Lab 3.2, Stat 214, Spring 2025

April 29, 2025

1 Introduction

Understanding how the brain processes natural language remains a major challenge in cognitive neuroscience. Recent advances in language modeling, particularly contextual embedding models like BERT, offer new opportunities for mapping linguistic information to brain activity. Voxelwise encoding models, which predict fMRI signals from stimulus features, provide a framework to test how well different embeddings capture the structure of neural responses during language comprehension.

Our previous analysis using static embeddings such as Word2Vec and GloVe captured some semantic information. Unlike static models, contextual models like BERT dynamically adjust word representations based on surrounding words, potentially aligning more closely with brain processes. In this report, we aim to explore whether a more dynamic and context-dependent embedding process can capture more meaningful information and result in stronger predictions in voxel outputs.

2 Methods

Similar to Lab 3.1, we leveraged a ridge regression model with bootstrapped cross-validation to model each voxel's response as a function of context-based embedding features initialized by BERT. The bootstrap ridge regression approach was selected for its robustness to noisy data, which was particularly important given the nature of fMRI voxel responses and the high-dimensional BERT embeddings.

Bootstrapping provides more reliable estimates of standard errors and enables effective feature selection by evaluating the variance of regression coefficients across different bootstrap samples, thus supporting optimal model regularization and reducing overfitting.

Model performance was evaluated by computing the Pearson correlation coefficient (CC) between predicted and actual voxel responses on a held-out test set. To assess the reliability of voxel prediction, we computed a stability score by correlating voxelwise CCs obtained independently from two halves of the test dataset.

2.1 Pre-training

2.1.1 Encoder Model

The first step in the implemented design lies in constructing the input representation. Raw text tokens are transformed into dense vectors through the summation of three distinct embedding types, all constrained to the same dimensionality. This summation strategy aims to create a rich, multi-faceted initial representation. Firstly, token embeddings provide the basic lexical-semantic meaning. The chosen dimensionality reflects a critical trade-off: higher dimensions might capture finer semantic distinctions but increase model size and computational demands, potentially risking overfitting, whereas lower dimensions might underfit.

Secondly, positional embeddings are incorporated. Given that the core self-attention mechanism is permutation-invariant, explicit positional information is essential. The decision implemented here, to use learned positional embeddings rather than fixed sinusoidal ones, was made to offer greater flexibility. This allows the model to adapt positional representations to the temporal patterns and linguistic structures in the podcast data, potentially leading to more tailored and effective representations than fixed alternatives.

Thirdly, token-type embeddings are included, primarily for architectural consistency with standard Transformer implementations like BERT. While less functionally critical for single continuous narratives, their inclusion incurs minimal overhead and maintains compatibility. The resulting summed vector thus encodes lexical identity, learned sequential position, and segment membership, providing a comprehensive input.

Next, the computational engine comprises a stack of identical Transformer blocks. Stacking multiple layers enables the learning of hierarchical representations, potentially mirroring hierarchical linguistic processing in the brain. Each block contains two primary sub-layers, critically employing residual connections and layer normalization. These specific choices are essential design decisions, vital for effective training. Residual connections directly facilitate gradient flow, mitigating vanishing gradients and enabling the training of deeper models. Layer normalization, applied before the sub-layer input in standard pre-LN Transformer variants or after, as implemented here, stabilizes activations across the feature dimension for each instance independently, improving optimization dynamics and reducing sensitivity to parameter initialization, contributing to faster and more reliable convergence towards a useful model.

The first sub-layer implements multi-head self-attention. Projecting the input into distinct Query, Key, and Value spaces provides the necessary flexibility. The specific formulation of scaled dot-product attention, including the scaling factor, was chosen as it is crucial for preventing excessively large dot products and stabilizing training. Adopting a multi-head approach is a key design decision, allowing the model to simultaneously attend to different types of information from different representational perspectives, yielding a richer overall representation. An attention mask is applied to prevent attention to padding tokens, ensuring correct handling of variable-length sequences.

