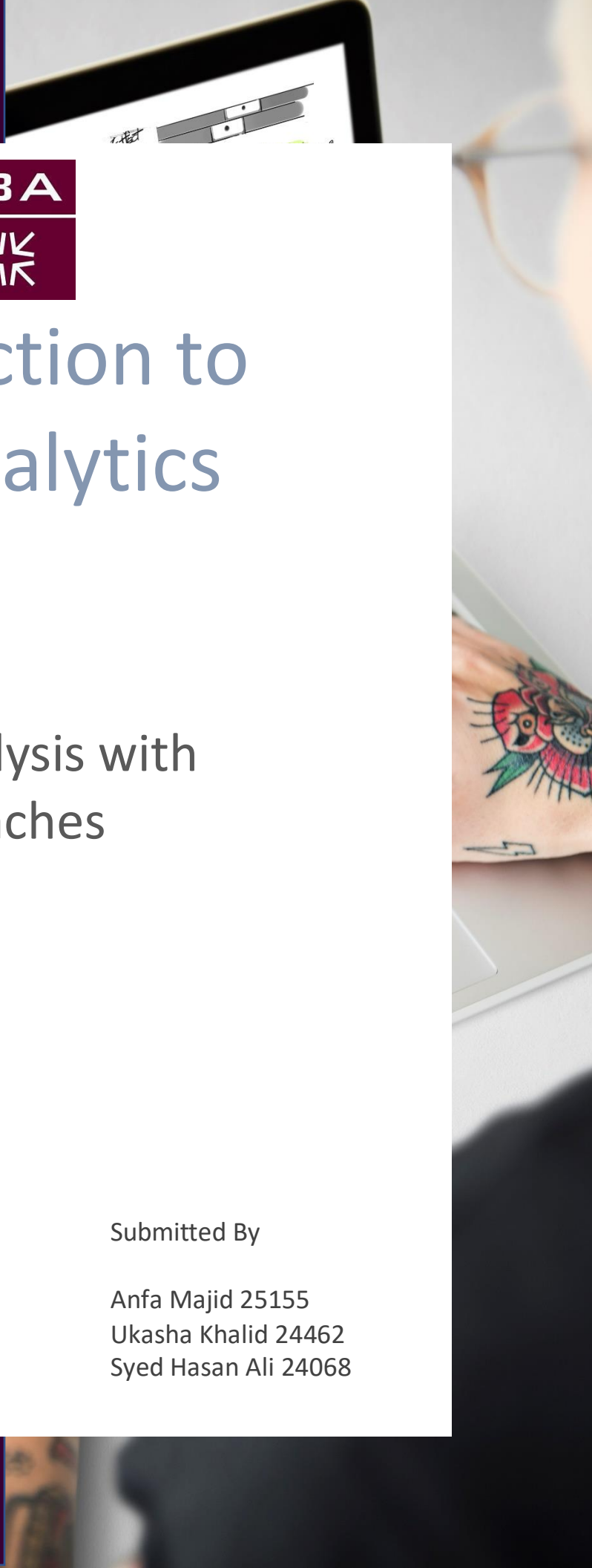

Introduction to Text Analytics

Sentiment Analysis with Various Approaches

Submitted By

Anfa Majid 25155
Ukasha Khalid 24462
Syed Hasan Ali 24068

TABLE OF

WORK BREAKDOWN

1. Lexicon-based Approach – Anfa Majid/Ukasha Khalid
2. Classical Machine Learning Approaches – Syed Hasan Ali
3. Customized Word Embedding – Anfa Majid
4. Fine-tuning Pre-trained Language Models (PLMs) – Ukasha Khalid
5. Hybrid Approach – Ukasha Khalid/Anfa Majid

DistilBERT BenchMark:

Accuracy: 0.83315

Time taken: 2098.8907935619354 seconds

The DistilBert model without finetuning gave an accuracy of 83.32%. Throughout the rest of the report, our scores will be compared against it. We tried our best to ensure that we were able to surpass this score where possible.

