improvements across all models they have evaluated with LR as the best classification algorithm overall.

A study on DL-based SA have been explored by [15] in financial domain where it predicts people's sentiment on stock price changes. They tested several ways to represent the text including lexical, statistical, word encoder, sentence encoder, and current state-of-the-art (SOTA) Transformers model. The experiments with conventional ML models are conducted using SVM and Extreme Gradient Boosting (XGBoost) while DL models are conducted with Dense Neural Network (DNN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and Transformers. Specifically for Transformers, they fine-tune the existing pretrained model by adding one dense layer after the last hidden state. Evaluation on combination of Financial Phrases and SemEval-2017-TASK5 shows the Transformers model outperforms other methods and concludes that a distilled version of the Transformers model will perform comparable to their bigger counterpart with more efficiency.

1.2 Objective

Based on the description, we are tasked to build a sentiment classifcation model based on movie reviews dataset introduced in [21] called Stanford Sentiment Treebank (SST). It consists of over 12K total sentences and 215K unique phrases extracted from Stanford parser. It contains a continuous sentiment score between 0 and 1 which can be discretized into 5 categories as shown in Table 1.

Table 1: Label Description

Label	Index	Sentiment Score Range
Very Negative	0	$[0, 0.2]$
Negative	1	$(0.2, 0.4]$
Neutral	2	$(0.4, 0.6]$
Positive	3	$(0.6, 0.8]$
Very Positive	4	$(0.8, 1.0]$

This dataset is hard to model given multiple degrees of sentiment which may provide some ambiguity between some labels such as very negative and negative. Hence, we formulate some sequences of problems to solve:

1. How do we simplify the original problem to reduce the ambiguity between labels?
2. Is the dataset appropriate and balanced enough to simplify the task?
3. What words are the most prevalent between the labels? Are they important?
4. What techniques are suitable to build the SA model?
5. How does the techniques we have chosen perform when scaled to multiclass classification?

We will evaluate several techniques for feature extraction and classification which we have mentioned in Section 1.1 will be explained on summary in Section 4.

2 Data Preparation

Different data preparation approaches were used dataset as a preprocessing step. The rationale behind such approaches is manifold: shorter training times, enhanced generalization by reducing overfitting, and greater accuracy. We are using SpaCy [5] library to handle NLP specific task like lemmatization and stopwords removal. We brief several step-by-step methods we have done to do data preparation.

Text cleaning: It is essential to simplify our model such that we want unnecessary words and characters that are not important for the model to learn to be removed. We removed punctuation, numbers and special characters from the movie reviews. [1] We also normalize certain non-english character like "ö" to "o". Furthermore we convert all the characters with its lowercase form since it is important especially on statistical feature extraction due to it being very case-sensitive.

Stop words removal: Stop word is a common word that appear but do not add any understanding. Words such as “a” and “the” are examples. These words also appear very frequently, become dominant in your analysis, and obscure the meaningful words [1, 9, 15]. Furthermore, the words "movie", "film" and "character" were removed from the dataset since our findings in Section 3 has shown that these two words occur too frequently in the data. Therefore, we consider those also as stop words and do not give further knowledge about a sentence.

Lemmatization: It is the process of assembling the inflected parts of a word such that they can be recognized as a single element, called the words lemma or its vocabulary form. It is defined as an

[1]Character normalization is based from `https://github.com/singhalprerana/SST_data_extraction` implementation for SST’s preprocessing.

algorithm technique of finding the lemma of a word which is a root word rather than a root stem. It is based on the intended meaning the word is trying to convey [1, 9].

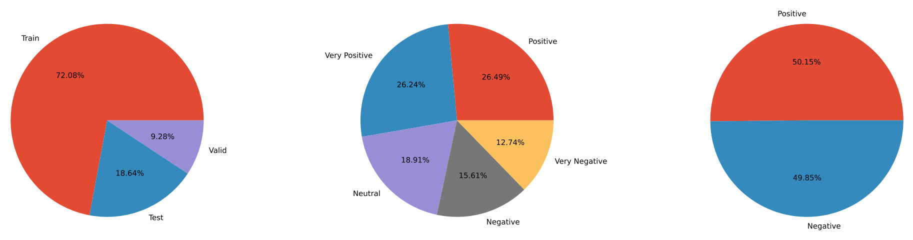
Target Binarization: To solve our first problem formulation in Section 1.2, we are able to simplify the problem into a binary classification by thresholding the sentiment score higher than 0.5 to be the positive class and vice versa for the negative since the sentiment score are continuous values between zero and one.

