ZEIT4151 Machine Learning Assignment 4B

Predictive Text Module Development

Module 3A: Word-Level Predictive Text Module

Author: Harrison Faure

Student ID: z5364422

Date: November 4, 2024

Table of Contents

Contents

1 Introduction

1.1 Background and Motivation

Amyotrophic Lateral Sclerosis (ALS) impairs nerve cells in the brain and spinal cord due to its neurodegenerative nature. The disease leads to severe muscle control loss and significant mobility limitations, and can affect traditional communication methods. Leveraging eye movements and brain waves lead to the creation of Human-machine interfaces (HMIs) which improve the quality of life for patients by enabling text prediction. Natural Language Processing (NLP) plays a critical role alongside Machine Learning (ML) in helping to develop these predictive modules to streamline communication which offers ALS patients an efficient way to convey messages.

1.2 Objective

The aim was to develop a word-level predictive text module to suggest a word based on previously typed words providing 3 different options to improve text input efficiency for ALS patients. This included analyzing a dataset from Project Gutenberg mapping text prediction to an ML model and training three different ML models evaluating their performance and suitability to support an effective HMI system.

1.3 Approach Overview

There were three ML models implemented:

1. **Recurrent Neural Network (RNN):** This was a basic model utilized as RNNs are well suited for sequential data and can use past information to predict future words.
2. **Gated Recurrent Unit (GRU):** GRUs are better at addressing the vanishing gradient problem and the GRU model was implemented to hopefully improve learning of long-term dependencies.
3. **Long Short-Term Memory (LSTM):** LSTMs retain information over longer sequence and therefore the LSTM aims to enhance prediction accuracy by capturing context from longer input sequences.

1.4 Report Structure

The report is structured as follows:

- **Section 2 - Dataset**: Dataset characteristics, preprocessing steps, and encountered challenges.
- **Section 3 - ML Models**: The implementation of three ML models, including data splitting and selection rationale.
- **Section 4 - Results**: The experimental setup, parameter tuning, and performance comparison.
- **Section 5 - Conclusion**: Summarises findings.

2 Dataset

2.1 Dataset Description

The dataset used was text from *The Adventures of Tom Sawyer* by Mark Twain, found in the Project Gutenberg.

This novel provides a substantial amount of textual data suitable for training.

2.1.1 Size and Features

- **Dataset Length:** 1,954,026
- **Total Number of Words:** 367022
- **Unique Words:** There are 18,450 unique words.
- **Rare Words:** Out of the unique words, 8,265 words appear only once.
- **Average Word Length:** 4.18 characters

Each word in the dataset is a feature for the predictive model which are represented as tokens and the sequences of tokens are used to predict the next word.

2.1.2 Redundancy and Missing Data

The dataset does not contain missing data as it is a continuous text without gaps. There is redundancy in the form of frequently occurring words which may dominate the dataset and influence the training process.

2.1.3 Descriptive Statistics

To get insights into the dataset, the following statistics were calculated:

Word Frequency Distribution The frequency distribution shows that the dataset follows Zipf's law, common in natural language (NLP) processing tasks.

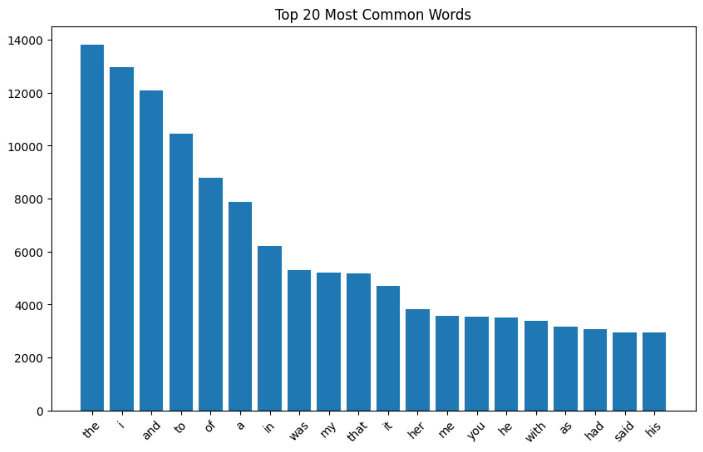

Figure 1: Word Frequency Distribution

- **Occurrence of Common Words:**

Table 1: 10 Common Words

Word	Frequency
the	13179
and	11952
to	10410
a	7829
of	8732
was	5298
he	2919
in	6114
that	5154
it	4288