The second sub-layer is a position-wise feed-forward network (FFN). Its inclusion significantly increases the model's capacity to learn complex, non-linear transformations. The implemented structure involves two linear transformations with an intermediate non-linear activation. The specific choice of GELU as the activation function aligns with models like BERT. GELU's smooth, non-monotonic nature, compared to the simpler ReLU, has been empirically shown to often lead to better performance in Transformer models. This choice is hypothesized to facilitate the learning of more complex functions within the FFN, potentially contributing to richer final embeddings. The dimensionality expansion in the intermediate layer allows the network to project the representation into a higher-dimensional space where more complex patterns might be learned before projecting back.

Finally, for the specific objective of pre-training, the output from the final Transformer block undergoes one last layer normalization before being passed to a dedicated linear output layer. This head projects each token's final contextualized hidden state into a score distribution across the entire vocabulary. Its sole purpose during this phase is to produce the logits necessary for the MLM prediction task. This head is task-specific and would be discarded when using the encoder for downstream feature extraction, where the final hidden states themselves are the desired output.

2.1.2 Masked-language Model Training

To help the Encoder model with a nuanced understanding of linguistic structure and context derived directly from the podcast narratives, a pre-training phase utilizing the Masked Language Model (MLM) objective was implemented. The rationale for selecting MLM is its inherent ability to foster bidirectional contextual representations. By training the model to predict masked tokens based on both preceding and succeeding contexts, MLM encourages the learning of representations that encapsulate a more complete understanding of a token's meaning within its full utterance. This bidirectional context is valuable for modeling brain responses during natural language comprehension.

MLM's operational core involves the masking strategy implemented in the `mask_tokens` function. Adhering to the empirically validated practices established by BERT, approximately 15% of the input tokens (excluding padding) were randomly selected for prediction. This 15% probability represents a trade-off. The specific treatment of these selected tokens followed the widely adopted 80/10/10 heuristic, implemented via Bernoulli sampling:

- **80% replaced with `MASK`:** This explicitly forces prediction from context.
- **10% replaced with a random token:** This design choice introduces noise, compelling the model to rely more heavily on the surrounding context rather than potentially trusting a corrupted input token, thereby enhancing robustness.
- **10% unchanged:** This subtle but important element reduces the discrepancy between pre-training and downstream use, ensuring the model learns effective representations even for unmasked words.

Concurrently, labels were generated, storing the original token IDs only for the masked positions and setting

others to -100. This precise label construction ensures that the loss computation exclusively targets the intended prediction task.

The training procedure within train_bert was designed to refine the model's parameters iteratively. Processing data in batches was chosen for computational efficiency. Another design decision reflected in the code is the implementation of dynamic masking, where mask_tokens are called within the training loop for each batch. This prevents the model from merely memorizing static masked positions and encourages a more generalized contextual understanding. Within each training step, a batch underwent this dynamic masking. The masked sequences were processed by the Encoder to yield output logits. The Cross-Entropy loss function, standard for multi-class classification, was employed. The specific configuration to ignore the index -100 was crucial, focusing the learning signal entirely on the MLM task by excluding non-masked tokens from the loss calculation.

Parameter optimization was performed using the AdamW algorithm. The conscious choice of AdamW over standard Adam was motivated by its improved handling of weight decay regularization, often leading to better generalization performance on downstream tasks. Achieving good generalization during pre-training is highly desirable here, as the goal is to produce embeddings that are broadly useful for the subsequent fMRI prediction task on potentially unseen data. The learning rate (lr) was identified as a critical hyperparameter requiring careful tuning. This entire cycle was repeated across the dataset for a specified number of epochs.

Consistent with model development, systematic hyperparameter tuning was planned. Close monitoring of training and validation loss curves during this phase is key for identifying a configuration that promotes effective learning without substantial overfitting.

2.1.3 Hyperparameter Tuning and Loss Analysis

We trained the masked language model (MLM) using our custom encoder on a dataset of sentences. To evaluate how different model settings affect performance, we performed a series of experiments with varying hyperparameters.

