Data Mining Project: Brain-to-Text Report

Team 15

Valentin Lhermitte, Catherine Tao, Emma Prüfer

December 16, 2025

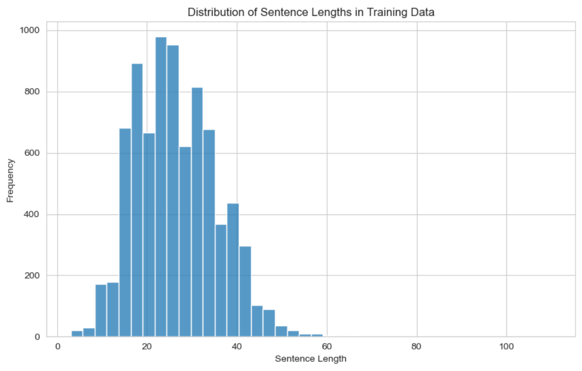
Contents

Chapter 1

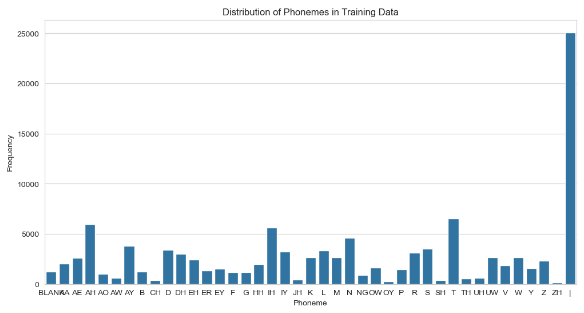
Introduction

People with ALS or brainstem stroke can lose the ability to move and speak. A speech brain-computer interface (BCI) can translate brain activity into text. In this project, our goal is to take in neural activity and decode them into its corresponding English sentence. The data was collected by recording neural spiking activity of 256 microelectrodes in the speech motor cortex. The data set consists of 10,948 sentences from a single participant. The input is a time series, with each timestep has 512 features to describe brain activity. In the dataset, the phoneme that was used is encoded as a numeric ID.

After some initial data exploration, we found that the median sentence length is 26 words (Figure 1.1a) and the phoneme distribution (Figure 1.1b).



(a) The distribution of sentence lengths is fairly even, with most between 2–35 words.



(b) Phoneme distribution shows "I" is the most frequently used phoneme.

Figure 1.1: Data found from initial exploration.

During the process of achieving our decoding goal, we tried various approaches. We began with a combination of CNN and RNN, then moved onto Conformer (a Transformer architecture that integrates convolutional layers), before finally settling for the Transcriber. We also iterated through different algorithms before settling on a greedy beam search approach.

Chapter 2

Methods

In this chapter, we detail the methodologies employed in this project, encompassing data preprocessing, model architecture, training procedures, and evaluation metrics.

2.1 Data Preprocessing and Augmentation

2.1.1 Preprocessing

Data preprocessing is handled dynamically during loading. We apply the following steps:

- **Smoothing:** Optional Gaussian smoothing is applied to the time series with $\sigma = 1$.
- **Normalization:** Input features are Z-score normalized per trial using the mean and standard deviation along the time axis.
- **Clipping:** To mitigate outliers/artifacts, values are clamped to the range $[-5.0, 5.0]$.
- **Label Encoding:** The target transcriptions are mapped to a dense character-level vocabulary (ASCII), with 0 reserved as the `<blank>` token for the RNN-Transducer.

2.1.2 Augmentation

To improve model robustness against non-stationarity and hardware noise, we implement a custom augmentation pipeline applied with specific probabilities during training:

1. **Gaussian Noise:** Additive white noise ($\mathcal{N}(0, 1)$ scaled by 0.1) to simulate thermal noise.
2. **DC Shift:** Random constant offsets added to electrode channels to simulate baseline voltage shifts.
3. **Random Walk Drift:** Cumulative sum of small random steps to simulate gradual impedance drift over time.
4. **Array Dropout:** Randomly zeroing out inputs corresponding to one of the four hardware arrays (indices 0-64, 65-128, etc.) to simulate specific pedestal failures.
5. **Modality Dropout:** Randomly dropping all Threshold Crossings (0-255) or all Spike Band Power features (256-511).
6. **Time Masking:** Zeroing out contiguous blocks of time steps to simulate loss of attention or transient artifacts.