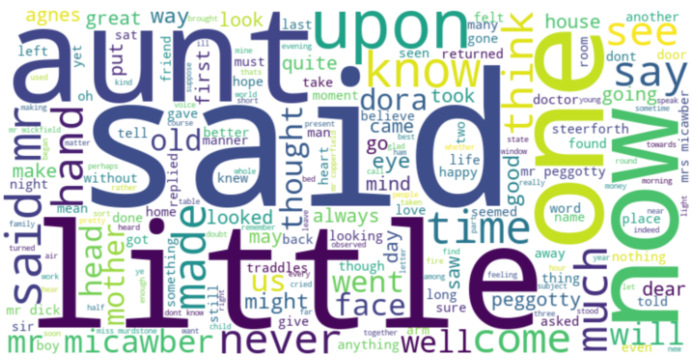

Figure 2: Word Cloud

Sentence Length Distribution

- **Total Sentences:** 23144.
- **Average Sentence Length:** 15.86 words.
- **Sentence Length Range:** 1 to 233 words per sentence.

2.1.4 Data Distribution

The dataset has a highly imbalanced distribution with a few words occurring very frequently and lots of words occurring rarely. This causes a challenge for training the model to learn less frequent words

2.1.5 Data Separability

Separability is the the model's ability to distinguish between different word sequences and relates to the accuracy of predicting the next word. Separability is very complex and depends on capturing long-term dependencies due to the sequential and contextual nature of the english language.

2.2 Data Pre-processing

The pre-processing is the most important step to clean the raw text and convert it to a suitable format:

2.2.1 Pre-processing Steps

(a) **Removing Boilerplate Text:** Removed using regular expressions.

(b) **Lowercasing:** Text was converted to lowercase reducing the vocabulary size and maintaining consistency.

(c) **Expanding Contractions:** Contractions were expanded using the `contractions` library.

(d) **Converting Numbers to Words:** Numerical digits were converted to words using the `num2words` library.

(e) **Removing Punctuation:** Punctuation was removed.

(f) **Whitespace Normalization:** Extra whitespace was removed.

2.3 Dimensionality Reduction

2.3.1 Use of Embeddings

Word embeddings were used to apply dimensionality reduction on the dataset - the high-dimensional representation of words (one-hot) was transformed into dense, lower-dimensional vectors through embedding layers.

2.3.2 Justification

The approach used aimed to capture semantic relationships between words and reduce complexity enabling the model to learn more efficiently from the data.

2.4 Challenges with the Dataset

Some of the challenges included:

- **High Vocabulary Size:** 18,450 unique word is massive and increases computational complexity.
- **Rare Words:** 8,265 words appear only once, causing a heavy-tailed distribution.
- **Class Imbalance:** Frequent words dominate, which causes bias.
- **Contextual Complexity:** Presence of archaic language.

3 Machine Learning Models

3.1 Data Splitting Strategy

the dataset was tokenizied which converts the text into sequences used to train the predictive models. The data splitting involved ensuring the models can learn effectively from the data whilst reliably evaluating the performance.

Sequences of fixed text were generated using a sliding window approach - for each position in the text a sequence of the previous 30 words was used as an input and the next word used as the target label.This creates a large amount of input-output pairs suitable for training. The sliding window approach may cause data leakage between validation and training sets due to how it works. Within this time frame that is unavoidable but a large scale implementation of this would want to reduce that.

3.1.1 Filtering and Oversampling

To address the class imbalance and issues regarding rare words the sequences that contain the rare words(words appearing less than or equal to two times) were over sampled by by a factor of 5 to 10 times. This was done to ensure that the models had sufficient exposure to the less frequent words. The oversampling helps the model predict rare words more accurately.

3.1.2 Data Shuffling and Splitting

Training and validation sets should also both be a representation of the overall data distribution therefore the dataset was shuffled to ensure that there is a good distribution of sequences. The Training and validation splits was 90%–10% split. This helps the model generalized better and perform well on unseen data and helps to combat over fitting.

