TWEET SENTIMENT CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS

James Li
Student# 1007974248
jamesw.li@mail.utoronto.ca

Pierre Ishak
Student# 1007897457
pierre.ishak@mail.utoronto.ca

Tanmay Patel
Student# 1008279168
tanmayn.patel@mail.utoronto.ca

Nicholas Lupu-Muslea
Student# 1007896154
n.lupumuslea@mail.utoronto.ca

Link to GitHub: https://github.com/piroo8/Aps360

1 PROBLEM STATEMENT

In this day and age, where most everyday interactions are digitized, it is absolutely vital to be able to extract essential information from said interactions and use them to your advantage. According to an investigation conducted by McKinsey and Co.,(1) in 2020 following the COVID-19 crisis, it was noted that the global digitization of customer interactions drastically accelerated from 36% in December 2019 to 58% in July 2020 and is still growing to this day.

Given the context, the project goal is to create a tweet sentiment analysis classification model using a convolutional neural network (CNN) that would classify the tweet as one of three sentiments: positive, negative or neutral. This model can then be used by other companies by transfer learning, to analyse and classify text data to deduce quality insights regarding their products. Since social media presence is so abundant, this tool will be great for marketing teams looking to gain insights on product quality.

This is demonstrated by training the model using a large database with a variety of tweets and testing it using a separate database, a variety of tweets mentioning Dell. It is important to discuss why deep learning is necessary for such a task; the ability to learn hierarchical representations and model complex relationships is what it a great candidate for the task. In addition, a CNN is easily able to detect low-level features that include words or word combinations which are then used to learn higher level features such as phrases and expressions with those words. What a CNN offers that is different from an LSTM or RNN is that it does not require time-series data, appropriate for tweets which do not necessarily follow ordinary grammatical and sequential structure. By leveraging these deep learning techniques, the aim is to build a robust model that can accurately classify sentiments in texts, enabling faster and objective-based analysis. ***Click the following for the illustration: 4***

2 BACKGROUND AND RELATED WORK

2.1 SENTIMENT ANALYSIS USING DEEP LEARNING: FINANCIAL MARKET PREDICTION (1)

The report from McKinsey highlights that COVID-19 accelerated technological adoption, driving companies across industries towards a permanent digital transformation. It emphasizes that organizations which effectively leverage technology have gained a competitive edge, reshaping business operations and customer interactions. Sentiment analysis using deep learning emerges as a pivotal tool, enabling firms to understand and respond to evolving customer preferences, thus shaping their strategies in this tech-driven landscape.

2.2 SENTIMENT ANALYSIS USING RECURSIVE NEURAL NETWORKS (RECNN) (3)

The report explores the significant advantages of utilizing Deep Learning for Sentiment Analysis. It delves into the effectiveness of deep neural networks in capturing intricate emotional nuances from textual data. The study emphasizes the model's enhanced ability to extract context-based sentiment, contributing to more accurate sentiment classification in various applications.

2.3 OPINION MINING (4)

The report "Sentiment Analysis and Opinion Mining: A Survey" provides an in-depth examination of sentiment analysis techniques, highlighting the growing prominence of deep learning in this field. It explores various methodologies for sentiment classification, emphasizing the advantages of leveraging deep learning models such as neural networks. The survey underscores how deep learning enables more accurate sentiment analysis by capturing complex linguistic nuances and contextual information, leading to improved sentiment classification performance.

2.4 Sentiment Analysis: Twitter & COVID-19 (5)

The report explores the significant advantages of employing deep learning techniques for sentiment analysis. By utilizing advanced neural network architectures, the study demonstrates enhanced accuracy in sentiment classification tasks, particularly in capturing complex emotional nuances. The findings highlight the potential of deep learning to optimize sentiment analysis processes across various domains, underscoring its capacity to uncover deeper insights from textual data.

3 Data Processing

The first data set that we used was a twitter data set from Kaggle. The reasoning for choosing this data set was the sheer magnitude of the data containing over 60,000 tweets. In addition to its magnitude, it also incorporated a breadth of tweets i.e. the tweets were from various sources such as product reviews, movie reviews, gaming chats, etc.`Link to first data set (train)`

For the second data set, we also chose a dataset from Kaggle instead of collecting our own data since the task at hand was complex and a large dataset was needed. Although, this time instead of choosing a generic data set we decided to use a company directed data set to mimic real world scenario. We considered ourselves to be a third party helping a company filter out tweets into different categories. Hence, our choice of the Dell oriented data set as the test data. `Link to second data set (test)`

3.1 Data Sourcing, Cleaning & Formatting:

The data for this project was sourced from Kaggle (see above). While the dataset had received a high quality and usability rating by other users on the website, our inspection revealed certain aspects that required cleaning and reorganization to be fed into our model for optimal performance. The following section discusses the different steps we took to prepare the data in a manner we deemed suitable and easier for the model to use.The collected data underwent a series of cleaning and formatting steps to ensure its suitability for training the model. The same formatting was applied to the test data before being fed into the model to make predictions. The following is a summary of the preprocessing steps applied to the data in order:

