Lab 3.3 - FMRI, Stat 214, Spring 2025

May 16, 2025

1 Introduction

In this lab, we extend our previous work on predicting BOLD responses for voxels using pre-trained embeddings and transformer models by incorporating advanced fine-tuning and interpretability techniques. First, we utilize a pre-trained BERT model and fine-tune it on our dataset to enhance the predictive performance across voxels. To achieve parameter-efficient fine-tuning, we implement Low-Rank Adaptation (LoRA), allowing us to update only a small subset of model parameters while maintaining computational efficiency.

Next, we focus on model interpretability by examining the word-level contributions to voxel predictions using SHAP and LIME, identifying the influential words associated with well-performing voxels and assessing how these words differ across voxels and stories. Through this analysis, we aim to uncover patterns in the learned embeddings that align with neural responses and to explore how the fine-tuning process alters these patterns.

2 Pre-trained BERT and Fine-tuning

2.1 Model fitting with BERT-Base Uncased

The BERT-Base Uncased model, developed by Google, is a transformer-based model pre-trained on large-scale text data from the English Wikipedia and BooksCorpus datasets [1]. BERT (Bidirectional Encoder Representations from Transformers) leverages a bidirectional training approach, allowing it to effectively capture contextual information from both preceding and succeeding words in a sentence. Unlike traditional language models that process text sequentially, BERT simultaneously considers both left and right contexts, making it highly effective for generating rich, contextual embeddings.

The model consists of 12 transformer layers with 768 hidden units and 12 self-attention heads, resulting in approximately 110 million parameters. The "uncased" version of BERT does not differentiate between uppercase and lowercase letters, treating "apple" and "Apple" as the same token. This characteristic helps reduce vocabulary size and computational complexity.

We utilized BERT-Base Uncased to extract embeddings for each word in our training and testing stories. Unlike in Lab 3.2, where we trained a model using the 70 training stories, we expect the BERT embeddings here to perform better since they are derived from a model pre-trained on a vast corpus of text data.

2.2 Embedding Extraction using Pre-trained BERT and Ridge Regression

To extract embeddings for each word in the training and test stories, we developed a custom function, `make_word_embeddings`, implemented in `embeddings.py`. This function leverages the pre-trained `bert-base-uncased` model to generate contextualized word embeddings, maintaining the original word order of each story.

The objective of maintaining separate embeddings for each story is to preserve the contextual nuances captured by BERT, allowing the same word to have different vector representations depending on its surrounding context.

Since the BERT model has a maximum input length of 512 tokens, and the tokenizer often splits words into subwords (e.g., *transformer* becomes *transform* and *##er*), we set the chunk size to 400 words. This ensures that even after tokenization, the total length remains within the 512-token limit, preventing potential truncation and loss of information. For longer stories containing more than 400 words, the text is divided into multiple chunks, and the resulting embeddings are concatenated at the end to form a complete embedding matrix for the story.

A key aspect of the implementation involves handling subwords and contractions to maintain a consistent number of embeddings per word in the original word list. For instance, a word such as *Adoll's* may be tokenized into the subwords *Adoll*, *´*, and *s*. Similarly, a word like *transformer* may be divided into *transform* and *##er*.

To address the issue of subword tokenization, we aggregate the embeddings of subwords by averaging them, thereby producing a single vector representation for each word. This approach ensures that even when words are split into multiple subwords, the embedding matrix maintains the same number of rows as the original word list. This consistency is crucial for subsequent alignment with the voxel response matrices, which have a fixed number of rows per story.

After generating the 768-dimensional vector for each word in each story, we proceed with the same preprocessing steps as in previous labs. We downsample each embedding matrix using the provided `downsample_word_vectors` function to align the temporal dimensions with the response matrices and apply `make_delayed` to account for temporal lags, resulting in a final feature matrix of size $768 \times 4 = 3{,}072$ columns for each story.

The ridge regression fitting process follows the procedure established in Labs 3.1 and 3.2, where we utilize 5-fold cross-validation to determine the optimal ridge regularization parameter (α) for each voxel independently.