2.2 Strategy

In this Kaggle competition, the authors provide a baseline model that predict phonemes from neural data and translate them to text using a pre-trained language model. However, while this approach is effective, the pre-trained language model requires a large amount of RAM, over 60 GB for a 3-gram model and 200 GB for a 5-gram model, making it impractical for many users (including ourselves). Therefore, we propose an end-to-end model that directly transcribes neural data to text. We tried two different architectures: a CTC-based model [**ctc2006**] and an RNN-Transducer model [3].

The CTC-based model would yield decent results, but only for short sequences, as CTC assumes conditional independence between output tokens.

On the other hand, the RNN-Transducer model is able to model dependencies between output tokens, making it more suitable for this task. We also implement a day-specific adaptation layer to account for session variability in neural recordings, following prior work by *Card et al.* [2]. Finally, we apply a lightweight Large Language Model (Llama-3-8B-Instruct) as a post-processing step to correct minor spelling and grammar mistakes without altering the semantic meaning of the transcriptions.

2.3 Model Architecture

The framework consists of a Transcriber (Encoder), a Predictor, and a Joint Network.

2.3.1 Day-Specific Adaptation

We implement a session-specific linear transformation at the encoder input. For input $\mathbf{x}_t$ and session d, adapted features $\mathbf{h}_0$ are:

$$\mathbf{h}_0 = \text{Softsign}(\mathbf{x}_t \mathbf{W}_d + \mathbf{b}_d) \tag{2.1}$$

where $\mathbf{W}_d$ and $\mathbf{b}_d$ are learnable parameters initialized to identity and zero.

2.3.2 Encoder (Transcriber)

We utilize a **Conformer** backbone to capture local and global dependencies:

- **Projection:** Linear projection with LayerNorm and Dropout maps inputs to hidden dimension H.
- **Backbone:** A stack of Conformer layers (4 heads, depthwise kernel size 31) optimized for neural data temporal resolution.

2.3.3 Predictor and Joint Network

The predictor is a single-layer uni-directional LSTM receiving the embedding of the previous non-blank token.

The Joint Network fuses the encoder output $\mathbf{f}_t$ and predictor output $\mathbf{g}_u$ using an **additive** mechanism. This broadcasts the tensors efficiently to form the logits $z_{t,u}$:

$$z_{t,u} = \mathbf{W}_{out} \cdot \text{GELU}(\mathbf{f}_t + \mathbf{g}_u) + \mathbf{b}_{out} \tag{2.2}$$

where $\mathbf{W}_{out}$ and $\mathbf{b}_{out}$ project to the vocabulary size.

2.3.4 LLM Correction

We employ a local Llama-3-8B-Instruct [1] with temperature $T = 0.1$. It receives phonetically decoded text and corrects minor spelling/grammar errors while strictly preserving semantic meaning.

2.4 Training and Inference

Training: We minimize the Transducer Loss (negative log-likelihood), which marginalizes over all possible alignment paths $\mathbf{a} \in \mathcal{B}^{-1}(\mathbf{y})$:

$$\mathcal{L} = -\ln \sum_{\mathbf{a} \in \mathcal{B}^{-1}(\mathbf{y})} P(\mathbf{a}|\mathbf{x}) \tag{2.3}$$

Inference: We utilize standard greedy decoding. The model processes inputs online; if the non-blank token has the highest probability, it is emitted and fed back to the predictor; otherwise, the encoder advances to the next time step.

Evaluation: Performance is measured using the Word Error Rate (WER), calculated as the sum of substitutions, deletions, and insertions divided by the total reference words.

Chapter 3

Results

The primary challenge of this brain-to-text project was the development of a robust and memory-efficient end-to-end model to directly translate neural signals into text. The initial experiments with a Connectionist Temporal Classification (CTC)-based architecture served as an important comparison point for the final RNN-Transducer (RNN-T) model. The numerical results of the Word Error Rate (WER) confirm the superiority of the RNN-T approach.

Model Architecture	**Word Error Rate**	**Performance Evaluation**
CTC-based Model	1.1245	Fails with long sequences
RNN Transducer	0.367	Significant improvement; can handle long sequences

Table 3.1: Comparison of Model Architectures and Their Performance

The discrepancy in WER is due to the difference in the way the models handle the temporal dependencies of the output:

3.0.1 Failure of CTC Model

The CTC model assumes that the probability of emitting a token depends only on the corresponding encoder output—and is conditionally independent of the previously emitted tokens. For short sequences, this assumption is often less severe, but for longer and linguistically more complex utterances, the model fails. The high WER of 1.1245, mainly triggered by excessive insertions, confirms that the model was unable to generate plausible text sequences without modeling linguistic context.