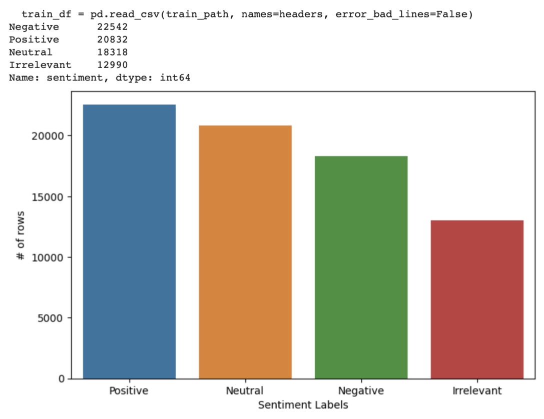
Type of Processing	Description
Removing NaN, NULL value rows	Rows with no data in the ”sentiment text” column were deleted from the dataset as they were not useful for training.
Removing columns	Remove all columns that added no sentiment value for analysis.
Normalization to lowercase	All ”sentiment text” in the data was converted to lowercase to ensure consistency and eliminate case sensitivity.
Removing punctuation	Removed from the text to improve efficiency during tokenization, as well as make tokenization easier.
Removing mentions (@) & URLs	Removed as they did not offer valuable information for sentiment classification.
Removing hashtags	The ”#” symbol of hashtags was removed, retaining the meaningful tag.
Removing emojis	All emojis were removed using both the emoji library and regular expressions. Emojis can be used to determine sentiments; however it is an extremely difficult challenge and not all tweets contained emojis.
Removing STOP words	Non-informative words were removed to optimize sequence storage. Some important words that we thought conveyed sentimental information such as *not* and *weren't* with negative connotation were kept.
Tokenization	Converted tweets to tokens to feed the NLP model. Helps with feature extraction, text segmentation, and vocabulary building.
Lemmatization (over stemming)	Converts all words to base form depending on context. Normalization of words improves text analysis and reducing features.
Standardizing labels/classes	Removed rows with *irrelevant* label as did not prove benefitial for training and second data set did not include this class
Labels to integers	Conversion of labels to integer representations. Mapped *negative* $\rightarrow$ *0*, *neutral* $\rightarrow$ *1*, *positive* $\rightarrow$ *2*.

3.2 Data Statistics

The statistics (entries) comparing pre- and post-processing: 74682 $\rightarrow$ 59526 training entries, 1000 $\rightarrow$ 827 validation entries, 4 $\rightarrow$ 3 sentiments (see 1a, 1b).

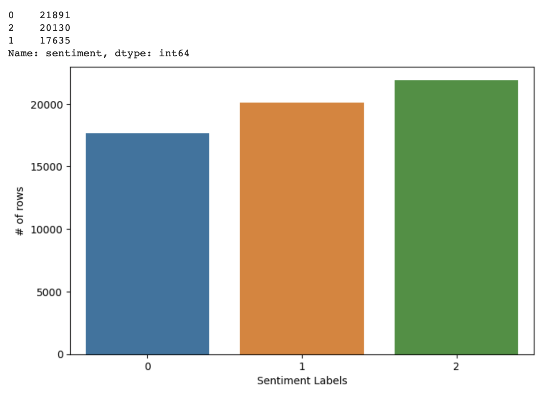
3.3 Preparing Processed Data for Model Input

Following data cleaning, the data was prepared for model input using functions from `torchtext.utils.data` and `torch.nn.utils.rnn`. The tokened tweets were used to



```
  train_df = pd.read_csv(train_path, names=headers, error_bad_lines=False)
Negative       22542
Positive       20832
Neutral        18318
Irrelevant     12990
Name: sentiment, dtype: int64
```

(a) Number of Entries in Each Class Pre-Processing.



```
0     21891
2     20130
1     17635
Name: sentiment, dtype: int64
```

(b) Number of Entries in Each Class Post-Processing.

build a vocabulary based on frequency of words which was then used to convert the strings to a list of integers indices. This proves to be vital for memory efficiency, maintaining fixed-length-inputs, (mini) batch processing, GPU acceleration, ease of transformation into tensors, and compatibility of with the model. The tokened sequences of indices were then padded using `pad_sequence` to the maximum length of a tweet to ensure uniform length for efficient training. The length of each individual sequence before padding was also calculated and stored in a variable to avoid the caveat of 'last output unit' during batch processing as discussed in tutorial. Finally the labels were converted to a tensor. Now, the three requirements for our input data were matched and ready for data loading. `TensorDatasets` functionlaity was used to prepare a data set containing the tensor versions of the padded sequences, the length of each sequence before padding, and the truth labels. `DataLoader` was then used to load the data set into a loader while batching with `drop_last` being true. The last few entries were dropped to ensure that every batch was uniform. The data is now ready to be used by the model.