As shown in Table 1, the CCs derived from the pre-trained BERT model exhibit significant improvements over previous methods, including Word2Vec, GloVe, and the BERT model trained in Lab 3.2. This highlights the efficacy of the pre-trained BERT model, which benefits from extensive training on large-scale datasets, allowing it to generate more contextually rich and informative embeddings.

2.3 Fine-tuning using LORA

To adapt a pre-trained BERT model for the downstream task, we employed a parameter-efficient fine-tuning approach using Low-Rank Adaptation (LoRA) with Masked Language Modeling (MLM) as the training objective. Masked Language Modeling(MLM) is a self-supervised learning used to pre-train transformer models. We convert and tokenize the given stories using `bert_base_uncased`, and for each tokenized sequence, we will randomly mask 15% of the tokens by replacing them with [MASK] token. The model will predict the original token IDs as labels and learn to capture the semantic meaning of stories better.

We fine-tuned with LoRA, which is a parameter-efficient fine-tuning technique. Instead of updating the entire weight matrix of the transformer layer, LoRA freezes the pre-trained weights and will only update two low-rank matrices. In our task, we applied LoRA to Bert's key, query, and value layers. And we conducted a grid search over four critical hyperparameters: learning rate, batch size, LoRA rank, and LoRA alpha. The learning rate decides the step size of gradient updates during optimization, controlling how quickly the LoRA parameters adapt to the MLM task. The batch size specifies the number of training samples processed before updating the model's parameters. The LoRA rank determines the size of the low-rank matrices, which will control the number of trainable parameters in the LoRA update. LoRA alpha is a scaling factor that adjusts the magnitude of adaptation relative to the pre-trained weights. Higher values amplify the update's impact, while lower values maintain subtler changes.

2.4 Performance

We used the fine-tuned model to generate embeddings and fit ridge regression as before. The result is shown below in Table 1.

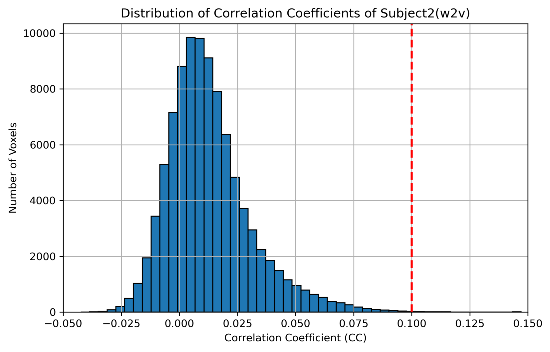
Embedding Method	**Mean CC**	**Median CC**	**Top 1% CC**	**Top 5% CC**
		Subject 2		
Bag of Words	0.0042060038	0.0037176975	0.0364758933	0.0244398914
Word2Vec (w2v)	0.0129826982	0.0102065939	0.0710272557	0.0478154413
GloVe (glv)	0.0136914078	0.0109419302	0.0708861335	0.0482397245
Pre-trained Encoder(lab3.2)	0.0049736367	0.0045462046	0.0355192952	0.0251747680
BERT base uncased	0.0245770679	0.0191937620	0.1093268256	0.0748373350
LoRA Fine-tuned BERT	**0.0245770680**	**0.0191937621**	**0.1093268280**	**0.0748373351**
		Subject 3		
Bag of Words	0.0049011371	0.0039830452	0.0450822272	0.0270554954
Word2Vec (w2v)	0.0208148823	0.0168553494	0.0945664687	0.0634634620
GloVe (glv)	0.0191659305	0.0156664018	0.0832613235	0.0582989910
Pre-trained Encoder(lab3.2)	0.0081298207	0.0067826482	0.0511691383	0.0332598499
BERT base uncased	0.0332981187	0.0263395620	0.1372150348	0.0945452686
LoRA Fine-tuned BERT	**0.0332981191**	**0.0263395640**	**0.1372150355**	**0.0945452698**

Table 1: Comparison of Test Correlation Coefficients (CC) for Subject 2 and Subject 3 Across Embedding Methods