Specifically, we tested configurations by changing the number of transformer layers (num_layers), the hidden size (hidden_size), and the learning rate (lr). After several tries, parts of the best configurations are listed below:

- lr=5e-4, num_layers=2, hidden_size=128
- lr=5e-4, num_layers=4, hidden_size=256
- lr=1e-4, num_layers=6, hidden_size=512

For each configuration, we trained the model for 35 epochs and recorded both training and validation loss at each epoch. (The epoch number is also tuned based on the loss analysis). The training was performed using a fixed batch size and a constant number of attention heads (num_heads=4). The intermediate size in the feedforward layers was always set to twice the hidden size.

Two loss plots listed below show the training and validation loss curves for each configuration, which illustrate the convergence behavior and generalization performance of the models. Based on the two configurations, we can conclude that 35 epochs have been enough for training, as training loss begin to converge above 6 and validation loss coverge around 7.

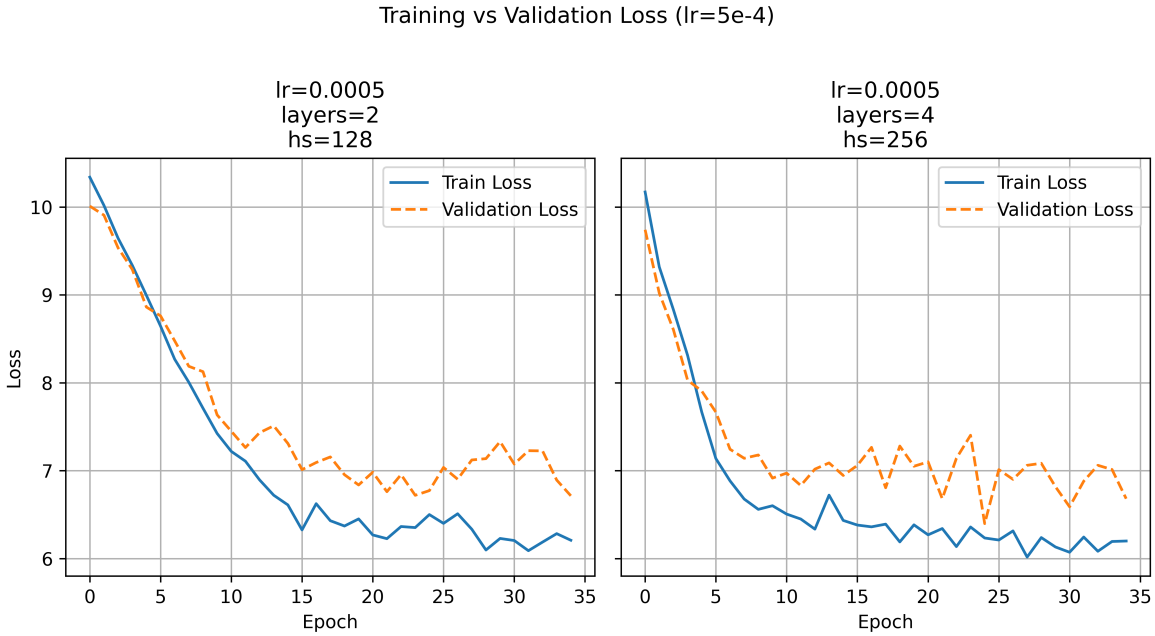

Figure 1: Loss Curve for the Best-performing Configuration,`lr=1e-4`.

We can observe that models with larger hidden sizes and more layers generally performed better on the validation set, especially when trained with a smaller learning rate (`1e-4`). Among the three, the best validation performance was achieved by the model with 6 layers, a hidden size of 512, and `lr=1e-4`. The validation loss for this model setting is around 6.5-7, while the training loss is around 6-6.5.