3.2 Model Selection

Three different neural networks were selected to be the model Recurrent Neural Network (RNN), Gated Recurrent Unit (GRU), and Long Short-Term Memory (LSTM). These were chosen for there ability to capture temporal dependencies which is important for predicting the next word.

3.2.1 Recurrent Neural Network (RNN)

The RNN is a baseline for data modeling as it processes input sequences one element at a time whilst maintaining an internal state carrying information about previous atempts. this allows the model to capture-short term dependence well however it may suffer from the vanishing gradient problem. (Bengio et al., 1994).

Implementation Details The implementation used PyTorch and consisted of:

- An embedding layer which converts word indices into vectors.
- Two RNN layers with sizes of 128 and 64.
- A fully connected layer.
- An output layer predicting the next word.

3.2.2 Gated Recurrent Unit (GRU)

The GRU model should be an improvement over the RNN, it introduces gating mechanisms to help capture long-term dependencies and combat the vanishing gradient problem (Cho et al., 2014). The GRU model should reduce computational complexity while maintaining performance.

Implementation Details The implementation used Keras and consisted of:

- An embedding layer
- Two GRU layers with 128 and 64 units.
- A fully connected layer
- An output layer with softmax activation.

3.2.3 Long Short-Term Memory (LSTM)

The LSTM model uses memory cells and gating mechanisms allowing it to learn when to forget previous hidden states and when to update them(Hochreiter and Schmidhuber, 1997). This architecture enables LSTMs to capture long-term dependencies efficently.

Implementation Details The implementation used Keras and consisted of:

- An embedding layer.
- Two stacked LSTM layers with 128 and 64 units.
- A fully connected layer.
- An output layer with softmax activation.

3.3 Model Ranking

Based on the expected performance and the characteristics of each model as well as their suitability for the prediction task, the models were ranked for expected performance as:

1. **LSTM Model**
2. **GRU Model**
3. **RNN Model**

3.3.1 Justification

The LSTM model has the ability to capture long term dependencies efficiently in the text via its gating mechanisms and therefore it is anticipated to perform the best as it will determine which information to keep and which inform to discard across sequences making it effective. The GRU model has a simpler structure and should perform slightly less effectively than the LSTM model. The model is expected to struggle with long term dependencies even though it should be more resource efficient.(Chung et al., 2014). The RNN model will have the lowest performance as it is effected by the vanishing gradient problem and lacks the ability to retain context during large sequences.(Bengio et al., 1994).

3.3.2 Reflection on Model Selection

The ranking of the models and selection for the predictive task was done based on the models architecture and handling of sequential data. The superiority ability of the LSTM to retain information over the long sequence should make it the most suitable whereas the GRU offers a more balanced approach between performance and efficiency. The RNN will be used as a baseline to compare the improvements offered by the advanced architectures.

3.3.3 Anticipated Performance Metrics

The LSTM is expected to achieve higher accuracy and lower loss for both training and validation when compared to the RNN and GRU models The RNN is expected to converge and perform slower with higher loss due to its limitations.

4 Results

4.1 Experimental Setup

The set up involved the implementation of the three models - recurrent neural network (RNN), a gated recurrent unit (GRU), and a long short-term memory (LSTM) network. The models were trained on the exact same data for comparison which consisted of the text extracted from the first 10 chapters of the text. The default parameters were selected after tuning.

The input sequence was selected to be 30 tokens with a batch size of 32 tokens for all models. The size of the vocabulary varied depending on the tokenizer but was generally around 16000 words. The models were trained using the adam optimizer for 10/20 epochs where relevant with a learning rate that varied based on the model. Sparse categorical cross-entropy was the loss function used and the metrics tracked accuracy and sparse top-3 categorical accuracy.

4.2 Parameter Tuning

The LSTM model underwent a substantial amount of parameter tuning and iterations due to its superior performance initially. Each iteration focussed on adjusting different parameters to improve model performance as described below.