3 Exploratory Data Analysis

In this section we will explore the dataset to find insight on the kinds of processes the dataset may require. We are using Matplotlib [6] and Seaborn [22] for data visualization while Pandas [17] is used for handling tabular dataset processing.

3.1 Subset and Label Distribution

To determine whether the dataset is well split and its label are balanced or vice versa, we visualized the distribution as shown in Figure 1. Based on Figure 1a, we observe that we have approximately 8600 training, 1100 testing, and 2200 validation samples. Figure 1b shows the original label distribution, we see that the dataset is fairly balanced with slightly more samples on the positive and very positive label. Figure 1c shows similar distribution when the label is binarized, we observed that after binarizing the score, the distribution is almost completely balanced. Since the dataset is almost balanced, we do not see any reason to apply oversampling or undersampling methods which is intended to reduce the impact of imbalanced label distribution by increasing the underrepresented samples or reducing the overrepresented samples. Thus solving Section 1.2 second problem formulation concluding that the data is appropriate and balanced enough to be simplified into a binary SA problem.



(a) Subset (b) Multiclass (c) Binarized

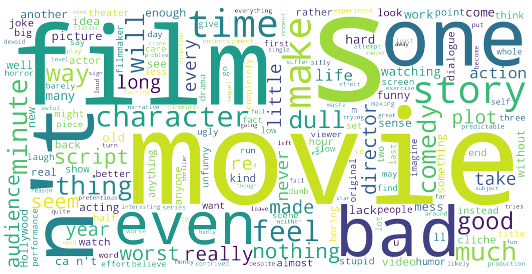
Figure 1: Subset and Label Distribution

3.2 Most Frequent Words

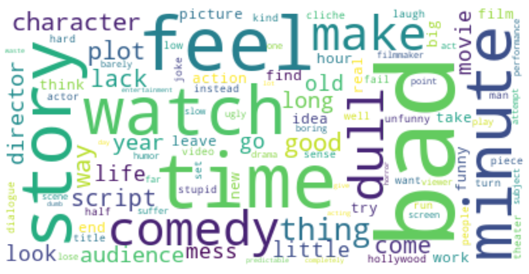
We explore the most frequent words within the dataset. We sampled two label extrema and visualize the word's frequency in a WordCloud [16] as shown in Figure 2. We observe that both pre-cleaned sample are dominated with words such as "movie", "film", and abbreviations such as "n't". While in the context of aspect-based SA these words are essential since these words could be represent aspect. The task is specifically intended for movie review sentiment analysis. Hence, we decided to consider these words as stopwords and removed it on the cleaned data. Thus answering our third problem formulation in Section 1.2.

3.3 Sentence Length

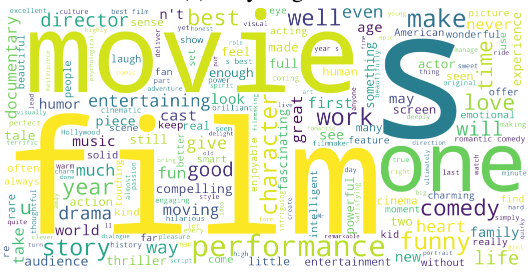
We explore the distribution of sentence length on each classes within the dataset. The sentence length is based on the number of tokens within a sentence. From Figure 3a's, we observe that the number of tokens are usually around 10-25 within a single sentence on each classes. Based on the 1.5IQR rule, we also observe that possible outliers tend to have token count greater than 45 tokens. After cleaning the text as, the number of tokens as shown in Figure 3b, we have reduced the number of tokens where the maximum are slightly above 25 tokens. The distributions between tokens also stay uniform



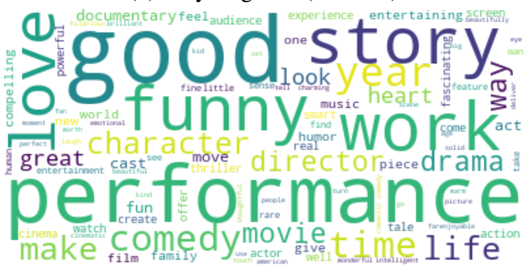
(a) Very Negative



(b) Very Negative (Cleaned)