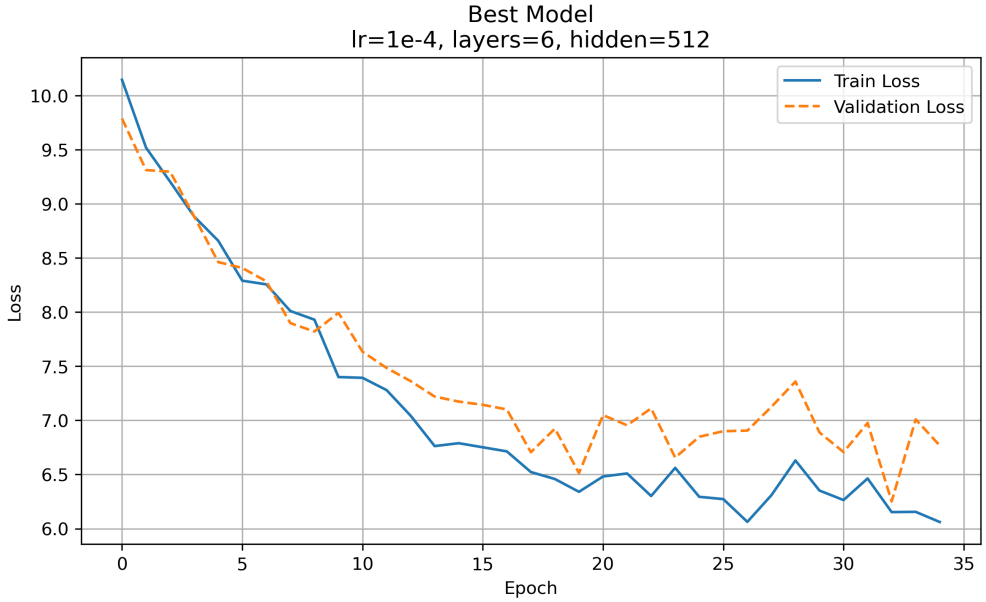

Figure 2: Loss Curve for the Best configuration

This outcome suggests that deeper models benefit from greater representational capacity, as long as they are trained with a relatively small learning rate that allows stable convergence. In part 2, we will focus on this

best configuration. We will generate corresponding embeddings for the modeling and evaluation part.

3 Modeling & Evaluation

3.1 Embedding Generation

In order to generate token-level embeddings for each story, we employed a custom-trained BERT-style encoder model. Below, we describe the full embedding generation pipeline.

We used the `bert-base-uncased` tokenizer to tokenize the story text. The encoder architecture consisted of 6 transformer layers, with a hidden size of 512, 4 attention heads, and an intermediate size of 1024. The maximum sequence length was set to 128. The model was pre-trained using a masked language modeling objective, and we used the checkpoint `encoder_lr0.0001_layers6_hs512.pt` for final embedding extraction.

To extract embeddings, we processed each story word sequence using a sliding window strategy. Each window (or chunk) contained up to 128 words, and consecutive chunks overlapped with a stride of 64 words. Each chunk was tokenized with padding and truncation to length 128, then passed through the encoder to obtain token-level hidden states. These hidden states were mapped back to word-level using the tokenizer's `word_ids()` mapping, and all token representations corresponding to the same word were averaged to form a single word embedding. In cases where a word appeared across multiple overlapping chunks, its embeddings from different chunks were averaged. If a word failed to obtain any valid embedding (e.g., due to truncation), it was assigned a zero vector of size 512.

After obtaining word-level embeddings, we interpolated them to match fMRI TR-level timestamps using Lanczos interpolation. To remain consistent with the preprocessing pipeline, we trimmed the first 5 seconds and last 10 seconds from each story. We then generated delayed features by concatenating the original features with versions shifted by 1, 2, 3, and 4 TRs. The resulting matrices were saved as `.npy` files, one per story.

This pipeline ensured that the extracted embeddings were temporally aligned with fMRI acquisition and ready for downstream regression modeling.

3.2 Results

3.2.1 Performance Across Voxels

Figure (input) depicts the ridge regression results from BERT embeddings as follows: the mean correlation coefficient (CC) was 1.299%, the median CC was 1.246%, the top 5% CC was 3.947%, and the top 1% CC was 5.515%.

The distribution of voxelwise correlation coefficients was heavily skewed toward zero, indicating that most voxels were poorly predicted by the model, with only a small subset achieving higher correlations.

Despite the low average correlation coefficients across voxels, the presence of a positively skewed distribution and a subset of voxels with higher CCs indicates that the model was able to capture meaningful brain bloodflow and semantic relationships rather than fitting pure noise.