Lexicon-based Approach

The evaluation of the three sentiment analysis techniques reveals significant differences in both the accuracy and time taken for each method.

Model Name	Accuracy	Time Taken(s)
SentiWordNet	0.6675	326.207
VADER	0.7016	55.943
AFINN	0.72	181.227

Analysis

- There wasn't much to experiment with these three models except running our data on them.
- AFINN achieved the highest accuracy at 72.00%, making it the most effective in terms of sentiment classification for the given dataset. VADER follows closely with an accuracy of 70.16%, and SentiWordNet has the lowest accuracy at 66.75%
- Time Efficiency: VADER stands out as the most time-efficient method, with a total computation time of approximately 55.94 seconds. This is significantly faster compared to AFINN (181.23 seconds) and SentiWordNet (326.21 seconds), making VADER an attractive option for applications requiring rapid sentiment analysis.
- For applications prioritizing accuracy over speed, AFINN is recommended.
- VADER is best suited for scenarios requiring a good balance between accuracy and computational efficiency, particularly in real-time analysis contexts.
- SentiWordNet might be considered for in-depth analyses where the specific nuances captured by its approach justify the longer processing time, despite its lower overall accuracy.

The evaluation demonstrates a trade-off between accuracy and computational efficiency across the three lexicon-based sentiment analysis methods. AFINN leads in accuracy, making it the preferred choice for applications where the precision of sentiment classification is paramount. However, it requires a moderate amount of computation time. VADER, on the other hand, offers a balanced profile with reasonably high accuracy and the best time efficiency, suggesting its suitability for real-time or large-scale applications. SentiWordNet, while being the most computationally intensive, offers the lowest accuracy, which might limit its applicability in scenarios where both speed and precision are critical.

Classical Machine Learning Approaches

Random Forest:

Attempt No.	Accuracy	Stemming/Lemmatization	CV/TFIDF	Ngram range	Time Taken	max_df	min_df	Max_features	Model Tuning
Attempt 1	0.8592	Stemming	CV	(1,2)	84.3 sec	0.8	2	60000	N
Attempt 2	0.8559	Stemming	CV	(1,3)	83.6 sec	0.92	3	70000	N
Attempt 3	0.8589	Lemmatization	TFIDF	(1,2)	82.6 sec	0.8	2	60000	N
Attempt 4	0.8564	Lemmatization	TFIDF	(1,2)	76.4 sec	0.8	2	60000	N
Attempt 5	0.854	Stemming	CV	-	89 sec	-	-	60000	N

- Stemming and lemmatization performed similarly across the attempts, with stemming having a slight edge when combined with bigram analysis.
- The use of bigrams generally resulted in better performance compared to unigrams or trigrams. Trigrams might have captured some noise resulting in a lower accuracy
- There isn't a clear advantage between using CountVectorizer or TF-IDF Vectorization in this context, as both approaches yielded comparable results.
- The number of features (max_features) doesn't show a clear correlation with accuracy in this dataset; however, having too few negatively impacted performance.

Multinomial Naive Bayes:

Attempt No.	Accuracy	Stemming/Lemmatization	CV/TFIDF	Ngram range	Time Taken	max_df	min_df	Max_features	Model Tuning
Attempt 1	0.8759	Stemming	CV	(1,2)	0.013 sec	0.8	2	60000	N
Attempt 2	0.8864	Stemming	TFIDF	(1,3)	0.012 sec	0.9	3	70000	N
Attempt 3	0.8763	Lemmatization	CV	(1,2)	0.015 sec	0.8	2	60000	N
Attempt 4	0.8838	Lemmatization	TFIDF	(1,2)	0.012 sec	0.8	2	60000	Y
Attempt 5	0.8538	Stemming	CV	-	0.0097 sec	-	-	20000	N

- Attempt 4 is the most accurate model configuration, suggesting that TF-IDF with lemmatization and a moderate number of max features is a robust approach for this dataset.
- Attempt 2 shows that using a TF-IDF vectorizer with trigrams can also be quite effective, although it's slightly less accurate than Attempt 4.
- Stemming seems to be slightly less effective than lemmatization when combined with TF-IDF, as seen when comparing Attempt 1 and Attempt 4.
- The number of features (max_features) and the n-gram range play crucial roles in the model's performance, as reducing these hyperparameters too much can lead to a significant drop in accuracy (Attempt 5).
- The time taken for prediction is extremely low across all attempts, which is a typical advantage of the Naive Bayes classifier.