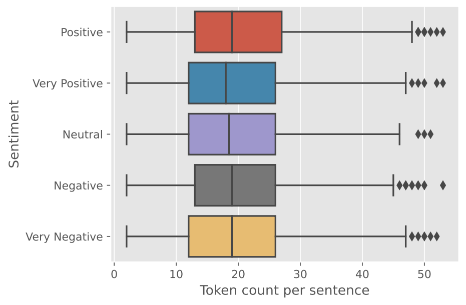
(c) Very Positive



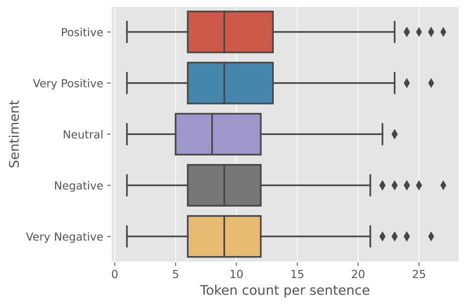
(d) Very Positive (Cleaned)

Figure 2: Top-100 WordCloud comparison between pre-cleaned and post-cleaned dataset

compared to pre-cleaned sentences although now the possible outlier is above 20 tokens. Since there are not much tokens on each sentence, we do not consider using document-level embedding such as Doc2Vec [10] that have been evaluated by [15] as an appropriate choice in this situation.



(a) Before cleaning



(b) After cleaning

Figure 3: Box plot based on the number of tokens per sentence within the dataset. We separate the distribution based on the sentiment of each sentence.

4 Methodology

In this section we will provide brief summary on the methods that we are going to use for building our SA model as a building block to solve our fourth problem formulation from Section 1.2.

4.1 Feature Extraction

Bag of Words (BoW) is one of the most common methods for categorization of text and objects. In the context of NLP, it records the number of occurrences of each bag that is created for each instance type or word disregarding the order of the words or the grammar [20]. BoW suffers from feature vanishing where it will give more weight to frequently shown words like stopwords which results in the lost of essential features [15]. Hence when using BoW, we need to be careful during text cleaning. We are using Scikit-Learn's [19] implementation for BoW which automatically does tokenization, lowercasing, and stopwords removal.

Term Frequency-Inverse Document Frequency (TF-IDF) is a statistical-based method which is similar to BoW but solves the feature vanishing issue since it reweights the token frequency in the sentence with the token frequency within the entire corpus. It calculates the term frequency which is the token frequency on a sentence and inverse document frequency to penalize if the token appears to frequently in more sentences [1, 15]. Same as BoW, we are using Scikit-Learn's [19] implementation for TF-IDF which does have the same automatic text processing as the former.

FastText is a word vector similar to Word2Vec [14] that is proposed by Facebook AI Research (FAIR) [3, 8] to tackle generalization on out of vocabulary (OOV) words. Unlike Word2Vec which does word-level embedding, FastText uses subword-level embedding where each character is model with an n-gram representation. This allows for better embedding training on smaller dataset and better generalization especially on structurally richer languages such as Arabic and Turkish [15]. We use the implementations provided in Gensim [26] to extract the word vector and vectors pretrained on Common Crawl corpous provided by [3, 8]. To feed the extracted features to the classifier, we fetch the embeddings for each word in a sequence and then take the average w.r.t. the sequence such that we have a 300 dimensional features as an input.

A Robustly Optimized BERT Pretraining Approach (RoBERTa) [12] is a variation of Bidirectional Encoding Representation from Transformer (BERT) [4] proposed by FAIR. It offers more optimized training procedure for BERT by introducing dynamic masking, removing Next Sentence Prediction (NSP), larger mini-batches, and larger byte-level Byte Pair Encoding. These procedures allows RoBERTa to outperform BERT in almost all downstream task [15]. We use the existing implementation for many popular Transformers models from Huggingface [23] implemented on PyTorch [18] which is pretrained by [13] on diachronic Twitter data with continual learning. We use the 512 dimensional features extracted from the pooler's output consisting to feed as the classifier's input.

4.2 Classification Algorithm

In this section we explain what classification algorithms we are applying to do our SA task. All of the implementations for the classifiers are done using Scikit-Learn [19] and tuned with Optuna [2].