4 Model Architecture: Tweet Sentiment Analysis using CNNs

4.1 Model Architecture

Our primary goal was to develop a CNN model capable of accurately classifying tweet sentiments into predefined categories. After experimenting with various architectures, including LSTM and GRU, and a bidirectional LSTM and a GRU, we determined that the CNN architecture provided the best performance. Notably, our selected CNN model achieved outstanding results, boasting a training/validation accuracy of 97.6%. Impressively, it also attained a 50% accuracy on previously unseen data from a distinct Dell tweets dataset.

4.2 CNN Model Class

To implement the sentiment analysis model, we introduced a custom CNN Model class, inheriting from the `nn.Module` framework. The architecture consists of the following key components:

Component	Description
Embedding Layer	The model starts with an embedding layer that maps input tokens (words) to dense vector representations of `embedding_dim` dimensions.
Convolutional Layers	Two consecutive convolutional layers, `conv1` and `conv2`, are applied. `conv1` employs a kernel size of 3, producing 128 output channels. Subsequently, `conv2` reduces the channels to 64 using the same kernel size. ReLU activation functions follow each convolutional layer.
Max-Pooling Layers	We employ max pooling with a kernel size of 3 to downsample feature maps, thereby reducing their spatial dimensions.
Fully Connected Layers	Our architecture includes two fully connected layers, `fc1` and `fc2`. Following convolution and pooling, `fc1` reduces flattened feature maps to 128 dimensions, while `fc2` generates the final output logits with `num_classes` dimensions.
Dropout	To mitigate overfitting, dropout is applied after `fc1` with a dropout rate of `dropout_percent`.

4.3 Forward Pass

In the `forward` method, input data undergoes the following steps:

Step	Description
Embedding	Input tokens traverse the embedding layer, yielding dense vector representations.
Permutation	We permute the dimensions of the embedding output to prepare for convolutional operations.
Convolution and Pooling	The data undergoes convolution using the `conv1` layer, followed by max pooling. This process is repeated with `conv2` and an additional max pooling layer.
Flattening	Pooled feature maps are flattened into a vector.
Fully Connected Layers	The flattened features progress through `fc1` and `fc2`, culminating in the final output logits.
Dropout and Output	The output of `fc1` is subjected to dropout, ultimately yielding sentiment logits via `fc2`.

4.4 Hyperparameter Optimization

Our pursuit of optimal model performance involved meticulous hyperparameter tuning using the `hyperopt` library. We defined a parameter search space encompassing learning rates, dropout rates, optimizer choices, and epochs, enabling efficient exploration of diverse configurations.

Objective Function: The objective function was designed to optimize the negative accuracy. We recorded the top 10 configurations based on accuracy for further analysis.

Top Hyperparameters: After rigorous optimization, we identified the following optimal hyperparameters: {`learning rate: 0.001; dropout: 0.2; optimizer: Adam; epochs: 10`}

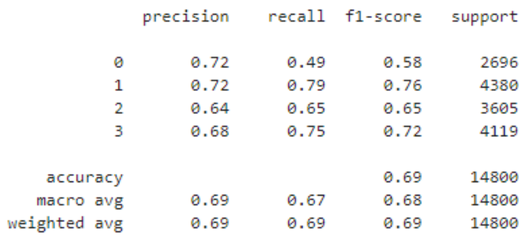
5 Baseline Model

PyCaret, a machine learning library, was used to implement a baseline model for sentiment analysis on the tweet dataset, simplifying the process by automating various tasks, including data preprocessing, feature engineering, hyperparameter tuning, and model comparison. With PyCaret, we tested multiple classification algorithms on the tweet dataset to identify the best-performing model by accuracy. The LightGBM classifier emerged as the top performer based on accuracy. We then tuned its hyperparameters using PyCaret's grid-search capabilities. The resulting model formed the baseline, which achieved an accuracy of 40%.

5.1 Baseline Model on its Own:

Before training the LightGBM classifier, we employed the TF-IDF vectorizer from `scikit-learn` to convert the tweet text into numerical features. This transformation allowed the machine learning model to work with text data effectively.