For stability check, we plotted both halves of the testing data in Figure (palceholder) to evaluate the correlation of voxels. The correlation coefficient was 38.65%, indicating moderate stability for our current regression model. Given the noisiness of fMRI and semantic, a stability score of approximately 0.39 is consistent with expectations for voxelwise encoding models.

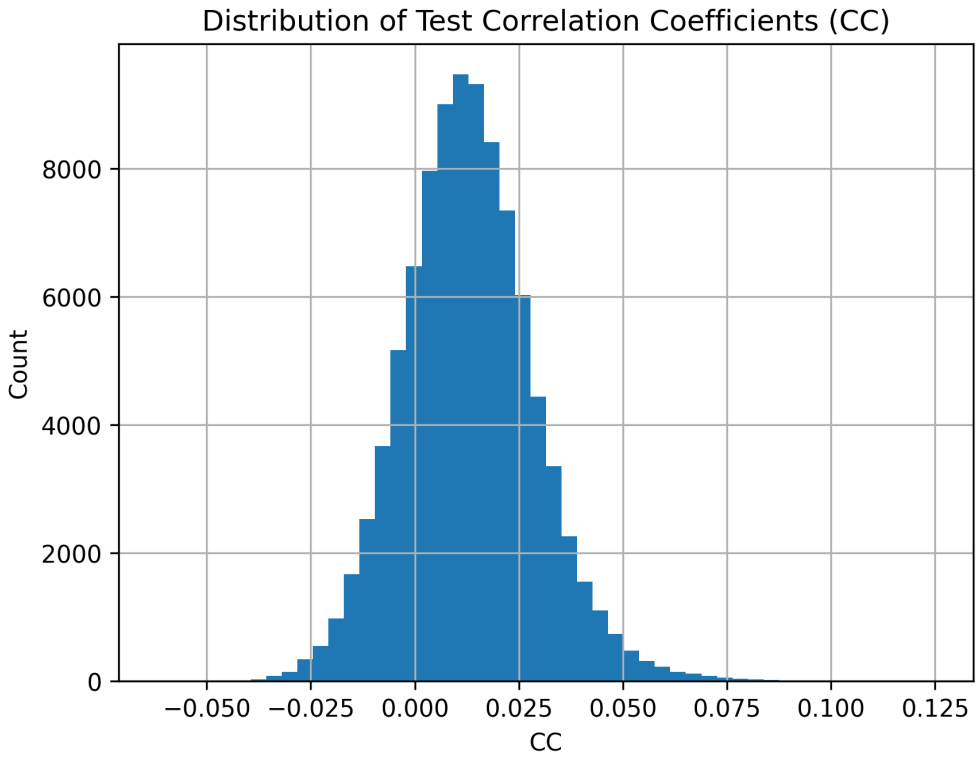

Figure 3: Distribution of Test Correlation Coefficients (CC)