Logistic Regression (LR) is a discriminative linear ML model similar to linear regression that is specifically used for classification. It analyzes the relationship between multiple independent variables and a categorical dependent variable, and estimates the probability of occurrence of an event by fitting data to a logistic curve [1, 7]. **Support Vector Machines (SVM)** is a hyperplane-based model which is efficient in high dimensions [1, 19]. The main idea of SVM is to find an optimal hyperplane of test dataset as decision surface and make the edge maximized between positive and negative examples of separation. Then the optimal hyperplane is used to perform classification on an unseen data [25]. **Multilayer Perceptron (MLP)** is an Artificial Neural Network (ANN) that learns a function to approximate certain output based on the data it trains on and the cost function it is optimizing. In the context of classification task, it can be considered as a more advanced LR model that learns a non-linear relationship with multiple hidden layers. It has higher level of abstraction than LR since it has multiple layers to extract and learn representations [19].

5 Results and Discussion

In this section we provide our evaluation on the test set. We evaluate our models with accuracy and macro-averaged precision. We chose macro-averaged precision as our additional metrics since we want our model to have as few false positives as possible. We provide only the best results for each feature extractions in this section. All of the best parameter search results that have been evaluated using the validation set will be shown in Appendix [A. To ensure more fair follow up comparison, we take the existing models for binary classification from Table 2 and re-evaluate it again for multiclass classification as shown on Table 3.

5.1 Binary Classification

We show that in Table 2, RoBERTa and FastText consistently outperforms their statistical counterpart. We observed that our text cleaning approach performs worse than the non-cleaned counterpart. We would like to note that a "Cleaned Text" for both BoW and TF-IDF in this case is just the use of lemmatization and punctuation removal since the implementation in Scikit-Learn by default does tokenization, lowercasing, and stopwords removal. We believe that it performs worse after lemmatization on the statistical methods due to lemmatization rule removing certain characters from a word that is already in its base form. This has worse effect on both FastText and RoBERTa due to its effect on removing adverbs which could be important to represent the contextual meaning which FastText and RoBERTa can infer since it also able to capture the semantic representation. The result overall is achieved by RoBERTa and LR with 87.06% accuracy and 87.06% macro-averaged precision. Thus answering fourth question in Section 1.2.

Table 2: Evaluation result performed on the test set for binary classification. **Bold** represents the best result overall while *Italic* represents the best score on each feature extraction.

Feat Ext.	Model	Cleaned Text	Accuracy (%)	Macro-Avg. Precision (%)
BoW	LR	✗	*75.97*	*75.95*
BoW	LR	✓	75.84	75.82
TF-IDF	SVM	✗	*76.88*	*76.94*
TF-IDF	SVM	✓	74.80	75.03
FastText	MLP	✗	*78.28*	*78.26*
FastText	SVM	✓	75.70	75.91
RoBERTa	LR	✗	**87.06**	**87.06**
RoBERTa	LR	✓	77.33	77.31

5.2 Multiclass Classification

After we have shown in Section 5.1 that we are able to build a binary SA model that performs well. We take the best techniques from Table 2 w.r.t. to each feature extraction methods and scaled up the problem to the original problem which is a multiclass classification. Unfortunately, our results from Table 3 shows that it performs not as optimal as on the binary task. Although RoBERTa and LR still outperforms other methods by a noticable margin with 52.58% accuracy and 52.53% macro-averaged precision, we still think this is still not ideal to do SA since it will on approximately 50% of the time predict the wrong sentiment. Thus answering out final problem formulation from Section 1.2 that, although the models perform sufficiently well in binary classification task, it still does not able to scale well to a multiclass classification task which we believe is due to ambiguous distinction between sentences in "very" and non-"very" label.

Table 3: Evaluation result performed on the test set for multiclass classification. **Bold** represents the best result across all feature extraction methods.

Feat Ext.	Model	Cleaned Text	Accuracy (%)	Macro-Avg. Precision (%)
BoW	LR	✗	39.23	38.52
TF-IDF	SVM	✗	41.64	40.07
FastText	MLP	✗	43.67	47.58
RoBERTa	LR	✗	**52.58**	**52.53**

6 Conclusion

In this report, we present our approach on building a SA model for movie reviews. We show that based on our results, a pretrained Transformers such as RoBERTa and word vectors such as FastText are very capable as a feature extractor to do SA because these methods also learn the word's semantics. Our result on cleaned data does not perform well since it does not work well particularly on FastText and RoBERTa since we believe that we lose the contextual meaning of the sentence by using lemmatization. We also show that for multiclass sentiment, we still cannot build a well performing model due to the ambiguous distinction between "very" and non-"very" label. We would like to see further study on other text cleaning techniques such as dictionary-based and synonym-based word normalization to improve upon our results further.