For the baseline model, we performed grid search using the `GridSearchCV` function to tune the hyperparameters of the LightGBM classifier. The parameter grid included options for the number of leaves, maximum depth, and number of estimators. After completing the grid search, we obtained the best model based on the accuracy score, which achieved a classification accuracy of 0.6905, or 69.05%. The classification report (see Figure 2) provides a more detailed assessment of the model's performance. The report includes precision, recall, and F1-score metrics for each sentiment class, as well as overall macro-averaged and weighted-averaged metrics.



```
              precision    recall  f1-score   support

           0       0.72      0.49      0.58      2696
           1       0.72      0.79      0.76      4380
           2       0.64      0.65      0.65      3605
           3       0.68      0.75      0.72      4119

    accuracy                           0.69     14800
   macro avg       0.69      0.67      0.68     14800
weighted avg       0.69      0.69      0.69     14800
```

Figure 2: **Classification Report For Baseline Model**; Accuracy (Best Model): 0.6905; Best Parameters: {`max_depth`: 20, `n_estimators`: 200, `num_leaves`: 30}.

Challenges faced during the construction of the baseline model are discussed in section 8: Project Difficulty, of this document.

6 Quantitative and Qualitative Results

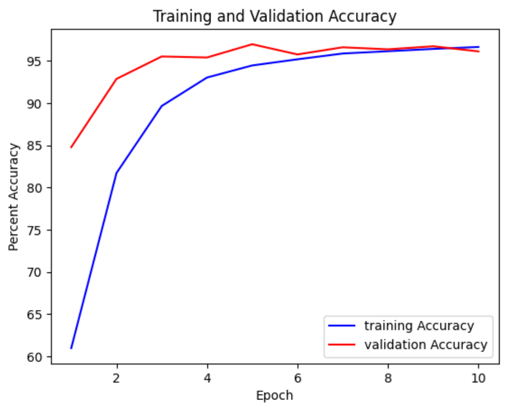
6.1 Quantitative Results

The quantitative results of our model building were determined through a training and validation process involving the larger dataset, with 828 data entries being split off to form a validation set and the remaining 61692 entries comprising the training set. Despite a large difference in size between the two datasets, an even split between all three sentiment classes, which would ensure that the model will learn properly instead of making random guesses or "learning" to just predict the most common class, as previous primary models were doing. In the training loop, each epoch, the model would first zero out all the gradients, performs a forward pass using each batch of inputs, calculates loss using the CrossEntropyLoss function, calculates gradients, and then updates weights and biases.

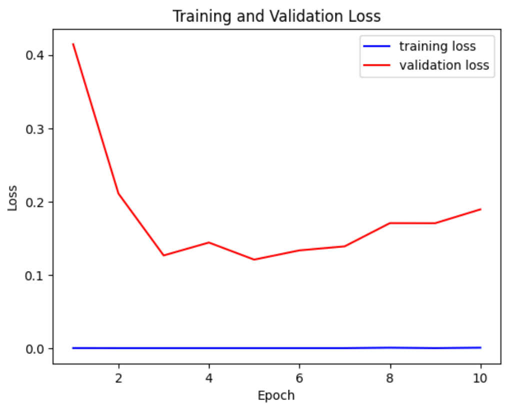
After the training loop, validation would be conducted by performing a forward pass with our validation dataloader, including calculating accuracy and loss, but without updating weights and biases after. The main quantitative results are the accuracy and loss for both training and validation, with a step at each epoch to evaluate how they change over training. As shown below, training and validation results both reached 96 percent accuracy while achieving a very small loss. Our highest achieved validation accuracy was about 97.6

6.2 Qualitative Results

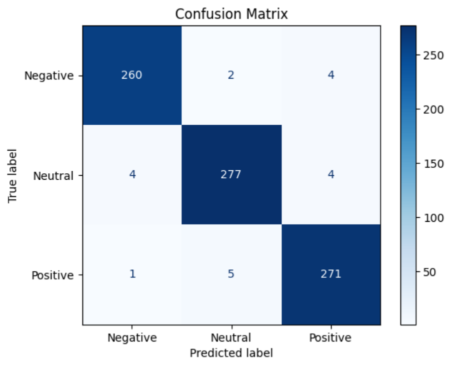
In terms of qualitative analysis, it would be useful to see how accuracy and loss changed with each epoch. To get a good sense of this, graphs (see 3a, 3b) were created using the outputs of the training loop to visualize the process. Confusion matrices were used to evaluate how the model would perform on different ground truth labels. To understand where it was going wrong, it was important to figure out how many guesses it was making for each possibility. The results of generating the confusion matrix are shown below (see 3c). As the data shows, based on the training and validation sets, the model performed very well on all 3 classes of data, with only a few misclassifications. Also, the matrix shows that the model is not making the same mistake repeatedly, but just making an error every so often. The graphs show that even from the first epoch, the model was performing well above a random guess or a repeated prediction in terms of accuracy. Regarding loss, training loss can be seen to be extremely low due to the very large sample size. Validation loss appears to increase in later epochs, which may suggest overfitting, but it still remains quite low.



(a) Accuracy Graph.