Attempt	Accuracy	Stemming/Lemmatization	CV/TFIDF	Ngram range	Time Taken	max_df	min_df	Max_features	Model Tuning
Attempt 1	0.8873	Lemmatization	CV	(1,3)	0.09 sec	0.8	3	60000	Y
Attempt 2	0.8583	Lemmatization	CV	(1,3)	0.09 sec	-	-	50000	N
Attempt 3	0.8596	Lemmatization	CV	(1,2)	0.11 sec	0.8	3	80000	N
Attempt 4	0.8597	Stemming	TFIDF	(1,2)	105.9 sec	0.8	3	80000	N
Attempt 5	0.8712	Stemming	TFIDF	(1,3)	302 sec	-	-	50000	Y

XGBOOST

- Attempt 1: Achieves the highest accuracy (87.26%) with lemmatization, CountVectorizer (CV) configured to use unigrams to trigrams, and 60,000 features. This was the highest score we got with our ML models and was perhaps due to hyperparameter tuning of not only the bag of word model but also the xgboost model.
- Attempt 2: Shows the lowest accuracy (85.83%) with lemmatization, CV using unigrams to trigrams, and a reduction in the number of features to 50,000.
- Attempt 3: Has a similar accuracy to Attempt 2 (85.96%) with lemmatization, CV using unigrams and bigrams, and an increase in max features to 80,000.
- Attempt 4: Utilizes stemming with TF-IDF vectorization and unigrams to bigrams. It achieves an accuracy close to Attempts 2 and 3 (85.97%) but takes a significantly longer time for prediction (105.9 seconds), suggesting that while TF-IDF is typically more compute-intensive than CV, it doesn't necessarily lead to better accuracy in this case.

- Attempt 5: Shows a substantial increase in prediction time (302 seconds) but yields a high accuracy (87.12%), second only to Attempt 1. This attempt uses stemming, TF-IDF vectorization with unigrams to trigrams, and reduces the max features to 50,000.
- In attempt 1 and 5, xgboost model was trained through hyperparameter tuning resulting in good score. It also showed that xgboost model did not respond as highly to changing text features as compared to tuning its hyper parameters.

Parameters used in attempt 1 (Winning Model):

n_estimators=1000, learning_rate=0.1, max_depth=7, min_child_weight=1, gamma=0.1, subsample=0.8, colsample_bytree=0.8, objective='binary:logistic', use_label_encoder=False, eval_metric='logloss

Customized Word Embedding

WORD2VEC:

The evaluation focuses on the performance of machine learning models utilizing Word2Vec for feature extraction from text data.

Time taken: 32 minutes.
Parameters: Vector size=300, Window=8, Min count=5, Workers=4, SG=1, Epochs=20

Model Name	Accuracy	Time Taken Training (s)	Time Taken Prediction (s)
Random Forest	0.83655	58.53	0.41
XG Boost	0.8665	16.95	0.07
Knn	0.77365	0.02	22.27
Gausian NB	0.73705	0.0553	0.0567