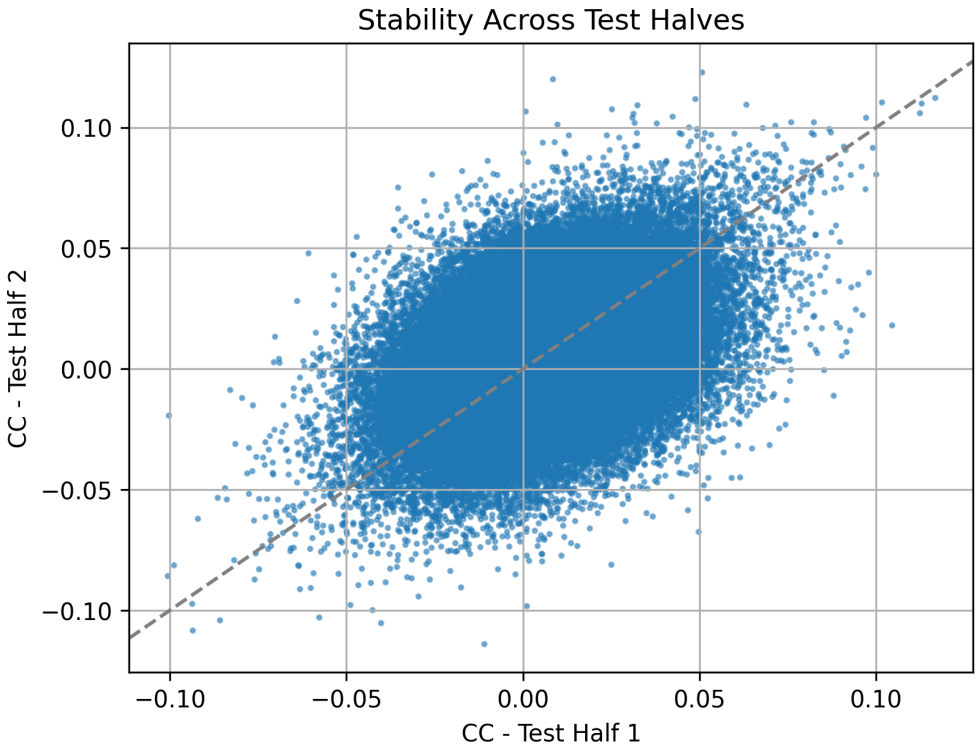

Figure 4: Stability Check: Voxel Correlation

To achieve further robustness in our ridge regression modeling, we increased the number of bootstrap samples to 20 and used a chunk length of 20 during cross-validation.

Table 1 presents the updated voxelwise correlation coefficient (CC) results. These results show a modest improvement compared to our previous analysis, following a similar distribution pattern: the majority of voxels exhibit low predictive performance, while a smaller subset of voxels achieves higher CCs. The smaller voxels that achieve higher CC are indicative meaningful brain blood flow and story semantic relationships because they show that some signal has been captured during the fMRI scans and that it wasn't all noise.

Table 1: Updated Ridge Regression Performance Metrics (BERT Embeddings)

Metric	**Value**
Mean CC	1.356%
Median CC	1.267%
Top 5% CC	4.137%
Top 1% CC	6.098%

The stability correlation score, however, decreased to about 34.15% following our increase in bootstrap samples and chunk lengths. This slight decrease compared to our earlier score of approximately 38.65% reflects our stricter cross-validation procedure and improvement in the generalizability of our model. Such trade-offs are expected when adopting more stringent validation protocols.

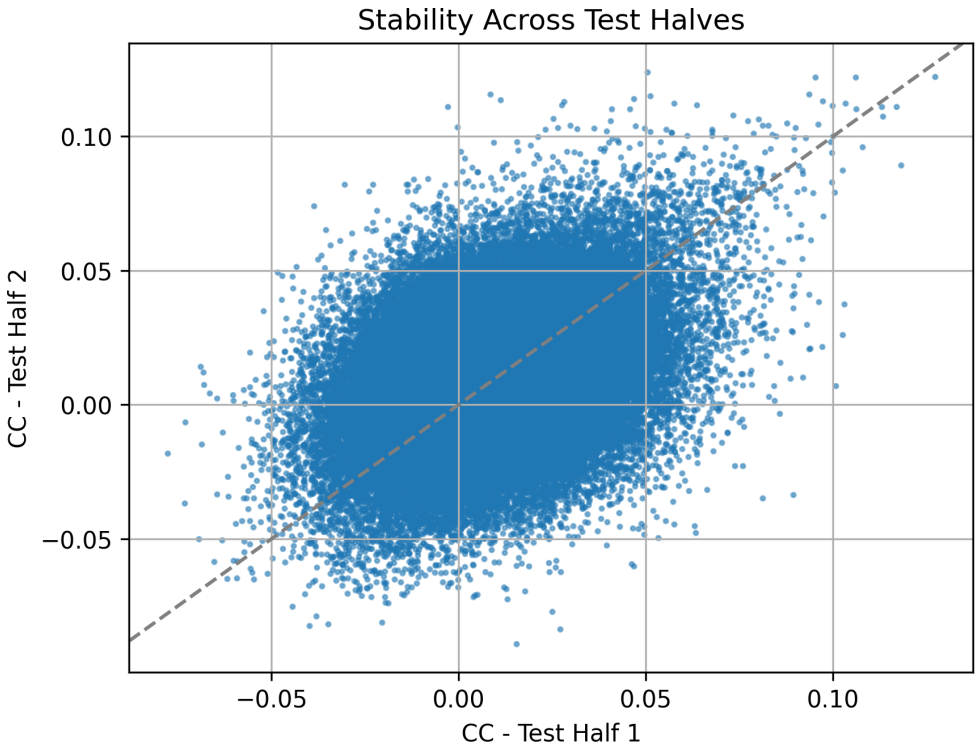

Figure 5: Stability Check 2: Voxel Correlation w/ Stringent Bootstrapping Protocol