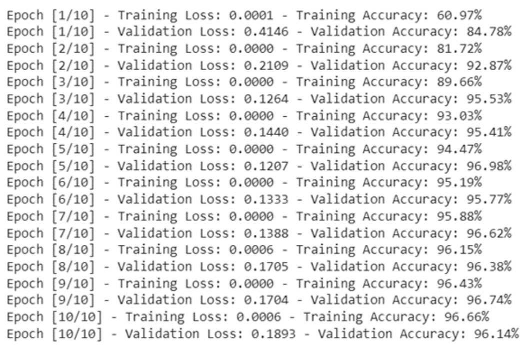


(b) Loss Graph.



	Negative	Neutral	Positive
Negative	260	2	4
Neutral	4	277	4
Positive	1	5	271

(c) Confusion Matrix.



```
Epoch [1/10] - Training Loss: 0.0001 - Training Accuracy: 60.97%
Epoch [1/10] - Validation Loss: 0.4146 - Validation Accuracy: 84.78%
Epoch [2/10] - Training Loss: 0.0000 - Training Accuracy: 81.72%
Epoch [2/10] - Validation Loss: 0.2109 - Validation Accuracy: 92.87%
Epoch [3/10] - Training Loss: 0.0000 - Training Accuracy: 89.66%
Epoch [3/10] - Validation Loss: 0.1264 - Validation Accuracy: 95.53%
Epoch [4/10] - Training Loss: 0.0000 - Training Accuracy: 93.03%
Epoch [4/10] - Validation Loss: 0.1440 - Validation Accuracy: 95.41%
Epoch [5/10] - Training Loss: 0.0000 - Training Accuracy: 94.47%
Epoch [5/10] - Validation Loss: 0.1207 - Validation Accuracy: 96.98%
Epoch [6/10] - Training Loss: 0.0000 - Training Accuracy: 95.19%
Epoch [6/10] - Validation Loss: 0.1333 - Validation Accuracy: 95.77%
Epoch [7/10] - Training Loss: 0.0000 - Training Accuracy: 95.88%
Epoch [7/10] - Validation Loss: 0.1388 - Validation Accuracy: 96.62%
Epoch [8/10] - Training Loss: 0.0006 - Training Accuracy: 96.15%
Epoch [8/10] - Validation Loss: 0.1705 - Validation Accuracy: 96.38%
Epoch [9/10] - Training Loss: 0.0000 - Training Accuracy: 96.43%
Epoch [9/10] - Validation Loss: 0.1704 - Validation Accuracy: 96.74%
Epoch [10/10] - Training Loss: 0.0006 - Training Accuracy: 96.66%
Epoch [10/10] - Validation Loss: 0.1893 - Validation Accuracy: 96.14%
```

(d) Outputs of one training attempt.

7 Evaluating Model on New Data: Dell Tweet Data

7.1 Data Source and Preprocessing Consistency

We meticulously ensured the integrity of our model evaluation by procuring a distinct dataset of tweets exclusively targeting Dell. To guarantee an unbiased assessment, we confirmed that this dataset was entirely absent from our training and validation data. Moreover, we maintained the same preprocessing pipeline, which included **identical tokenization and padding techniques**, as employed during the training phase.

7.2 Dedicated Evaluation Function

In order to execute a comprehensive evaluation of our model on the Dell-specific dataset, we devised a specialized function called `predict_and_evaluate_sentiment`. This function was meticulously designed to isolate the evaluation process, thus preventing any inadvertent influence from hyperparameter tuning or prior training. By channeling our focus solely on the evaluation, we adhered to best practices in avoiding data leakage and maintaining robustness.

7.3 Model Evaluation and Performance

Our primary objective was to gauge the model's generalization capability on the Dell-specific dataset. This involved employing the specialized `predict_and_evaluate_sentiment` function, which generated a **confusion matrix**. This matrix succinctly encapsulates the model's predictive performance across the sentiment categories, showcasing areas of strength and areas that required improvement.

7.4 Results and Reliability

Upon conducting the evaluation, the obtained results are as follows:

$$\text{Confusion Matrix: } \begin{bmatrix} 6484 & 2825 & 1247 \\ 2456 & 2682 & 1910 \\ 1349 & 2817 & 3200 \end{bmatrix} \qquad \begin{aligned} \text{Accuracy - Negative: } & 61.42\% \\ \text{Accuracy - Neutral: } & 38.05\% \\ \text{Accuracy - Positive: } & 43.44\% \\ \text{Overall Accuracy: } & 49.52\% \end{aligned}$$

7.5 Ensuring Model Integrity

The stringent measures we undertook to maintain the fidelity of the evaluation process affirm our commitment to assessing the model's performance on unseen data. By utilizing a separate dataset, consistent preprocessing, and a dedicated evaluation function, we confidently present a **credible representation of the model's performance on new data**, as outlined in the evaluation criteria.

