Project Report

Shivam Siddhapura (030824766)
Yash Shah (030867146)

Topic: Sentiment Analysis on Mix-Domain Dataset

1. How to run your code?

- https://colab.research.google.com/drive/16XpkZot9mB_SpBbjUeFLiwtKJbX19GJD?usp=sharing
- In the code folder check the requirements to download the requirements and then run the code.
- All the graphs and trained models can be found in the colab link.

2. About project

- Initially, we considered leveraging sentiment analysis through hybrid models by integrating BERT(Bidirectional Encoder Representations from Transformers) with other neural network architectures like CNN, LSTM, Attention Mechanisms, and Capsule Networks. Our goal was to train and test these hybrid models on mixed-domain datasets, aiming to enhance the efficacy and accuracy of sentiment analysis across diverse data sources. However, as we delved deeper into the literature and existing research, we discovered that BERT, as a standalone model, already demonstrates exceptional performance and reaches peak accuracy levels in sentiment analysis tasks. Many of the papers (Medhat et al.) (Nhan Cach Dang, María N. Moreno-García, Fernando De la Prieta) we reviewed also successfully applied BERT to mixed-domain datasets, including Sentiment140, IMDB reviews, Twitter tweets, and multi-perspective question-answering data.
    - **Objectives:**
      This thing led us to reconsider our approach. We decided to pivot from using the BERT model which is already advanced and instead we focus on developing and refining basic machine learning models to improve their effectiveness on mixed-domain datasets. This shift allowed us to explore various techniques to improvise sentiment analysis using basic models in different contexts.
- *Terminologies:*
    1. **Sentiment Analysis:** the process of computationally identifying and categorizing opinions expressed in a piece of text, especially in order to determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral.
    2. **CNN:** A type of neural network used primarily in image processing and computer vision. CNNs are designed to automatically and adaptively learn spatial hierarchies of features, from low-level to high-level patterns, through convolutional layers that filter inputs for useful information.
    3. **LSTM:** A type of recurrent neural network (RNN) designed to remember long-term dependencies in sequential data. LSTMs are particularly useful in time series analysis, natural language processing, and other domains where understanding the sequence is crucial.
    4. **Word2Vec:** A group of related models that are used to produce word embeddings, which are vector representations of words. These models are trained to capture the context of a word in a document, resulting in vectors where words that have similar meanings are located close to each other in the vector space.
    5. **Word Cloud:** A visual representation of text data where the size of each word indicates its frequency or importance in the document or we can say most prominent words in the given document.

3. Dataset

- For our project, we used multiple datasets that encompassed a variety of sources, including movie reviews, Yelp reviews, and twitter-tweets. Each dataset brought unique perspectives and linguistic styles, enriching our analysis and enhancing the robustness of our models.
    1. **Movie Reviews:** This dataset consisted of critiques from a popular movie review aggregator. The reviews ranged from short, concise statements to longer, detailed evaluations, providing a diverse linguistic dataset.
    2. **Yelp Reviews:** These reviews came from the Yelp platform and included customer feedback on various services and restaurants. The language used here was more informal and included slang and local dialects, offering a different challenge for sentiment analysis.
    3. **Tweets:** The dataset of tweets was a collection of short messages from Twitter. Due to the character limit on tweets, the text was often abbreviated and used more hashtags and mentions, differing significantly from the more verbose reviews.
- To manage these datasets, we employed Pandas to randomize and scale down the data effectively. Initially, we reduced each dataset to maintain computational efficiency while ensuring enough data to train and test our models accurately. We changed the resultant sentiment to 0 (positive) or 1 (negative) in each dataset. Additionally, we added a new field which maintains the source of the data and we made sure that the merged data will contain equal positive and negative sentiments data, as shown in *fig 3.1*.