Analysis

- Accuracy: XG Boost leads in accuracy at 86.65%, making it the most effective model for the given dataset when combined with Word2Vec embeddings. Random Forest follows with an accuracy of 83.655%, and Knn and Gaussian NB show lower accuracies of 77.365% and 73.705% respectively.
- Training Efficiency: Knn demonstrates the fastest training time at virtually instantaneous (0.02 seconds), followed by Gaussian NB (0.0553 seconds). XG Boost and Random Forest require more time, with XG Boost being more efficient (16.95 seconds) than Random Forest (58.53 seconds).
- Prediction Efficiency: XG Boost also excels in prediction efficiency, requiring only 0.07 seconds, making it highly suitable for applications needing rapid responses. Gaussian NB also shows very quick prediction times (0.0567 seconds). Conversely, Knn is considerably slower in prediction, taking 22.27 seconds.
- XG Boost is recommended for projects prioritizing accuracy and efficiency, particularly when working with Word2Vec-embedded text data.
- Random Forest could be considered for its robustness and accuracy, especially in scenarios where training time is less of a concern.
- Knn might be suitable for smaller datasets or applications where the model can be frequently retrained and instant prediction is not critical.
- Gaussian NB offers a good starting point for text classification tasks, providing a balance between simplicity, training efficiency, and prediction speed.

The application of Word2Vec embeddings with various machine learning models has demonstrated significant differences in performance, both in terms of accuracy and computational efficiency. XG Boost emerges as the overall best performer, offering the highest accuracy combined with quick training and prediction times. This suggests its suitability for both high-quality text classification tasks and scenarios requiring prompt decision-making.

Random Forest, while offering competitive accuracy, is less efficient in training, which might be a consideration for larger datasets or in resource-constrained environments. Knn, despite its almost instantaneous training, suffers from long prediction times, limiting its applicability in real-time or large-scale prediction tasks. Gaussian NB presents a balanced option with moderate accuracy and very efficient training and prediction times, suitable for baseline models or initial explorations.

WORD2VEC WITH TF-IDF:

The evaluation explores the impact of integrating Word2Vec embeddings with TF-IDF (Term Frequency-Inverse Document Frequency) on the accuracy and computational efficiency of various machine learning models for sentiment analysis or text classification tasks.

Time Taken: 17 Minutes
Parameters: vector_size=300, window=8, min_count=5, workers=4, sg=1, epochs=20

Model Name	Accuracy	Time Taken Training (s)	Time Taken Prediction (s)
Random Forest	0.8278	59.02	0.43
XG Boost	0.8654	17.42	0.07
Knn	0.7649	0.01	14.62
Gaussian NB	0.7693	0.0553	0.0681

Analysis

- Accuracy: XG Boost demonstrates the highest accuracy at 86.54%, indicating its effectiveness in leveraging the combined Word2Vec and TF-IDF features for the dataset. Random Forest follows with an accuracy of 82.78%. Gaussian NB and Knn show relatively lower accuracies of 76.93% and 76.49%, respectively.
- Training Efficiency: Similar to the previous evaluation without TF-IDF, Knn shows an almost instantaneous training time (0.01 seconds), with Gaussian NB also being highly efficient (0.0553 seconds). XG Boost and Random Forest require longer training times, though XG Boost remains relatively efficient at 17.42 seconds compared to Random Forest's 59.02 seconds.
- Prediction Efficiency: XG Boost stands out for its rapid prediction time (0.07 seconds), making it highly suitable for real-time applications. Gaussian NB also demonstrates quick prediction capabilities (0.0681 seconds). However, Knn, despite its quick training, exhibits a slower prediction time (14.62 seconds), which might impact its suitability for immediate feedback systems.
- XG Boost is recommended for achieving high accuracy with efficient training and prediction, especially when working with enriched feature sets like Word2Vec combined with TF-IDF.
- Random Forest may be suitable for applications where model robustness and accuracy are prioritized over training speed.
- Knn could be considered for small datasets or in scenarios where the model training frequency is high, but its prediction time may limit its use in real-time applications.

- Gaussian NB serves as a solid baseline model, especially when quick deployment and efficiency are crucial.

The integration of Word2Vec with TF-IDF enhances the textual feature representation, leading to improved model performance in certain cases. XG Boost, with its high accuracy and quick prediction times, emerges as the most efficient and effective model for handling complex feature sets derived from text data. This suggests its robustness and adaptability to various text analysis tasks.