References

[1] Ravinder Ahuja, Aakarsha Chug, Shruti Kohli, Shaurya Gupta, and Pratyush Ahuja. The impact of features extraction on the sentiment analysis. In *Procedia Computer Science*, volume 152, pages 341–348. Elsevier B.V., 2019.

[2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2623–2631, New York, NY, USA, 2019. Association for Computing Machinery.

[3] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[5] Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. spaCy: Industrial-strength Natural Language Processing in Python. 2020.

[6] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[7] Park Hyeoun-Ae. An introduction to logistic regression: From basic concepts to interpretation with particular attention to nursing domain. *jkan*, 43(2):154–164, 2013.

[8] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[9] Divya Khyani, BS Siddhartha, NM Niveditha, and BM Divya. An interpretation of lemmatization and stemming in natural language processing. *Journal of University of Shanghai for Science and Technology*, 2021.

[10] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1188–II–1196. JMLR.org, 2014.

[11] Alexander Ligthart, Cagatay Catal, and Bedir Tekinerdogan. Systematic reviews in sentiment analysis: a tertiary study. *Artificial Intelligence Review*, 54(7):4997–5053, 2021.

[12] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[13] Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and José Camacho-Collados. Timelms: Diachronic language models from twitter. *CoRR*, abs/2202.03829, 2022.

[14] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In Yoshua Bengio and Yann LeCun, editors, *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

[15] Kostadin Mishev, Ana Gjorgjevikj, Irena Vodenska, Lubomir T. Chitkushev, and Dimitar Trajanov. Evaluation of Sentiment Analysis in Finance: From Lexicons to Transformers. *IEEE Access*, 8:131662–131682, 2020.

[16] Layla Oesper, Daniele Merico, Ruth Isserlin, and Gary D Bader. Wordcloud: a cytoscape plugin to create a visual semantic summary of networks. *Source code for biology and medicine*, 6(1):7, 2011.

[17] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[18] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[19] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[20] Wisam A. Qader, Musa M. Ameen, and Bilal I. Ahmed. An overview of bag of words;importance, implementation, applications, and challenges. In *2019 International Engineering Conference (IEC)*, pages 200–204, 2019.

[21] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[22] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[23] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Perric Cistac, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-Art Natural Language Processing. pages 38–45. Association for Computational Linguistics, 10 2020.

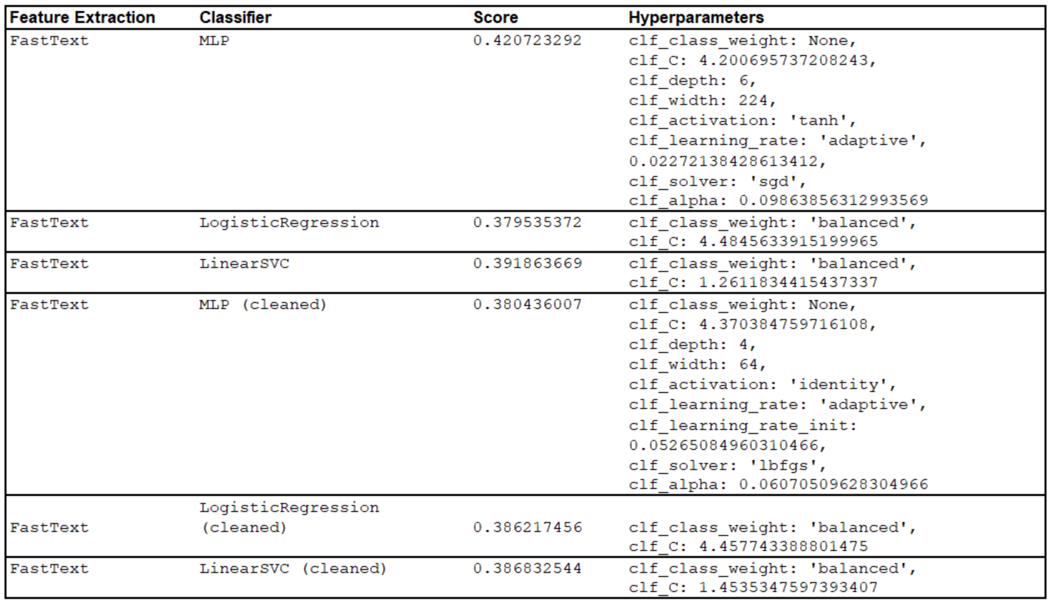
[24] Lei Zhang, Shuai Wang, and Bing Liu. Deep learning for sentiment analysis: A survey. *WIREs Data Mining and Knowledge Discovery*, 8(4):e1253, 2018.