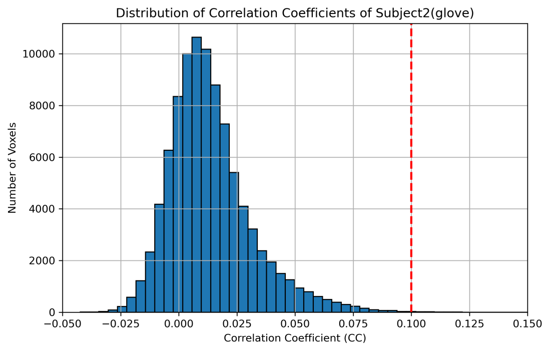
Both `bert-base-uncased` and LoRA Fine-tuned BERT significantly outperform other embedding methods (Bag of Words, Word2Vec, GloVe, Pre-trained Encoder) across all metrics for both subjects, indicating BERT's superior ability to capture contextual representations. However, comparing `bert-base-uncased` and LoRA Fine-tuned BERT, we found our fine-tuning only generates an extremely small improvement, on the order of 10^{-9}, and such a difference is practically negligible.

The minimal performance gains can be attributed to several factors. First, our training dataset only consists of 70 stories, which is relatively small for fine-tuning a large model like Bert. Such a small dataset constrains the model's ability to learn meaningful semantic refinements, which reduces the impact of fine-tuning. Second, if the linguistic characteristics of our training stories are similar to those in the corpus used to pre-train `bert-base-uncased`, then further MLM approach may yield minimal benefits. Finally, our fine-tuning procedure involved only MLM and did not incorporate voxel-level supervision. As a result, the updated embeddings may better reflect the linguistic characteristics of the training texts, but are not explicitly optimized to improve voxel prediction accuracy.

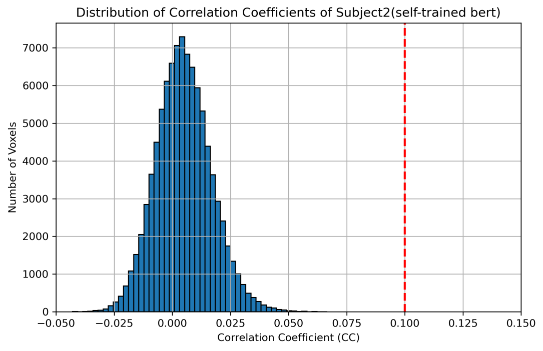
Below, we present the CC distributions across voxels for all embedding methods used in this study, as shown in Figure 1 for Subject 2 and Figure 2 for Subject 3. The Bag-of-Words distribution is excluded from the visualizations, as it consistently performed the worst among all methods. A notable observation is that the distributions for Subject 3 are more positively skewed compared to those for Subject 2, indicating higher CC values across embeddings. For both subjects, models (d) and (e), which are derived from the pre-trained and fine-tuned BERT models, exhibit the highest CC values, outperforming the other embeddings. Specifically, for Subject 3, a larger proportion of CCs exceed 0.1, highlighting a significant improvement over previous embedding methods and demonstrating the effectiveness of the pre-trained BERT models in capturing neural response patterns.



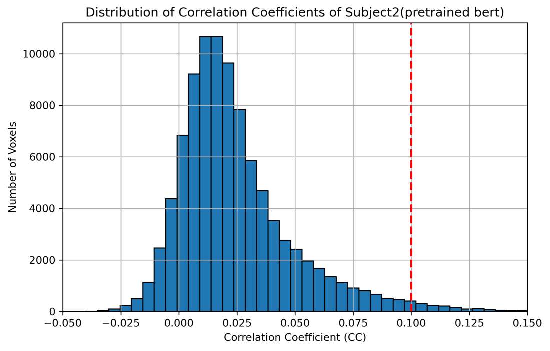
(a) Word2Vec



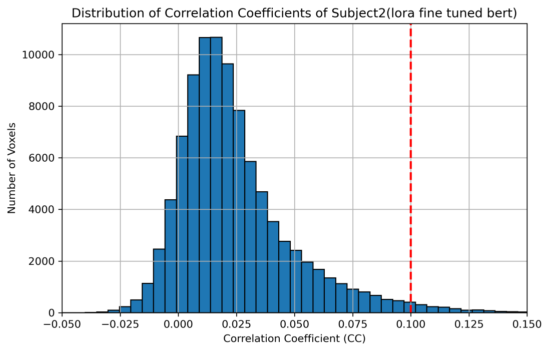
(b) GloVe



(c) BERT (Lab 3.2)