Random Forest, while offering good accuracy, lags in training efficiency, indicating a potential drawback for large-scale applications. Knn, despite its near-instantaneous training, is hindered by slower prediction times, affecting its applicability in scenarios requiring quick responses. Gaussian NB presents an interesting balance, offering reasonable accuracy with very efficient training and prediction times, suitable for baseline models or when computational resources are limited.

DOC2VEC:

The evaluation focuses on the performance of machine learning models trained on Doc2Vec embeddings, which capture the semantic meaning of text data.

Time Taken: 13 Minutes.
Parameters: vector_size=200, window=5, min_count=2, workers=4, epochs=20

Model Name	Accuracy	Time Taken Training (s)	Time Taken Prediction (s)
Random Forest	0.78975	70.51	0.46
XG Boost	0.81555	12.14	0.06
Knn	0.7957	0.01	11.69
Gaussian NB	0.7381	0.03701	0.05524

Analysis

- Accuracy: XG Boost demonstrates the highest accuracy at 81.555%, indicating its effectiveness in leveraging Doc2Vec embeddings for sentiment analysis or text classification tasks. Random Forest follows with an accuracy of 78.975%. Knn and Gaussian NB show relatively lower accuracies of 79.57% and 73.81%, respectively.
- Training Efficiency: Knn shows an almost instantaneous training time (0.01 seconds), with Gaussian NB also being highly efficient (0.03701 seconds). XG Boost requires slightly longer training time compared to the other models, but it remains relatively efficient at 12.14 seconds. Random Forest requires the longest training time at 70.51 seconds.
- Prediction Efficiency: XG Boost stands out for its rapid prediction time (0.06 seconds), making it highly suitable for real-time applications. Gaussian NB also demonstrates quick prediction capabilities (0.05524 seconds). However, Knn, despite its quick training, exhibits a slower prediction time (11.69 seconds), which might impact its suitability for immediate feedback systems.

- XG Boost is recommended for achieving high accuracy with efficient training and prediction, especially when working with Doc2Vec embeddings for text processing.
- Random Forest may be suitable for applications where model robustness and accuracy are prioritized over training speed, given its competitive accuracy.
- Knn could be considered for small datasets or in scenarios where the model training frequency is high, but its prediction time may limit its use in real-time applications.
- Gaussian NB serves as a solid baseline model, especially when quick deployment and efficiency are crucial.

Doc2Vec embeddings provide a powerful representation of text data, capturing semantic meanings and enabling machine learning models to achieve competitive performance in sentiment analysis or text classification tasks. XG Boost emerges as the most effective model, offering the highest accuracy combined with efficient training and prediction times.

Random Forest, while offering good accuracy, requires longer training times, which might be a consideration for large-scale datasets or resource-constrained environments. Knn, despite its near-instantaneous training, exhibits slower prediction times, which might limit its applicability in real-time scenarios. Gaussian NB presents a balanced option, offering reasonable accuracy with very efficient training and prediction times.

GLOVE:

The evaluation focuses on the performance of machine learning models trained on GloVe embeddings, which capture semantic information from pre-trained word vectors.

Time Taken: 4 Minutes.

Model Name	Accuracy	Time Taken Training (s)	Time Taken Prediction (s)
Random Forest	0.76995	40.83	0.38
XG Boost	0.78765	12.33	0.13
Knn	0.74935	0.01	10.78
Gaussian NB	0.72095	0.0605	0.02296

Analysis

- Accuracy: XG Boost demonstrates the highest accuracy at 78.765%, indicating its effectiveness in leveraging GloVe embeddings for sentiment analysis or text classification tasks. Random Forest follows with an accuracy of 76.995%. Knn and Gaussian NB show relatively lower accuracies of 74.935% and 72.095%, respectively.
- Training Efficiency: Knn shows an almost instantaneous training time (0.01 seconds), with Gaussian NB also being highly efficient (0.0605 seconds). XG Boost requires slightly longer training time compared to the other models, but it remains relatively efficient at 12.33 seconds. Random Forest requires the longest training time at 40.83 seconds.
- Prediction Efficiency: XG Boost stands out for its rapid prediction time (0.13 seconds), making it highly suitable for real-time applications. Gaussian NB also demonstrates

quick prediction capabilities (0.02296 seconds). However, Knn, despite its quick training, exhibits a slower prediction time (10.78 seconds), which might impact its suitability for immediate feedback systems.