[25] Yongli Zhang. Support vector machine classification algorithm and its application. In Chunfeng Liu, Leizhen Wang, and Aimin Yang, editors, *Information Computing and Applications*, pages 179–186, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[26] Radim Řehůřek and Petr Sojka. Software Framework for Topic Modelling with Large Corpora. Proceedings of LREC 2010 workshop New Challenges for NLP Frameworks, pages 45–50, Valetta, MT, 5 2010. University of Malta.

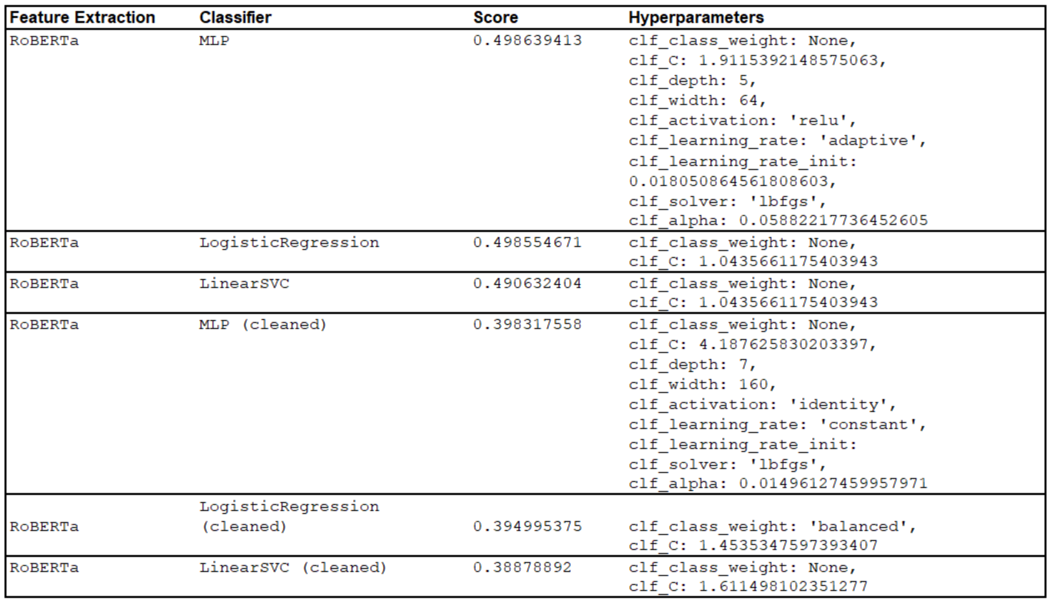
A Hyperparameter Search Result

A.1 FastText



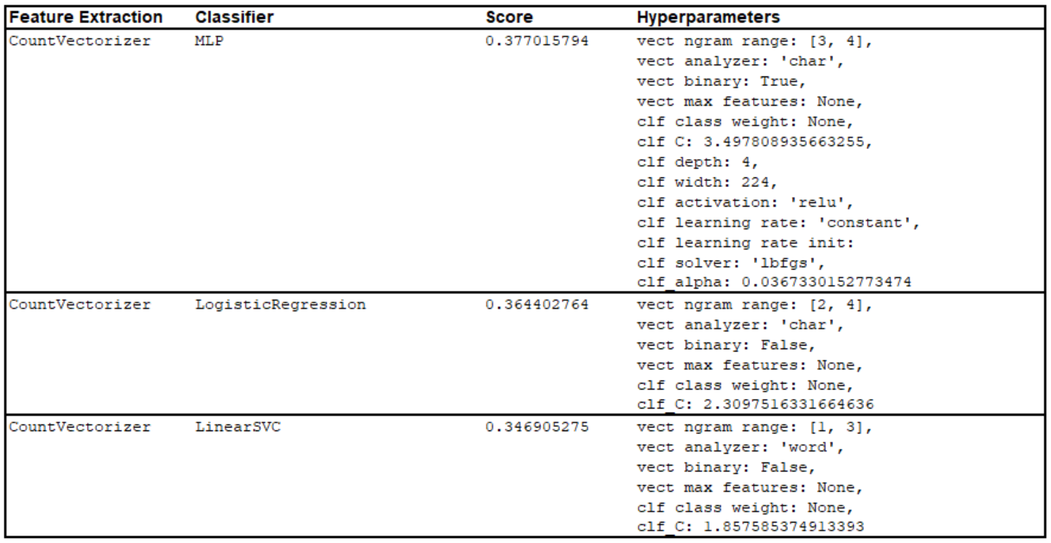
Feature Extraction	Classifier	Score	Hyperparameters
FastText	MLP	0.420723292	clf_class_weight: None, clf_C: 4.200695737208243, clf_depth: 6, clf_width: 224, clf_activation: 'tanh', clf_learning_rate: 'adaptive', 0.02272138428613412, clf_solver: 'sgd', clf_alpha: 0.09863856312993569
FastText	LogisticRegression	0.379535372	clf_class_weight: 'balanced', clf_C: 4.4845633915199965
FastText	LinearSVC	0.391863669	clf_class_weight: 'balanced', clf_C: 1.2611834415437337
FastText	MLP (cleaned)	0.380436007	clf_class_weight: None, clf_C: 4.370384759716108, clf_depth: 4, clf_width: 64, clf_activation: 'identity', clf_learning_rate: 'adaptive', clf_learning_rate_init: 0.05265084960310466, clf_solver: 'lbfgs', clf_alpha: 0.06070509628304966