4.2.1 Iteration 1

Iteration one used an embedding dimension of 10 and a sequence length of 12 the architecture included two LSTM layers with 128 and 64 units followed by a dense layer of 64 units. The model was trained with a batch size of 32 for 10 units and showed to be under fitting with training and validation plateauing early.

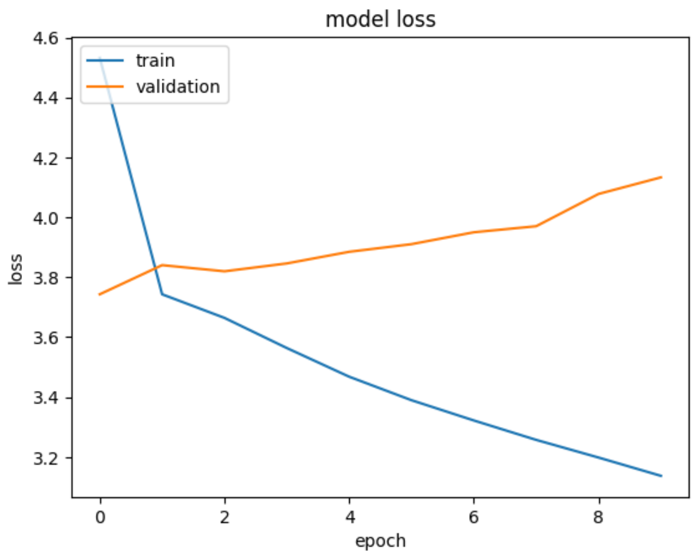

Figure 3: Training and Validation Accuracy and Loss - LSTM Iteration 1

4.2.2 Iteration 2

The sequence length remained the same however this time the model was trained on the entire cleaned dataset. This increased the vocabulary size and the model achieved a perfect accuracy - this indicated data leakage and that the model was just memorising the dataset which would not generalise well.

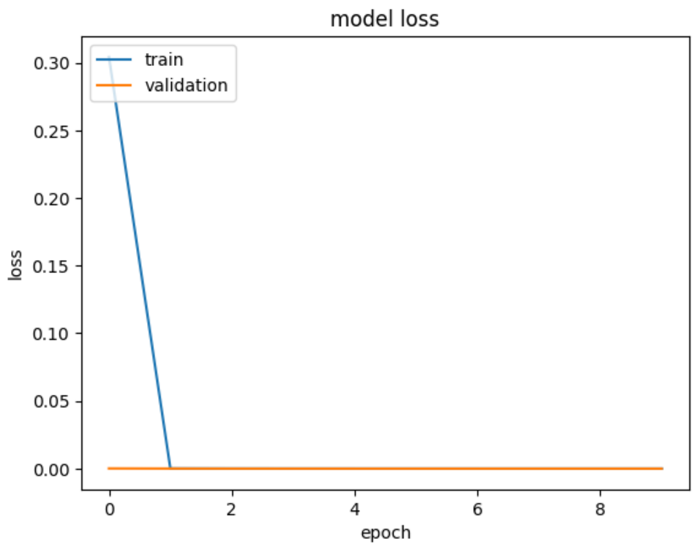

Figure 4: Training and Validation Accuracy and Loss - LSTM Iteration 2

4.2.3 Iteration 3

The third iteration increased the embedding dimensions up to 100 and the sequence length extended to 30 to provide the model with higher context. Dropout layers were also experimented in this model to try and prevent the overfitting. The model failed to compile possibly due to too many changes at once and the increased complexity

4.2.4 Iteration 4

The 4th iteration focused on trying to make the code more modular to try to determine what was actually wrong with the model. The pre-processing of the data was also greatly enhanced leading to better text cleaning. The tokenizer was also adjusted to better handle the tokens that were out of vocabulary - reprocessing improved the quality of the data whoever the model did not perform well suggesting the parameters needed adjusting.