- XG Boost is recommended for achieving high accuracy with efficient training and prediction, especially when working with GloVe embeddings for text processing.
- Random Forest may be suitable for applications where model robustness and accuracy are prioritized over training speed, given its competitive accuracy.
- Knn could be considered for small datasets or in scenarios where the model training frequency is high, but its prediction time may limit its use in real-time applications.
- Gaussian NB serves as a solid baseline model, especially when quick deployment and efficiency are crucial.

GloVe embeddings provide a powerful representation of text data, capturing semantic meanings and enabling machine learning models to achieve competitive performance in sentiment analysis or text classification tasks. XG Boost emerges as the most effective model, offering the highest accuracy combined with efficient training and prediction times.

Random Forest, while offering good accuracy, requires longer training times, which might be a consideration for large-scale datasets or resource-constrained environments. Knn, despite its near-instantaneous training, exhibits slower prediction times, which might limit its applicability in real-time scenarios. Gaussian NB presents a balanced option, offering reasonable accuracy with very efficient training and prediction times.

OVERALL COMPARISION:

Each embedding technique, including Doc2Vec, Word2Vec, Word2Vec with TF-IDF, and GloVe, offers distinct advantages and considerations for text processing tasks. Doc2Vec excels in capturing document-level semantics, while Word2Vec and GloVe focus on word-level semantics with different approaches. Word2Vec with TF-IDF combines the strengths of Word2Vec and TF-IDF, enhancing representations with document-specific information. The choice among these techniques depends on factors such as the level of contextual understanding required, training efficiency, representation flexibility, and handling of out-of-vocabulary words. Ultimately, selecting the optimal technique involves considering the specific task requirements, available resources, and desired trade-offs between different aspects of text processing.

Fine-tuning Pre-trained Language Models (PLMs)

Finetuning Attempt 1:

Model Name	Accuracy	Epochs	Batch Size	Learning Rate	Time Taken
Distilled-BERT	0.9173	4	16	2e-5	2hr10mins
GPT-2	0.92685	4	4	2e-5	2hr6mins
Roberta	0.9406	4	16	2e-5	2hr12mins

- DistilBERT achieved an accuracy of 0.9173 with a larger batch size of 16 and the standard learning rate of 2e-5, taking 2 hours and 10 minutes. The higher batch size may contribute to faster convergence but can also lead to less generalization.
- GPT-2 had accuracy of 0.92685 with a small batch size of 4 and the same learning rate, taking 2 hours and 6 minutes. The small batch size is due to the larger memory requirements of GPT-2, which generally has more parameters than DistilBERT.
- RoBERTa showed very good performance with the highest accuracy of 0.9406, matching DistilBERT's batch size and learning rate, but taking slightly longer (2 hours and 12 minutes). This suggests that RoBERTa may have benefited from its larger architecture and more parameters compared to DistilBERT, at the cost of additional training time.

Finetuning Attempt 2:

Model Name	Accuracy	Epochs	Batch Size	Learning Rate	Time Taken
Distilled-BERT	0.90375	5	12	3e-5	1hr46mins
GPT-2	0.9288	4	6	5e-5	2hr2mins
Roberta	0.9335	5	12	3e-5	2hr21mins

- DistilBERT showed a slight decrease in accuracy to 0.90375, despite an increase in epochs to 5 and a higher learning rate of 3e-5, and it took less time (1 hour and 46 minutes), which could indicate overfitting or that the higher learning rate caused instability in training.
- GPT-2 had a slight improvement in accuracy to 0.9288 with an increased batch size of 6 and a higher learning rate of 5e-5, taking 2 hours and 2 minutes. The larger learning rate seems to have been beneficial without causing overfitting or divergence.
- RoBERTa dropped to an accuracy of 0.9335 with more epochs and a higher learning rate, taking slightly longer than before (2 hours and 21 minutes). This suggests that RoBERTa can benefit from more training time and a higher learning rate, perhaps due to its robustness and capacity to generalize from larger datasets.