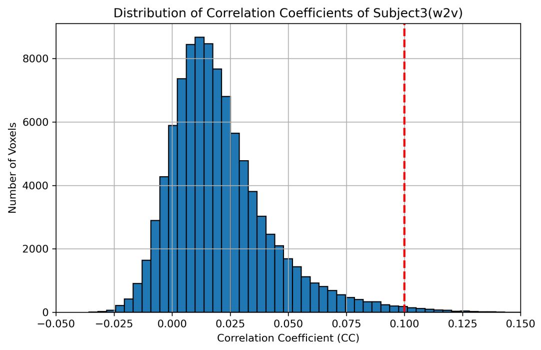


(d) Pretrained BERT-Base-Uncased

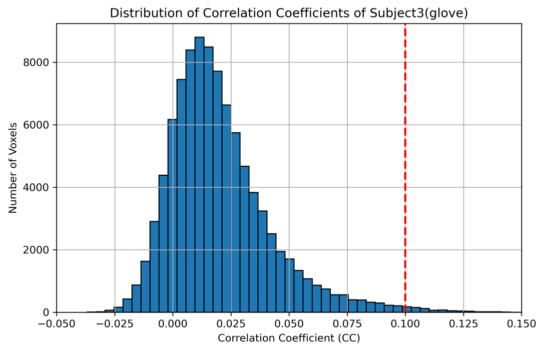


(e) Fine-tuned BERT (LoRA)

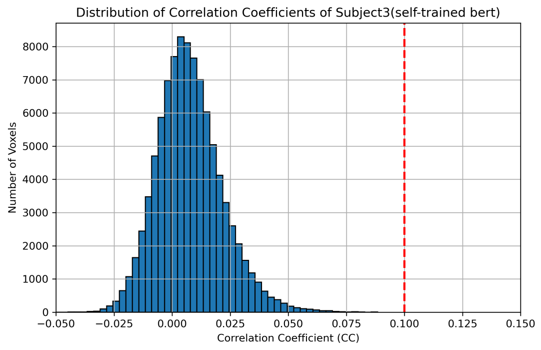
Figure 1: Distribution of correlation coefficients across voxels for Subject 2 using different embedding methods.



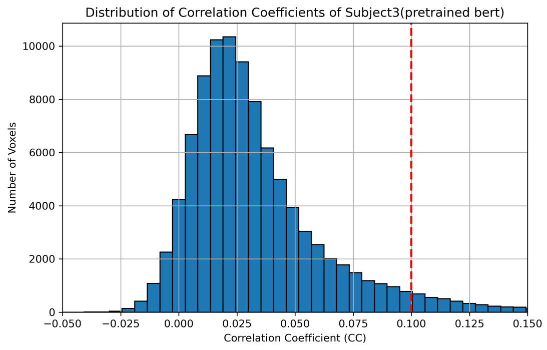
(a) Word2Vec



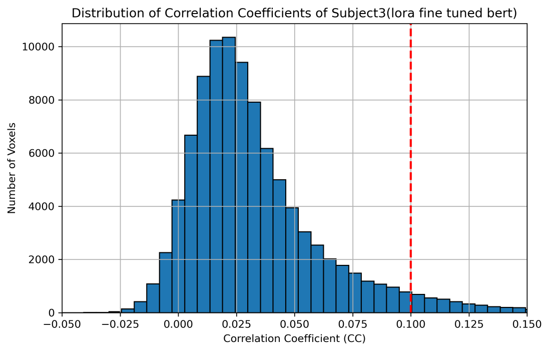
(b) GloVe



(c) BERT (Lab 3.2)



(d) Pretrained BERT-Base-Uncased



(e) Fine-tuned BERT (LoRA)

Figure 2: Distribution of correlation coefficients across voxels for Subject 3 using different embedding methods.