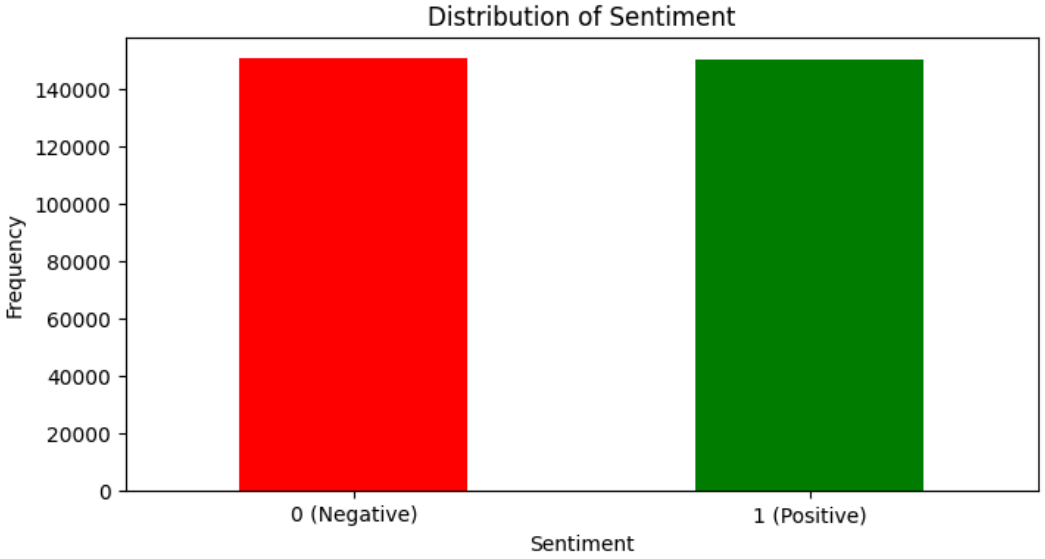

Fig. 3.1

- However, we faced a challenge with the tweets dataset, which was less in proportion compared to the movie and Yelp reviews. To address this, we adjusted the bias to zero, ensuring the number of tweets equaled the combined number of movie and Yelp reviews. This balancing act allowed for a more equitable representation(as in *fig 3.2*) across different text sources.

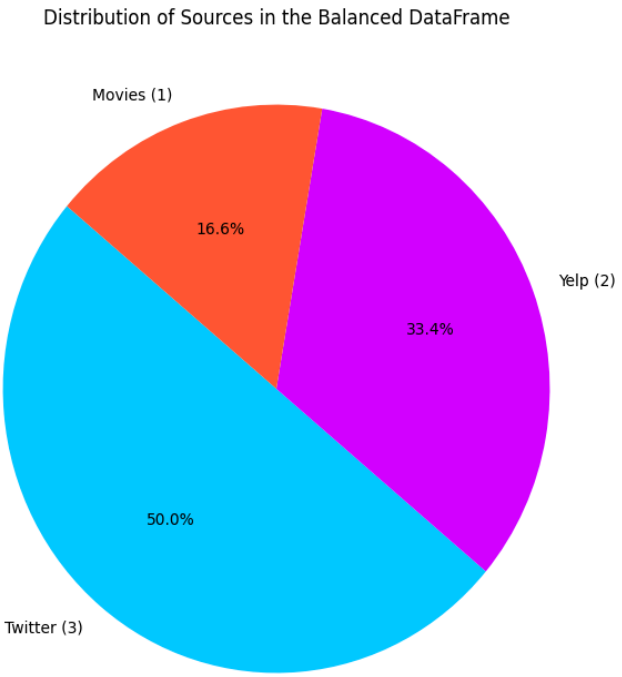

Fig. 3.2

- Lastly, for data cleaning, we used regular expressions (regex) to perform simple yet effective cleaning operations. This included removing unnecessary characters, standardizing the use of whitespace, and stripping any extraneous formatting that could interfere with the sentiment analysis.

4. Methods

4.1. Logistic Regression

- Start-off we took a pretty straightforward approach to figuring out the sentiments in our dataset. First, we split our data into parts: one part for training our model and the other part to check how well it's doing (i.e the validation and test sets). We used a tool called TF-IDF to turn our text into numbers that our model can understand, focusing only on the most important words and ignoring common ones that don't add much info.
- We picked Logistic Regression, a simple yet effective way to predict whether a review is positive or negative, and trained it on our training data. After that, we tested it first on the validation set to tweak things if needed and then on the test set to see how well it could handle new, unseen data.