FastText	LogisticRegression (cleaned)	0.386217456	clf_class_weight: 'balanced', clf_C: 4.457743388801475
FastText	LinearSVC (cleaned)	0.386832544	clf_class_weight: 'balanced', clf_C: 1.4535347597393407

A.2 RoBERTa

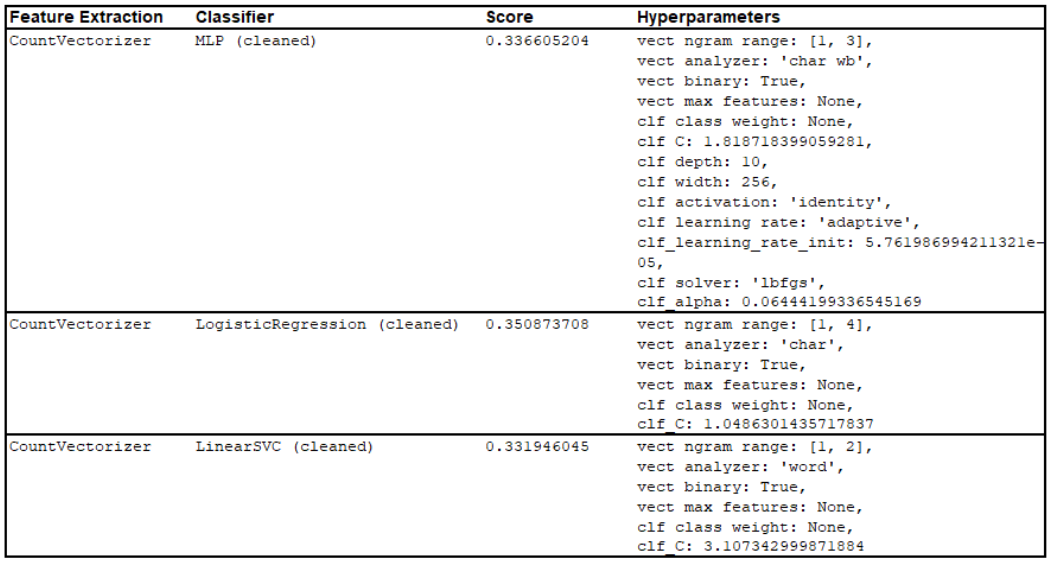


Feature Extraction	Classifier	Score	Hyperparameters
RoBERTa	MLP	0.498639413	clf_class_weight: None, clf_C: 1.9115392148575063, clf_depth: 5, clf_width: 64, clf_activation: 'relu', clf_learning_rate: 'adaptive', clf_learning_rate_init: 0.018050864561808603, clf_solver: 'lbfgs', clf_alpha: 0.05882217736452605
RoBERTa	LogisticRegression	0.498554671	clf_class_weight: None, clf_C: 1.0435661175403943
RoBERTa	LinearSVC	0.490632404	clf_class_weight: None, clf_C: 1.0435661175403943
RoBERTa	MLP (cleaned)	0.398317558	clf_class_weight: None, clf_C: 4.187625830203397, clf_depth: 7, clf_width: 160, clf_activation: 'identity', clf_learning_rate: 'constant', clf_learning_rate_init: clf_solver: 'lbfgs', clf_alpha: 0.01496127459957971
RoBERTa	LogisticRegression (cleaned)	0.394995375	clf_class_weight: 'balanced', clf_C: 1.4535347597393407
RoBERTa	LinearSVC (cleaned)	0.38878892	clf_class_weight: None, clf_C: 1.611498102351277