3.2.2 Comparing Embedding Results

We compared the voxelwise performance of our BERT-based embedding model to embedding models from our previous lab. The embedding methods conducted in the previous lab were Bag-of-Words (BoW), Word2Vec, GloVe. Table 2 summarizes the stability scores across methods.

Embedding Type	**Stability (Correlation)**
BERT1	0.3865
BERT2 - Stricter Bootstrap	0.3415
BoW	0.0123
Word2Vec	0.2782
GloVe	0.2531

Table 2: Stability (correlation of voxel CCs across halves) for different embeddings.

The results show that contextual embeddings from BERT substantially outperformed the non-contextual embeddings (BoW, Word2Vec, and GloVe) in terms of stability across voxels. BoW embeddings performed the worst, with near-zero stability, suggesting minimal predictive power. Both Word2Vec and GloVe achieved moderate stability. These results indicate that some static semantic information can predict brain responses, but not as reliably as contextualized embeddings that BERT produces.

Among the BERT-based models, the version using stricter bootstrap parameters (BERT2) showed a slight reduction in stability compared to the original (BERT1), reflecting the expected trade-off between robustness and predictive consistency when applying more rigorous cross-validation. Based on these results, we can conclude that different embedding methods do not perform equally well across voxels: contextual models like BERT provide superior voxelwise prediction stability compared to simpler or static embeddings like Glove or BoW.

4 Discussion and Conclusion

This report demonstrates that BERT-based contextual embeddings outperform static embeddings like BoW, Word2Vec, and GloVe for voxelwise prediction of fMRI responses. Although the average correlation coefficients were low, a subset of voxels consistently achieved higher predictive performance, confirming that meaningful signal—not just noise—was captured.

Stricter cross-validation with increased bootstraps and chunk length led to slightly improved CCs and more robust model selection, at the cost of a modest decrease in stability. This trade-off reflects better generalization rather than overfitting. Our findings are consistent with prior work showing that context-sensitive representations more closely align with brain activity during language comprehension.

Future work can include plotting only top 5% or 1% of voxels, which captured better signal than most previous voxels and can yield more accurate predictive results. In short, contextual embeddings such as BERT offer a clear advantage for brain encoding models, and future work could refine these approaches by targeting specific brain regions or extending context modeling.

5 Bibliography

Shailee Jain and Alexander G. Huth, *Incorporating Context into Language Encoding Models for fMRI*, Departments of Computer Science and Neuroscience, The University of Texas at Austin, 2020. Available: `https://papers.nips.cc/paper_files/paper/2018/hash/f471223d1a1614b58a7dc45c9d01df19-Abstract.html`

Francisco Pereira, Matthew Botvinick, and Samuel R. Gershman, *Toward a universal decoder of linguistic meaning from brain activation*, Nature Communications, vol. 9, 2018. Available: `https://www.nature.com/articles/s41467-018-03068-4`

A Academic honesty

A.1 Statement

We make the academic integrity pledge here: All our work in this report are done independently. All sources we used are properly cited, whether from LLMs, papers, textbooks, or classmates.

Academic research honesty is necessary in the academy and also the foundation of our society. It guarantees fairness in research, stimulates research's activeness, and exerts a positive impact on the progress of science and technology. We should always adhere to the rules, which will create a more regulated and benign world.

A.2 LLM Usage

Coding

We use GitHub Copilot for several parts of debugging. We additionally used it to help with making graphs and signatures for the functions in the notebooks.

Writing

Grammarly was used to check for any spelling issues.