Finetuning Attempt 3:

Model Name	Accuracy	Epochs	Batch Size	Learning Rate	Time Taken
Distilled-BERT	0.91815	10	16	2e-5	2hr27mins
GPT-2	0.9266	5	6	3e-5	2hr26mins
Roberta	0.938	6	8	5e-6	2hr55mins

- DistilBERT's accuracy increased slightly to 0.91815 with a significant jump in epochs to 10, but at the cost of increased training time (2 hours and 27 minutes). The increased epochs seem to have given the model more time to learn from the data without overfitting.
- GPT-2 maintained a similar accuracy level of 0.9266 with an increased number of epochs and a higher learning rate of 3e-5, taking 2 hours and 26 minutes. The balance between epochs and learning rate appears to be effective.
- RoBERTa achieved the best accuracy of all three attempts at 0.938 with a smaller batch size of 8 and a high learning rate of 5e-6, although it took the longest time to train (2 hours and 55 minutes). The smaller batch size, combined with more epochs and a high learning rate, might have allowed RoBERTa to effectively learn finer nuances in the data.

OVERALL COMPARISON:

RoBERTa consistently shows high performance, likely benefiting from its architecture and capacity, which allows it to learn complex patterns in the data.
GPT-2 demonstrates that even with a smaller batch size, a well-chosen learning rate can lead to high accuracy, although this comes with significant memory and time requirements.
DistilBERT presents a more time-efficient option with competitive accuracy, especially in the first and third attempts, suggesting it can be a good choice when resources are limited.

Hybrid Approach

1st approach:

Combination: Lexicon (Vader), Word2Vec, XG Boost.
Time taken for word2vec: 16 Minutes.

Model Name	Accuracy	Time Taken (s)
XG Boost	0.87575	0.1408

We decided on this approach to get an improvement on Lexicon based analysis and Word embeddings as they were on the lower side. While there was a significant jump in the case of Vader, Word2Vec and XgBoost gave us a similar accuracy as shown above which gives us the idea that Lexicon on its own is not ideal for sentiment analysis. Pairing it with a much powerful ML model that uses word embeddings will give us an overall better score. It comes at the cost of increased training time however as our model which was first being completed within a minute now took 16 minutes.

2nd Approach:

Combination: Roberta, XG Boost, Logistic Regression

Model Name	Accuracy	Time Taken Training (s)
Roberta	0.9406	22 mins
XGBoost	0.8873	93.88s
Logistic Regression	0.94115	30 mins

Params for Roberta: epochs = 2, batch size = 16, lr = 2e-5

Params for xgboost:n_estimators=1000,learning_rate=0.1,max_depth=7,min_child_weight=1,gamma=0.1,subsample=0.8,colsample_bytree=0.8,objective='binary:logistic',use_label_encoder=False,eval_metric='logloss'

The approach taken in the code combines predictions from the best version of our fine-tuned RoBERTa model and our best peforming XGBoost classifier(with hyperparameters) to create a hybrid model. It involves:

- Text Preprocessing and Vectorization: Preprocesses text data and converts it to numerical format using CountVectorizer.
- Sentiment Analysis with RoBERTa: Uses a RoBERTa model fine-tuned for sentiment analysis to predict sentiment probabilities.
- Machine Learning with XGBoost: Trains an XGBoost classifier on the vectorized text data to predict sentiment probabilities.
- Combining Predictions: Concatenates the sentiment probabilities from both RoBERTa and XGBoost models.
- Meta-Classification: Trains a logistic regression meta-classifier on the combined probabilities to make the final sentiment predictions.