A.3 BoW



Feature Extraction	Classifier	Score	Hyperparameters
CountVectorizer	MLP	0.377015794	vect ngram range: [3, 4], vect analyzer: 'char', vect binary: True, vect max features: None, clf class weight: None, clf C: 3.497808935663255, clf depth: 4, clf width: 224, clf activation: 'relu', clf learning rate: 'constant', clf learning rate init: clf solver: 'lbfgs', clf_alpha: 0.0367330152773474
CountVectorizer	LogisticRegression	0.364402764	vect ngram range: [2, 4], vect analyzer: 'char', vect binary: False, vect max features: None, clf class weight: None, clf_C: 2.3097516331664636
CountVectorizer	LinearSVC	0.346905275	vect ngram range: [1, 3], vect analyzer: 'word', vect binary: False, vect max features: None, clf class weight: None, clf_C: 1.857585374913393



Feature Extraction	Classifier	Score	Hyperparameters
CountVectorizer	MLP (cleaned)	0.336605204	vect ngram range: [1, 3], vect analyzer: 'char wb', vect binary: True, vect max features: None, clf class weight: None, clf C: 1.818718399059281, clf depth: 10, clf width: 256, clf activation: 'identity', clf learning rate: 'adaptive', clf_learning_rate_init: 5.761986994211321e- 05, clf solver: 'lbfgs', clf_alpha: 0.06444199336545169
CountVectorizer	LogisticRegression (cleaned)	0.350873708	vect ngram range: [1, 4], vect analyzer: 'char', vect binary: True, vect max features: None, clf class weight: None, clf_C: 1.0486301435717837
CountVectorizer	LinearSVC (cleaned)	0.331946045	vect ngram range: [1, 2], vect analyzer: 'word', vect binary: True, vect max features: None, clf class weight: None, clf_C: 3.107342999871884

A.4 TF-IDF

Feature Extraction	Classifier	Score	Hyperparameters
TfidfVectorizer	MLP	0.37629917	vect_ngram_range: [1, 4], vect_analyzer: 'char', vect_binary: False, vect_max_features: None, clf_class_weight: None, clf_C: 1.0165063173456912, clf_depth: 2, clf_width: 32, clf_activation: 'tanh', clf_learning_rate: 'adaptive', clf_learning_rate_init: 0.043036932487617974, clf_solver: 'sgd', clf_alpha: 0.09925479167659028
TfidfVectorizer	LogisticRegression	0.368130107	vect_ngram_range: [1, 2], vect_analyzer: 'word', vect_binary: False, vect_max_features: None, clf_class_weight: None, clf_C: 4.986924417738346
TfidfVectorizer	LinearSVC	0.376621306	vect_ngram_range: [1, 4], vect_analyzer: 'word', vect_binary: True, vect_max_features: None, clf_class_weight: None, clf_C: 4.3777142710994585

Feature Extraction	Classifier	Score	Hyperparameters
TfidfVectorizer	MLP (cleaned)	0.361379596	vect_ngram_range: [1, 4], vect_analyzer: 'char', vect_binary: False, vect_max_features: None, clf_class_weight: None, clf_C: 4.187462030703548, clf_depth: 1, clf_width: 160, clf_activation: 'tanh', clf_learning_rate: 'adaptive', clf_learning_rate_init: 0.01158624359852585, clf_solver: 'sgd', clf_alpha: 0.09327594501912145
TfidfVectorizer	LogisticRegression (cleaned)	0.362160933	vect_ngram_range: [3, 4], vect_analyzer: 'char_wb', vect_binary: True, vect_max_features: None, clf_class_weight: None, clf_C: 2.871179375994383
TfidfVectorizer	LinearSVC (cleaned)	0.352768731	vect_ngram_range: [3, 4], vect_analyzer: 'char_wb', vect_binary: True, vect_max_features: None, clf_class_weight: None, clf_C: 1.0127312230144438