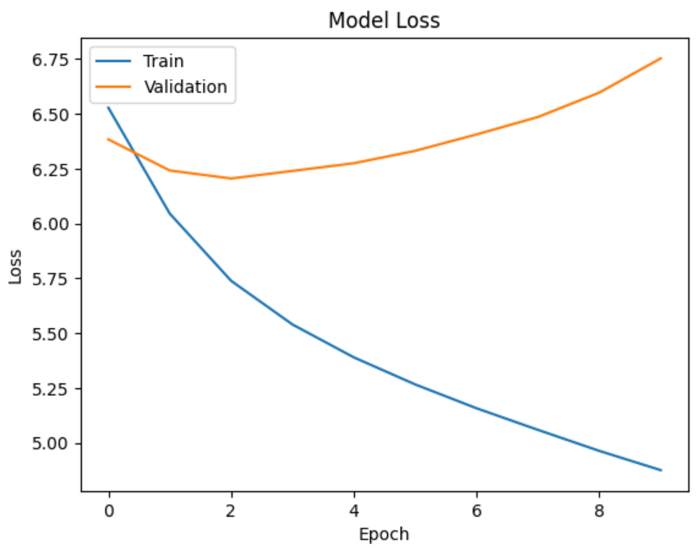

Figure 5: Training and Validation Accuracy and Loss - LSTM Iteration 4

4.3 Performance Comparison

Table 2 gives the performance metrics of the 3 models - using the final iteration of the LSTM model which has not yet been disccused. The RNN model demonstrated declining performance over the training epochs with a decreasing accuracy from 12.93% to 10.46%. The GRU model performed better but also seemed to converge indicating it is experiencing errors of some kind - the validation accuracy peaked at 9.78%.

The final LSTM model showed significant improvement over the over two models achieving a validation accuracy of 8.17% on the 20th epoch. The loss also steadily decreased over the epochs suggesting good convergence rate. This superior performance is to be expected and is attributed to the models ability to better handle long term dependencies as well as the parameter tuning that was conducted.

4.4 Final Parameter Selection

The Final LSTM model was configured as follows:

- **Embedding Dimension**: 100
- **Sequence Length**: 30
- **Batch Size**: 32
- **Learning Rate**: 0.001
- **Optimiser**: Adam
- **Architecture**:
  - Embedding layer.
  - Two LSTM layers with 128 and 64 units.
  - Dense layer with 64 units.
  - Output dense layer.

Increasing the embedding dimensions allowed the model to learn more nuanced word respresentations and the learning rate was kept moderate to ensure stability.

4.5 Tabular and Graphical Representations

Figure 8 Shows the validation accuracy and loss of the final LSTM model over the epochs. The graph demonstrates a steady increase in accuracy and a decrease in loss which is effective leanring.

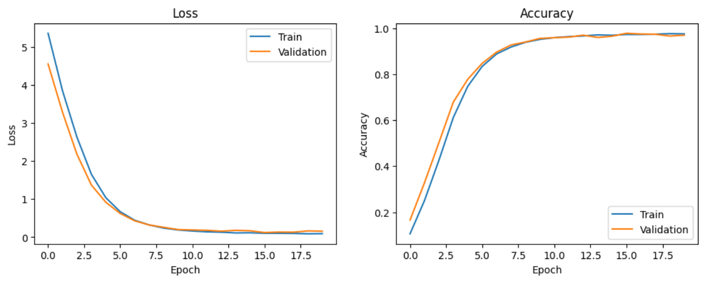

Figure 6: Training and Validation Accuracy and Loss for the Final LSTM Model

Table 3 tabulates the parameter changes and performance metrics across the four iterations of the LSTM model whilst the tuning is occurring. The FPS value is another comparative measured used to compared the three models and is given in Table 4. The RNN achieved the highest FPS which can be attributed to it having the simplest architecture whereas the LSTM had a reduced FPS because of its increased complexity which is a trade off that had to occur.