Due to time constraints I had to limit the number of epochs to 2 as compared to the orginal finetuned version. However despite that our hybrid model gave an even better accuracy. The parameters for the winning ML model above were also used here. Compared to the first approach, this one was comparatively better due to using a finetuned version of a PLM instead of word embeddings which are not that accurate. The time difference was also not that significant which suggests that this model can be preferred over Lexicon.

3rd Approach:

Lexicon(Vader), BagofWords(TFIDF), MLModel(Random Forest)

	Accuracy	Time Taken Training (s)
Model	0.8527	51 mins

The hybrid approach in this model involves combining lexicon-based sentiment analysis (using VADER) with machine learning techniques (using a RandomForestClassifier) and TF-IDF vectorization of text data. It preprocesses text data, extracts sentiment scores using VADER, and combines these scores with TF-IDF features. This enriched feature set is then used to train a RandomForestClassifier for sentiment classification. This method leverages both sentiment detection of a lexicon-based approach and the predictive power of machine learning models.

Once again Vader benefitted from hybrid approach further strengthening our claim of using Lexicon based models with other approaches. It took exceptionally long to run compared to the other two models due to a larger feature set.

Overall Comparison:

- First Approach: This combines a lexicon-based method (Vader), Word2Vec embeddings, and the XGBoost algorithm. It has moderate accuracy and a relatively fast word2vec processing time. This approach seems to balance traditional machine learning and modern NLP techniques, leveraging sentiment scoring, word embeddings for context capture, and a powerful gradient boosting machine learning model.

- Second Approach: This approach integrates the fine-tuned RoBERTa model, XGBoost, and a logistic regression as a meta-classifier. It has the highest accuracy among the three, indicating that the combination of deep learning for feature extraction (RoBERTa), gradient boosting (XGBoost), and linear modeling (logistic regression) for final predictions is highly effective. However, it also takes the longest time to train, likely due to the complexity of the models involved.

- Third Approach: This approach uses a lexicon-based method (Vader), Bag of Words with TF-IDF, and a Random Forest classifier. It has the lowest accuracy and a relatively longer training time. This method seems to rely on simpler NLP techniques and classical machine learning, which might be less capable of capturing complex patterns compared to deep learning methods.

In summary, while the second approach yields the best accuracy, it is also the most time-consuming. The first approach is a good trade-off between accuracy and efficiency, while the third approach, despite its simplicity, may not capture the nuances in the data as effectively as the others.

Comparison of different Approaches

1. Lexicon-Based Approach: Utilizes sentiment lexicons like AFINN, VADER, and SentiWordNet to classify sentiment. AFINN led in accuracy, but VADER was the most time efficient.

2. Classical Machine Learning: Random Forest and Multinomial Naive Bayes models are assessed with different preprocessing like stemming and lemmatization, and vectorization techniques. The models showed decent performance but varied based on the n-gram range and feature selection.

3. Word Embeddings: Word2Vec, Doc2Vec, and GloVe techniques are integrated with machine learning models. XGBoost combined with Word2Vec or GloVe demonstrated the highest accuracy and efficiency.

4. Fine-tuning PLMs: Pre-trained language models like RoBERTa, DistilBERT, and GPT-2 are fine-tuned with different hyperparameters. RoBERTa generally provided the highest accuracy but at the cost of longer training times.

5. Hybrid Approaches: Combine different techniques. One approach merged VADER, Word2Vec, and XGBoost, showing good balance and performance. Another integrated RoBERTa, XGBoost, and Logistic Regression, which yielded the highest accuracy but took the longest training time. A third combined VADER, TF-IDF, and Random Forest, which was less effective in accuracy.

In conclusion, while fine-tuned PLMs showed the highest accuracy, hybrid approaches leveraging both traditional and modern techniques offered a balance between performance and computational efficiency. The choice of method may depend on specific requirements like accuracy, training speed, and computational resources.

Ibtihaj

Uddin

220123204