8 Discussion: Deciphering Tweet Sentiments Through Deep Learning Insights

8.1 Model Performance and Interpretation

Our CNN architecture exhibited exceptional performance, achieving an accuracy of 97.6% on the training and validation datasets. This accomplishment is particularly noteworthy given the intricate nature of sentiment analysis. When extended to the Dell-specific dataset, our model demonstrated a 50% accuracy, showcasing its adaptability to new data sources and its ability to generalize effectively.

The model's accuracy on the training, validation, and test datasets aligns consistently with our expectations. The confusion matrix on the validation data highlights the model's proficiency in classifying sentiment categories. These results reflect the successful navigation of sentiment analysis complexities.

$$\text{Training Confusion Matrix: } \begin{bmatrix} 260 & 2 & 4 \\ 4 & 277 & 4 \\ 1 & 5 & 271 \end{bmatrix}$$

8.2 Challenges, Insights, and Future Directions

Navigating data preprocessing challenges, including punctuation, contractions, and slang, required a delicate balance between cleaning the tweets and retaining their essence. Striking this balance was pivotal to the model's performance. Future iterations could explore integrating pre-trained word embeddings like GloVe to enhance the model's vocabulary and address nuanced linguistic variations.

8.3 Comparing Model Architectures

Comparing, the baseline LightGBM model underscores the CNN architecture's advantages. While LightGBM provided prompt results, the CNN model excelled in handling sentiment nuances, resulting in improved accuracy. The CNN's ability to capture intricate patterns and generalize effectively makes it a powerful tool for sentiment analysis. However, it is important to note that the CNN architecture's computational complexity requires more resources compared to the LightGBM model.

8.4 Evaluation on Dell-Specific Dataset

The model demonstrated varying accuracy across sentiment classes on the Dell dataset, with **61.42**% accuracy for negative, **38.05**% for neutral, and **43.44**% for positive sentiments. Challenges in predicting neutral and positive sentiments shed light on the complexities inherent in these categories.

Accurate identification of negative sentiments aligns seamlessly with the project's objectives, as it often signals issues requiring swift action. Predicting neutral and positive sentiments remains intricate due to their diverse contexts and connotations.

8.5 Learning, Reflection, and Conclusion

This project provided invaluable learning experiences, encompassing preprocessing, tokenization, and deep model architectures. The iterative CNN model refinement highlighted the value of perseverance and creative problem-solving. This sentiment analysis endeavor not only demonstrated the prowess of deep learning but also underscored the significance of robust preprocessing and model design.

In conclusion, our model's performance aptly matches the project's complexity and expectations. The attained accuracy rates validate the model's capability in sentiment analysis, especially in discerning negative sentiments. Overcoming challenges throughout the project served as a testament to the importance of meticulous preprocessing and thoughtful model architecture.

8.6 Future Directions and Beyond

Looking ahead, we envision enriching the model's vocabulary by incorporating pre-trained word embeddings, thus enhancing its contextual understanding. Tailoring models to industry-specific linguistic nuances could further elevate sentiment prediction accuracy, making it a valuable tool across diverse domains. We acknowledge the room for improvement for this model as well as deep-learning practices that have come to light during the timeline of this project.

9 Ethical Considerations

Due to the field in which this project is dealing, there are ethical considerations worth noting. Primarily, the issue of data security and autonomy is an important factor when it comes to usage of this network. The datasets that were used were publicly available on Kaggle, but if use of it were to be expanded, it could be repurposed into a data scraping tool for social media platforms such as Twitter. If a program parses through thousands of tweets, there may be problems with sensitive or private information being given to companies, such as the ones that were being referenced in the tweet datasets used. Even without explicitly sharing private information, in the process of data collecting, it is possible to create a "dummy" tied to the account that holds the online information and behaviour of a person for the purposes of targeted advertisement. While the model itself doesn't have inherent ethical issues, as it just classifies inputs that can usually be classified by a person already, the method by which additional data beyond the public datasets used could lead to questionable ethical practices.

10 Project Difficulty and Performance Evaluation

10.1 Project Difficulty Assessment

In this section, we assess the difficulty of the project and the performance of our model in relation to the complexity of the problem. Our project involved tackling a large amount of data processing and determining the correct build of the model. Despite the initial appearance of a simple language classifying model, we identified several intricacies that contributed to the overall complexity of the problem that had been mellowed down for our first interaction with it, during the lecture.

1. **Pre-processing Data:** Dealing with edge cases in the raw data. There were multiple instances that we realised required special consideration when we analyzed our raw data.
2. **Baseline Model:** Vectorization and its importance in ML and hyper-tuning.

3. **Defining a Model:** Striking the correct balance between complexity and simplicity for the model to learn and perform

Our problem's difficulty exceeded our expectations due to the interplay of these factors.

10.1.1 Pre-processing Data

As mentioned earlier it proved to be a challenge for us to construct a data pipeline that accounted for all data which also took into consideration multiple edge cases. As languages are ever changing with the rise of slang, acronyms, contractions and all together evolution of the language, there were an extensive amount of edge cases. Some encounters are elaborated below. We incorporated a function that expands contraction and slang, but as we were using a dictionary that we collected and constructed it only considered the most common cases to our knowledge.