3.0.2 Success of the RNN-Transducer Model

The RNN-T model overcomes this bottleneck by using a predictor to explicitly model output dependencies. The RNN-T uses a Conformer encoder to process the neural input data and an LSTM predictor network to generate the next token based on the previous token. The Joint Network fuses these two streams of information. RNNTLoss optimizes the probability of the entire sequence, with the predictor ensuring linguistic plausibility. This leads to a drastic reduction in errors as the model is now able to form coherent words and sentences. The result underscores the central insight: modeling sequential linguistic dependencies is essential for brain-to-text transcription.

Chapter 4

Discussion of the Results

The performance of the model was measured using a common metric in the field of speech recognition, the Word Error Rate (WER). The evaluation was carried out on the validation dataset using greedy decoding and beam search decoding.

4.0.1 Word Error Rate

The Word Error Rate (WER) is the primary metric for evaluating the quality of transcriptions in speech recognition and serves as the main performance indicator of our brain-to-text model. It measures the minimal effort in terms of words required to convert the predicted transcription (hypothesis) into the correct transcription (reference). The WER is formally calculated as follows:

$$\text{WER} = \frac{\text{Substitutions} + \text{Deletions} + \text{Insertions}}{\text{Amount of words in reference}}$$

The following explains each term in more detail:
Substitutions: A word in the reference is replaced by an incorrect word in the hypothesis (e.g., "The house is red" → "The mouse is red").
Deletions: A word in the reference is missing in the hypothesis (e.g., "The house is red" → "The is red").
Insertions: A word in the hypothesis is not present in the reference (e.g., "The house is red" → "The house is small red").
Normalization factor: The total number of words in the reference. A lower WER value indicates better model performance.

4.0.2 Error Patterns

The errors occurring within the RNN-Transducer model can typically be traced back to the following word categories and patterns:

Function Words

Function words are often short, have low semantic content and therefore show weaker and less specific neural activation. Examples: articles ("the", "a"), prepositions ("in," "at," "on"), short conjunctions ("and," "or"). The model easily confuses these words with each other (e.g., substitution of "a" with "the") or tends to omit them (deletion), as they appear less relevant for decoding the main meaning.

Rare Words and Proper Nouns

Combining characters into a rare word presents a challenge. Words that occur rarely or not at all in the training data, such as specific proper nouns or technical terms, are difficult to transcribe because the predictor network lacks the necessary linguistic context to enforce the correct sequence of characters.

Phonetic Similarity and Homophones

The transcription is based on the underlying neural activity that reflects the articulation of phonemes. Examples: words that sound similar ("nice" vs. "mice"). Slight or overlapping neural activation patterns for phonetically similar sounds lead to substitutions. LLM correction is often able to fix these errors when the corrected sentence becomes grammatically more coherent.

Contribution breakdown

We had 3 members in our group. After discussing together, we agreed that Valentin Lhermitte did x work, Emma Prüfer did x word, and Catherine Tao did x work. This is the breakdown of the contribution

Name	**Student ID**	**Breakdown (%)**	**Tasks Completed**
Valentin Lhermitte	X1140067	XX%	Preprocessing data; experimented with CNN and RNN; implemented Conformer; developed Transcriber.
Emma Prüfer	X1140041	XX%	Reviewed research papers; created progress reports; experimented with CNN and RNN.
Catherine Tao	X1140008	XX%	Reviewed research papers; improved the baseline model; explored beam search; created progress reports; experimented with CNN and RNN.

Table 4.1: Team Contributions

Bibliography

[1] AI@Meta. "Llama 3 Model Card". In: (2024). URL: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

[2] Nicholas S. Card et al. "An Accurate and Rapidly Calibrating Speech Neuroprosthesis". In: *New England Journal of Medicine* 391.7 (2024), pp. 609–618. DOI: 10.1056/NEJMoa2314132. eprint: https://www.nejm.org/doi/pdf/10.1056/NEJMoa2314132. URL: https://www.nejm.org/doi/full/10.1056/NEJMoa2314132.

[3] Alex Graves. "Sequence Transduction with Recurrent Neural Networks". In: *CoRR* abs/1211.3711 (2012). arXiv: 1211.3711. URL: http://arxiv.org/abs/1211.3711.