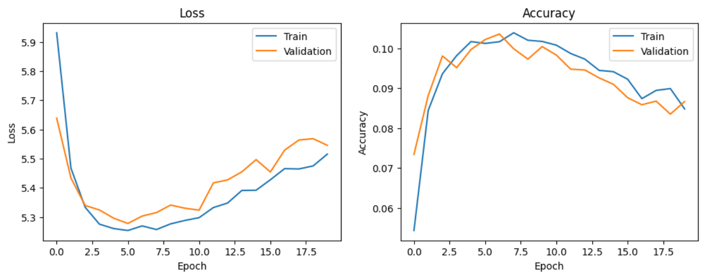

Figure 7: Training and Validation Accuracy and Loss - GRU

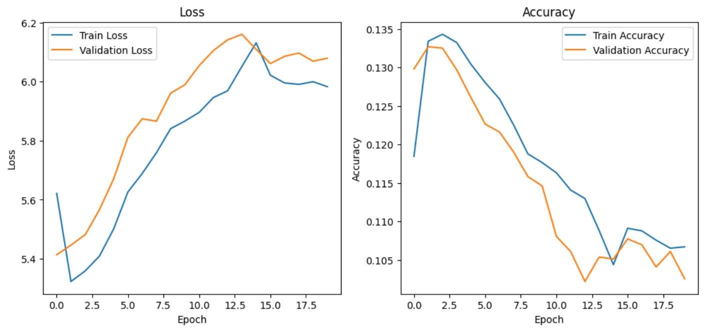

Figure 8: Training and Validation Accuracy and Loss - RNN

4.6 Sampling Distribution Analysis

The top 100 and bottom 100 token distributions were also graphed before and after sampling for visualization in Figures 9, 10, 11, and 12. This was to demonstrate the effects of the oversampling the tail end of the data with the objective to balance the dataset by increasing the frequency of the rarer words.

The oversampling creates a more uniform distribution with the bottom 100 words now occurring 300 ish times. This helps to even the class imbalance issues allowing the model to generalize a whole lot better. The most frequent words still remain common which is important as they retain their high frequency.