4.2. Logistic Regression with Pipelining

- For the second model, we added a few more steps to make our sentiment analysis more robust. This time, we also included a scaling step to ensure all our features are on the same scale, which can sometimes help the model perform better.
- We wrapped these steps into a pipeline, which is a neat way to bundle up preprocessing and modelling steps. This pipeline includes the scaling and our Logistic Regression model. Also, we added maximum iteration steps. We trained this combined setup on our training data, then checked how well it did on the validation set to fine-tune things if needed, and finally tested it on unseen data from the test set.

4.3. Basic Convolutional Neural Network (CNN)

- Now we explored sentiment analysis using a convolutional neural network (CNN) with Keras and TensorFlow. We first split our text data into training, validation, and test sets and then converted the text into numerical sequences. We ensured uniform input lengths by padding these sequences.
- Our CNN model, featuring an embedding layer followed by convolutional and pooling layers, aimed to identify patterns in the text to classify sentiments as positive or negative. We employed early stopping during training to prevent overfitting and maintain model

generalizability. Finally, we evaluated the model's performance on the test set to gauge its real-world accuracy.

4.4. CNN with increased Vocabulary size

- We enhanced our approach by increasing our tokenizer's vocabulary to 5000 words, allowing the model to capture a broader range of language nuances. Text sequences were converted to numerical sequences and uniformly padded for consistent input sizes.
- Further, adjusted padding based on the longest sequence, improving efficiency. Also, maintained architecture but ensured robustness through careful training and early stopping, boosting predictive accuracy.

4.5. Dense CNN

- To enhance efficiency and improve the model, we increased the kernel size in Conv1D to 5, added a Dense layer with 50 neurons, and included a Dropout layer with a rate of 0.5 to reduce overfitting. For learning rate reduction we added a ReduceLROnPlateau callback to decrease the learning rate of the validation loss plateaus, aiding in finer convergence.
- Also, updated training parameters by increasing the number of epochs to 20 and reduced the batch size to 64 for more frequent updates, resulting in dense CNN.

4.6. CNN with Word Cloud

- In this iteration, we started by identifying high-frequency words in our dataset using a word cloud, which helped visualize the most common words in the reviews. This step informed our text preprocessing and modelling strategies.
- Our convolutional neural network (CNN) was enhanced with an embedding layer for dense vector representation of words, convolutional and pooling layers to capture textual features, and a dropout layer to prevent overfitting.
- We compiled the model with an optimizer and loss function suited for binary classification and employed both early stopping and learning rate reduction during training to fine-tune the model's performance.

4.7. CNN with Word Cloud (TF-IDF)

- Further, the tokenizer is now initialized with the most important words as determined by TF-IDF, ensuring the model learns from the most significant parts of the text. We refined tokenization by only using top N words (based on TF-IDF scores), which helps the model focus on the most informative aspects of the text.
- Model training continues with the CNN structure, now potentially more focused and effective due to the refined vocabulary.

4.8. Word2Vec

- We reached saturation of CNN model, so now in order to improvise it further we have to move to a different model. We enhanced our approach by using Word2Vec to create word embeddings that capture semantic relationships and improve context understanding.
- We then trained a Word2Vec model on our dataset and created an embedding matrix to map words to their vector representations. We evaluated the model's performance on the test set, obtaining accuracy, precision, recall, and F1 scores.

4.9. CNN + Word2Vec

- We now combined CNN with Word2Vec. We trained a Word2Vec model on the tokenized texts to generate word embeddings, capturing deeper linguistic features. These embeddings were used to initialize a non-trainable layer in our neural network, which also included a convolutional layer to capture contextual relationships within the text.
- After the Embedding layer, we introduce a Conv1D layer to capture local dependencies in the text data. In the Pooling Layer we use a GlobalMaxPooling1D to reduce the

dimensionality after the convolutional layer, capturing the most important features from each filter.

4.10. Dense CNN + Word2Vec

- We made the model denser to improve accuracy by making the network denser by increasing the number of units in the Dense layers and potentially adding more Dense layers.
- We can increase units in dense layers by Increasing the number of neurons in the existing Dense layer. Also, introduced more Dense layers to give the model more capacity to learn complex representations.

4.11. LSTM + Word2Vec

- Integrating an LSTM layer into a text classification model significantly improves its ability to handle sequential data. Additionally, LSTMs adapt well to complex language features, such as varying sentence structures and synonyms, and when paired with Word2Vec embeddings, they can leverage semantic richness along with contextual word order.
- However, the effectiveness of LSTMs depends on factors like data complexity and size, with larger, more complex datasets seeing greater benefits. Proper hyperparameter tuning and managing the increased computational resources and training time are essential for optimizing performance.