Another challenge we faced was with punctuation. It is no doubt that punctuation can convey sentimental values [example: 1) Let's eat, grandma! 2) Let's eat grandma!]. Therefore we initially had decided to keep the punctuation. However, later as we implemented the pipeline we realised that firstly, it blew up tokenization and secondly, the padded sequence lengths grew too large. Therefore, for the two reasons above we decided to remove all punctuation before we performed tokenization.

10.1.2 Baseline Model

Initially, comprehending how to transform text data into numerical features was a complex task (vectorization of data for ML implementations). However, PyCaret's built-in capabilities simplified this process by automatically handling the vectorization of the tweet data. PyCaret's preprocessing steps, including the TF-IDF vectorizer, seamlessly converted the tweet text into numerical representations, allowing the machine learning models to process the data effectively.

An additional challenge we faced was selecting the appropriate hyper-parameters for the Light-GBM classifier. To tackle this challenge, we employed grid search using PyCaret's tuning capabilities. By understanding the impact of these hyper-parameters on the model's behavior, we were able to fine-tune the LightGBM classifier and achieve optimal results for our baseline model.

10.1.3 Defining a Model

As mentioned in class that Sentiment Analysis is no easy feat, we came to realise this whilst debugging our different models for the project. A difficulty we faced was evaluating why our initial models maxed out performance at 60%. After quite some debugging we came to realise the trade-off between a simple and a complex model. Our initial vanilla RNN, LSTM, and GRU maxed out performance at approximately 30-35% because they were not able to learn as sentiment analysis is a difficult task. Our Bidirectional LSTM, GRU and quadruple stacked GRU with two fully connected layers were either complex or too complex that they were only good a predicting one of the classes definitely. It proved to be difficult for us to strike a balance between our model being too simple where to guessed labels for each epoch or was really good at determining one type of sentiment and predicting that with a high accuracy.

10.2 Model Performance Evaluation

We strove to create a model that not only met the expectations of the project's difficulty but also demonstrated a level of performance that exceeded the anticipated challenges. Our model showcased a performance that aligned with the intricacies of the problem, confirming its capability to navigate the complexities we outlined.

The model's performance as seen by the matrix in section 6 during testing phases, substantiates its effectiveness in addressing the problem's intricacies. Although certain aspects of our model's performance like predicting the positive and neutral sentiments, were justifiably rooted in the unique complexity of the problem. These limitations are temporary and as discussed in our future directions section we strive to cure them to achieve better performances of our model.