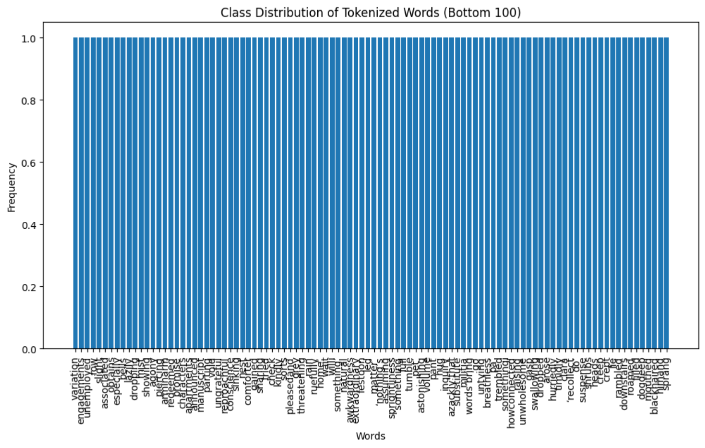

Figure 9: Class Distribution of Tokenized Words (Bottom 100) before Oversampling

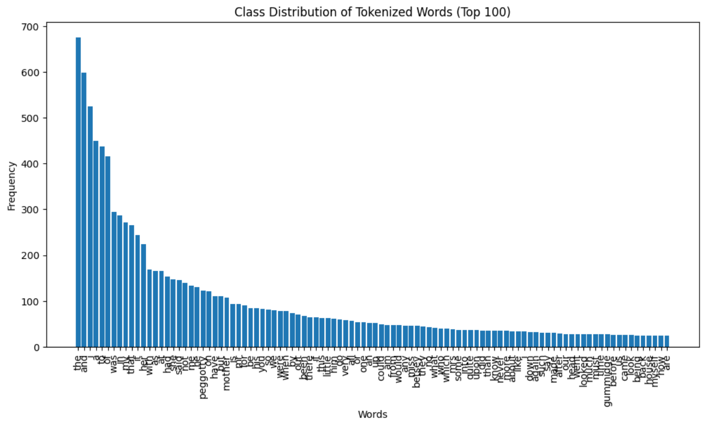

Figure 10: Class Distribution of Tokenized Words (Top 100) before Oversampling

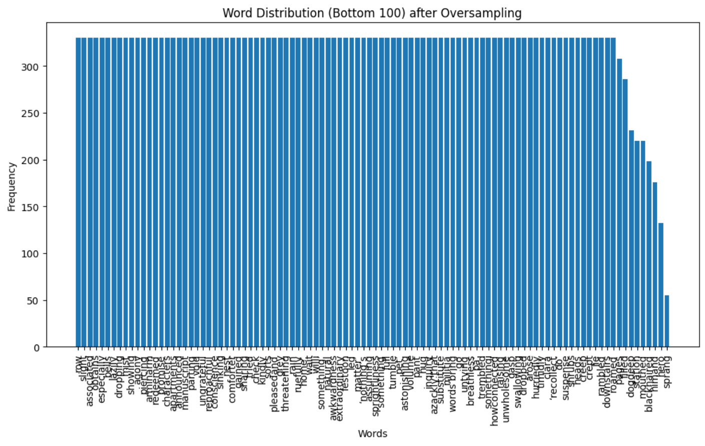

Figure 11: Class Distribution of Tokenized Words (Bottom 100) after Oversampling

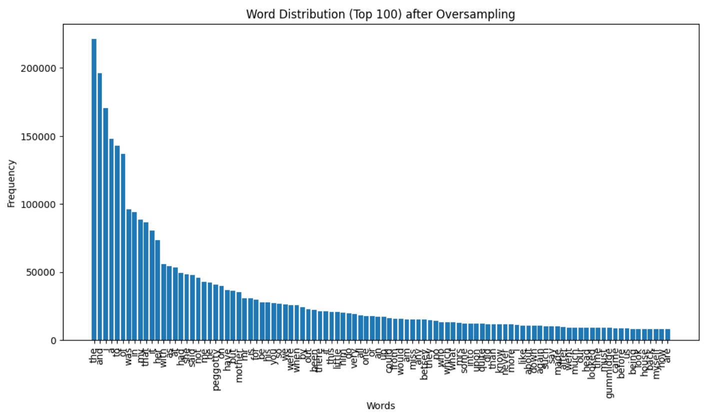

Figure 12: Class Distribution of Tokenized Words (Top 100) after Oversampling

4.7 Discussion

The LSTM model outperforms both the GRU and RNN models significantly in terms of accuracy and loss. the parameter tuning was shown to be successful in adjusting the models performance making it suitable for the language modeling task. The model was able to learn effectively after experimentation with the embedding dimensions, sequence lengths, and data preprocessing techniques.

I believe there to still be some form of data leakage which would need to be avoided for large scale implementation of the model due to how the sliding window technique works for creating sequences. However, A large amount of the under and over fitting was able to be tuned out and the proper data pre-processing eliminated a lot of noise and improved the quality of the model.

5 Conclusion

The project to develop a word-level predictive text module aimed at enhancing communication efficiency for ALS patients was successfully completed and 3 ML models were developed to showcase how different implementations could be achieved. Through the implementation of RNN, GRU, and LSTM models, I identified the LSTM as the most effective model for the task.

The LSTM was most effective due to its ability to capture long-term dependencies inherent in natural language and over the parameter adjustment process I was able to address challenges such as the class imbalance and rare word occurrence using techniques such oversampling. The model was therefore equipt to handle the diverse dataset.

Personally, I do not believe the model will generalize well As I am of the opinion that it tends to just ’memorise’ the datasets sentences rather than actually learn. this leads into future work which could explore the inclusion of more extensive and diverse datasets to generalize better and enhance the model’s robustness.

The project demonstrates the critical role of advanced AI ML models in passivate technology and the findings underscore the need for model selection, parameter tuning, and data preprocessing in creating effective solutions.

References

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

Cho, K., Van Merriënboer, B., Gulcehre, Ç., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Chung, J., Gulcehre, Ç., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Table 2: Model Performance Comparison

Model	**Validation Accuracy (%)**	**Validation Loss**	**Final Epoch**
RNN	10.46	5.9023	20
GRU	7.84	5.5803	20
LSTM	98.17	0.1230	20

Table 3: LSTM Model Iterations and Performance Metrics

Iteration	**Embedding Dim**	**Sequence Length**	**Val Accuracy (%)**	**Val Loss**
1	10	12	47.43	4.1333
2	10	12	100.00	5.96×10^{-7}
3	100	30	N/A	N/A
4	30	30	11.34	6.7530

Table 4: Average FPS Comparison Across Models

Model	**Average FPS**	**Time per Sample (ms)**
RNN	2775.26	0.36
GRU	26.95	37.10
LSTM	27.04	36.98