5. Results

<table>
<tr><th>Model</th><th>Accuracy</th></tr>
<tr><td>Logistic Regression</td><td>
Validation Accuracy: 0.8131813181318132
<table>
<tr><th>Class</th><th>Precision</th><th>Recall</th><th>F1-Score</th><th>Support</th></tr>
<tr><td>0</td><td>0.80</td><td>0.83</td><td>0.82</td><td>29914</td></tr>
<tr><td>1</td><td>0.82</td><td>0.80</td><td>0.81</td><td>30080</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.81</td><td>59994</td></tr>
<tr><td>macro avg</td><td>0.81</td><td>0.81</td><td>0.81</td><td>59994</td></tr>
<tr><td>weighted avg</td><td>0.81</td><td>0.81</td><td>0.81</td><td>59994</td></tr>
</table>
Test Accuracy: 0.8125479214588125
<table>
<tr><th>Class</th><th>Precision</th><th>Recall</th><th>F1-Score</th><th>Support</th></tr>
<tr><td>0</td><td>0.80</td><td>0.83</td><td>0.81</td><td>29901</td></tr>
<tr><td>1</td><td>0.82</td><td>0.80</td><td>0.81</td><td>30093</td></tr>
<tr><td>accuracy</td><td></td><td></td><td>0.81</td><td>59994</td></tr>
<tr><td>macro avg</td><td>0.81</td><td>0.81</td><td>0.81</td><td>59994</td></tr>
<tr><td>weighted avg</td><td>0.81</td><td>0.81</td><td>0.81</td><td>59994</td></tr>
</table>
</td></tr>
</table>

<table>
<tr><td>Logistic Regression with pipelining</td><td>Validation Accuracy: 0.8117145047838117<table><tr><th>Class</th><th>Precision</th><th>Recall</th><th>F1-Score</th><th>Support</th></tr><tr><td>0</td><td>0.80</td><td>0.83</td><td>0.81</td><td>29914</td></tr><tr><td>1</td><td>0.82</td><td>0.80</td><td>0.81</td><td>30080</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.81</td><td>59994</td></tr><tr><td>macro avg</td><td>0.81</td><td>0.81</td><td>0.81</td><td>59994</td></tr><tr><td>weighted avg</td><td>0.81</td><td>0.81</td><td>0.81</td><td>59994</td></tr></table>Test Accuracy: 0.8102810281028103<table><tr><th>Class</th><th>Precision</th><th>Recall</th><th>F1-Score</th><th>Support</th></tr><tr><td>0</td><td>0.80</td><td>0.82</td><td>0.81</td><td>29901</td></tr><tr><td>1</td><td>0.82</td><td>0.80</td><td>0.81</td><td>30093</td></tr><tr><td>accuracy</td><td></td><td></td><td>0.81</td><td>59994</td></tr><tr><td>macro avg</td><td>0.81</td><td>0.81</td><td>0.81</td><td>59994</td></tr><tr><td>weighted avg</td><td>0.81</td><td>0.81</td><td>0.81</td><td>59994</td></tr></table></td></tr>
<tr><td>Convolutional Neural Network (CNN)</td><td>Test Accuracy: 0.8189818859100342</td></tr>
<tr><td>CNN with increased vocab size</td><td>Test Accuracy: 0.842850923538208</td></tr>
<tr><td>Dense CNN</td><td>Test Accuracy: 0.8449178338050842</td></tr>
<tr><td>CNN with Word Cloud</td><td>Test Accuracy: 0.8464013338088989</td></tr>
<tr><td>CNN with TF-IDF Word Cloud</td><td>Test Accuracy: 0.8427509665489197</td></tr>
<tr><td>Word2Vec</td><td>Test Accuracy: 0.7928292751312256<table><tr><th>Precision</th><th>Recall</th><th>F1 Score</th><th>Time</th></tr><tr><td>0.821484</td><td>0.854352</td><td>0.837960</td><td>274.9616</td></tr></table></td></tr>
<tr><td>Word2Vec with CNN</td><td>Test Accuracy: 0.8338167071342468<table><tr><th>Precision</th><th>Recall</th><th>F1 Score</th><th>Time</th></tr><tr><td>0.821484</td><td>0.854352</td><td>0.837960</td><td>274.9616</td></tr></table></td></tr>
<tr><td>Word2Vec with Dense CNN</td><td>Test Accuracy: 0.8391839265823364<table><tr><th>Precision</th><th>Recall</th><th>F1 Score</th><th>Time</th></tr><tr><td>0.847314</td><td>0.828731</td><td>0.837920</td><td>282.9251</td></tr></table></td></tr>
</table>