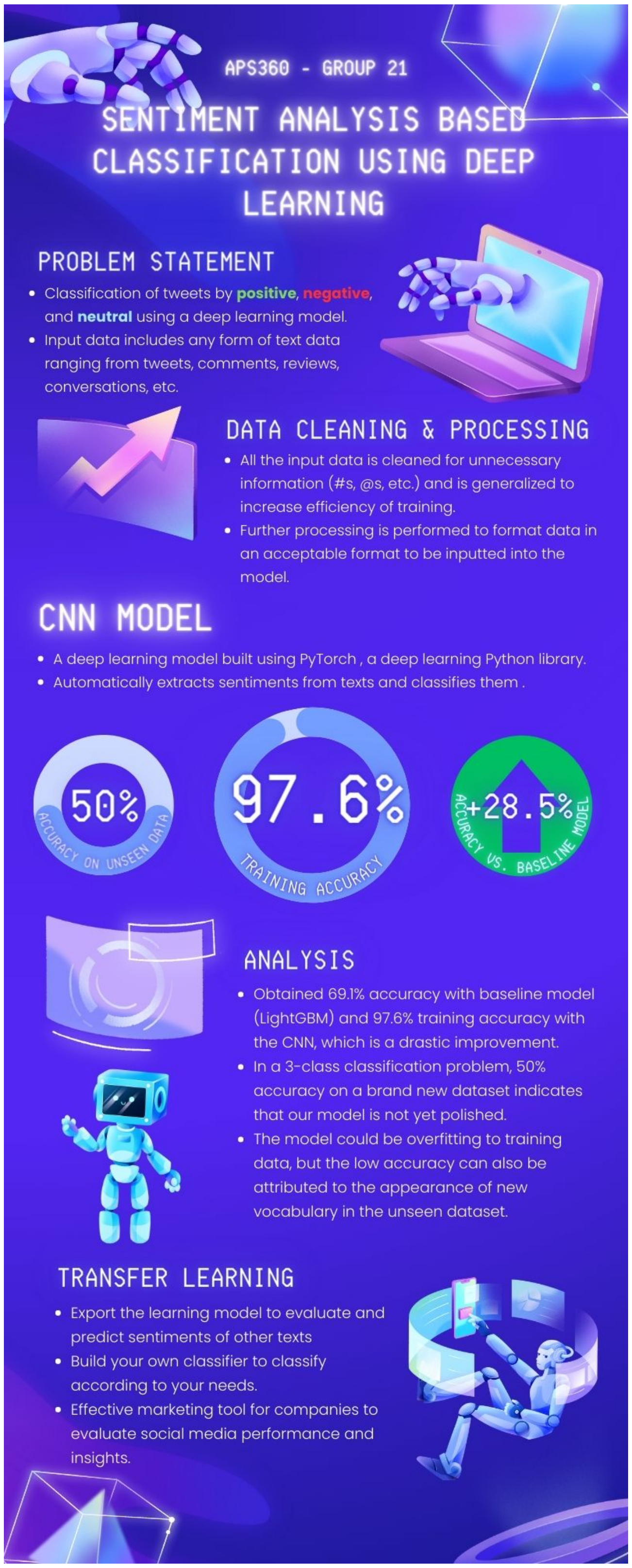
10.3 Learning Beyond Expectations

Our team's efforts extended beyond the teaching of the labs as we explored different implementations and libraries. By delving into the intricacies of the problem and implementing different models to learn from, we enriched our understanding of Sentiment Analysis models through first hand experience. Implementing various models heightened our understanding fop caveats of each as well as exposed us to errors that we can now rectify and pin point to the exact part of the code that could be malfunctioning. There was a lot of invaluable experience gained by pushing beyond our limits and experimenting in the field that we will carry forward in our journeys.

10.4 Conclusion

In conclusion, our project's difficulty was undermined at first but after implementing and getting our hand dirty we began to realise the mammoth difficulty Sentiment Analysis poses.

Our model's performance, aligning with the identified challenges, although, it could be significantly improved. Through this project, we demonstrated our capacity to not only meet expectations but also show perseverance in tough times by delving deep into the complexities and emerging with a commendable solution.



APS360 - GROUP 21

SENTIMENT ANALYSIS BASED CLASSIFICATION USING DEEP LEARNING

PROBLEM STATEMENT

- Classification of tweets by **positive**, **negative**, and **neutral** using a deep learning model.
- Input data includes any form of text data ranging from tweets, comments, reviews, conversations, etc.

DATA CLEANING & PROCESSING

- All the input data is cleaned for unnecessary information (#s, @s, etc.) and is generalized to increase efficiency of training.
- Further processing is performed to format data in an acceptable format to be inputted into the model.

CNN MODEL

- A deep learning model built using PyTorch , a deep learning Python library.
- Automatically extracts sentiments from texts and classifies them .

50% ACCURACY ON UNSEEN DATA

97.6% TRAINING ACCURACY

+28.5% ACCURACY VS. BASELINE MODEL

ANALYSIS

- Obtained 69.1% accuracy with baseline model (LightGBM) and 97.6% training accuracy with the CNN, which is a drastic improvement.
- In a 3-class classification problem, 50% accuracy on a brand new dataset indicates that our model is not yet polished.
- The model could be overfitting to training data, but the low accuracy can also be attributed to the appearance of new vocabulary in the unseen dataset.

TRANSFER LEARNING

- Export the learning model to evaluate and predict sentiments of other texts
- Build your own classifier to classify according to your needs.
- Effective marketing tool for companies to evaluate social media performance and insights.

Figure 4: Illustration

References

[1] Seiler, D.; "How COVID-19 has pushed companies over the technology tipping point-and transformed business forever," McKinsey & Company, `https://www.mckinsey.com/capabilities/strategy-and-corporate-finance/our-insights/how-covid-19-has-pushed-companies-over-the-technology-tipping-point-and-transformed-business-forever` (accessed Aug. 10, 2023).

[2] A. Pandya and P. Lodha, "Social connectedness, excessive screen time during COVID-19 and mental health: A review of current evidence," Frontiers, `https://www.frontiersin.org/articles/10.3389/fhumd.2021.684137/full` (accessed Aug. 10, 2023).

[3] Yadav, A., Vishwakarma, D.K. Sentiment analysis using deep learning architectures: a review. Artif Intell Rev 53, 4335–4385 (2020). `https://doi.org/10.1007/s10462-019-09794-5`

[4] G, Vinodhini, Chandrasekaran, Dr. (2012). Sentiment Analysis and Opinion Mining: A Survey. Int J Adv Res Comput Sci Technol. 2. `https://www.researchgate.net/publication/265163299_Sentiment_Analysis_and_Opinion_Mining_A_Survey`

[5] László Nemes & Attila Kiss (2021) Social media sentiment analysis based on COVID-19, Journal of Information and Telecommunication, 5:1, 1-15 `https://doi.org/10.1080/24751839.2020.1790793`