Word2Vec with LSTM	Test Accuracy: 0.852051854133606 Precision: 0.847421 | Recall: 0.859868 | F1 Score: 0.853599 | Time: 2155.5156

6. Conclusion

- In our project, we initially aimed to utilize advanced hybrid models, specifically integrating BERT with other neural network architectures such as CNN, LSTM, Attention Mechanisms, and Capsule Networks, for sentiment analysis on mixed-domain datasets. Our goal was to enhance the performance and accuracy of sentiment analysis across various data sources by leveraging the strengths of these combined models. However, upon reviewing the literature and existing research, we discovered that BERT, as a standalone model, already achieves exceptional performance and peak accuracy in sentiment analysis tasks.
- Given this insight, we decided to pivot our approach. Instead of focusing on an already advanced and highly effective model like BERT, we concentrated on improving and refining basic machine learning models for sentiment analysis on mixed-domain datasets. This decision allowed us to explore and implement various techniques to enhance the performance of these simpler models in different contexts.
- Our experimental results highlighted several key findings:
    1. **Logistic Regression:** This basic model achieved a validation accuracy of 81.32% and a test accuracy of 81.25%. While these results are respectable, they indicate the limitations of logistic regression in handling the complexities of mixed-domain datasets.
    2. **Logistic Regression with Pipelining:** Incorporating pipelining did not significantly improve the performance, yielding a validation accuracy of 81.17% and a test accuracy of 81.03%. The marginal improvement suggests that pipelining alone is insufficient for substantial performance gains.
    3. **Convolutional Neural Network (CNN):** Our CNN model achieved a test accuracy of 81.89%. When we increased the vocabulary size, the test accuracy improved to 84.29%, demonstrating the model's sensitivity to input vocabulary size. Dense CNN configurations further boosted the test accuracy to 84.49%, while incorporating Word Clouds and TF-IDF Word Clouds resulted in test accuracies of 84.64% and 84.28%, respectively. These results indicate that CNNs, with appropriate tuning, can perform well on mixed-domain datasets.
    4. **Word2Vec:** The standalone Word2Vec model achieved a test accuracy of 79.28%. This relatively lower performance underscores the limitations of using Word2Vec without additional enhancements.
    5. **Word2Vec with CNN:** Combining Word2Vec with CNN improved the test accuracy to 83.38%, highlighting the benefits of integrating word embeddings with neural network architectures.
    6. **Word2Vec with Dense CNN:** This combination yielded a test accuracy of 83.92%, further emphasizing the effectiveness of dense neural network configurations in processing word embeddings.
    7. **Word2Vec with LSTM:** This configuration achieved the highest test accuracy of 85.21%. The LSTM's ability to capture long-term dependencies in the data likely contributed to its superior performance. The model also demonstrated high precision, recall, and F1 scores, indicating robust and reliable sentiment analysis capabilities.
- In conclusion, our project revealed that basic machine learning models, when properly tuned and enhanced, can achieve impressive results in sentiment analysis, often rivalling more advanced models. Our focus on improving simpler models like Logistic Regression, CNNs, and especially the Word2Vec with LSTM configuration, demonstrated that these models could handle the complexities of mixed-domain datasets effectively.

7. References

- Medhat, Walaa, et al. “Sentiment analysis algorithms and applications: A survey.” *Research Gate*, 19 04 2014, https://www.researchgate.net/publication/261875740_Sentiment_Analysis_Algorithms_and_Applications_A_Survey. Accessed 19 04 2014.
- Nhan Cach Dang, María N. Moreno-García, Fernando De la Prieta. “Sentiment Analysis Based on Deep Learning: A Comparative Study.” *Cornell University*, 2020, https://arxiv.org/abs/2006.03541. Accessed 5 June 2020.