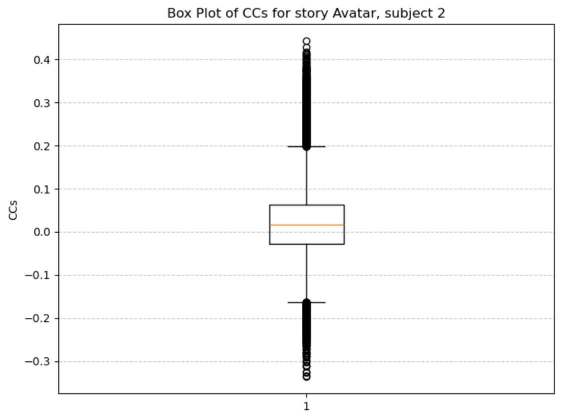
3 Interpretation

Given our first fine-tuned BERT model, we interpreted this model by investigating the word importances for various voxels.

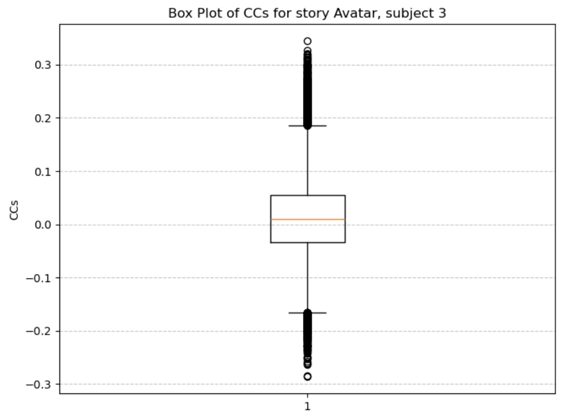
3.1 EDA

In the "Avatar" story, we began our analysis by identifying voxels where the model achieved strong predictive performance. We calculated the correlation coefficient (CC) between predicted and true voxel responses for each voxel (Figure 3). Voxels with a CC greater than 0.3 were considered "good-performing" because a larger CC indicates a stronger positive correlation. Among all these voxels, we further selected the **top 10** voxels with the highest correlation coefficients (CC) between the predicted and actual responses.

In the PCS framework, the "P" stands for Predictability. Without sufficient predictive performance, any attribution of importance to specific input features may simply reflect model noise rather than true underlying associations. Focusing on well-predicted voxels allows us to meaningfully explore which words influence neural activity, confident that the model has actually learned a reliable mapping for those regions. By restricting our interpretability analysis to this subset, we ensure that the explanatory techniques in the following part are applied to cases where the model has demonstrated significant predictive power.



(a) Subject 2



(b) Subject 3

Figure 3: Box plots of the distributions of CC in the Avatar story

3.2 Word importances

To understand what textual features influenced brain activity in the well-performing voxels, we used **SHAP** (SHapley Additive exPlanations) and **LIME** (Local Interpretable Model-agnostic Explanations) to interpret the model predictions. SHAP attributes feature importance based on cooperative game theory, while LIME explains predictions by approximating the model locally with an interpretable one. These methods were applied only to the selected voxels to ensure interpretability.

We first computed ridge regression weights for all voxels and then used them to predict test data responses. For a selected top voxel, we defined a prediction function that maps BERT embeddings to voxel responses. Using a sample of the test data:

- SHAP was applied to quantify the contribution of each input feature (word embedding) to the prediction.
- LIME perturbed the input and built a local interpretable linear model to identify influential features.

3.3 Post-EDA

We mapped the embeddings back to the plain word texts and compared the top influential words identified by SHAP and LIME for the same voxel. Both methods provided a ranked list of words based on their importance, which we visualized using bar plots in Figure 4.

SHAP and LIME identified largely overlapping sets of influential words (for example, "tom", "waits", "the", "loved"), although the ranking and emphasis differed between the two. SHAP tends to spread importance across a broader range of words, offering a more nuanced interpretation. In contrast, LIME focuses on fewer words with stronger weights, often highlighting localized or more sharply defined contributions.

The top words highlighted by both methods made intuitive sense—they were commonly used, emotionally or semantically salient terms, such as a character's name ("tom"), a verb ("loved"), or even a determiner ("the"). These results suggest that these words likely triggered distinct and interpretable neural responses. The fact that the selected words span different parts of speech also reflects the diverse linguistic features that the brain may be sensitive to during narrative comprehension.

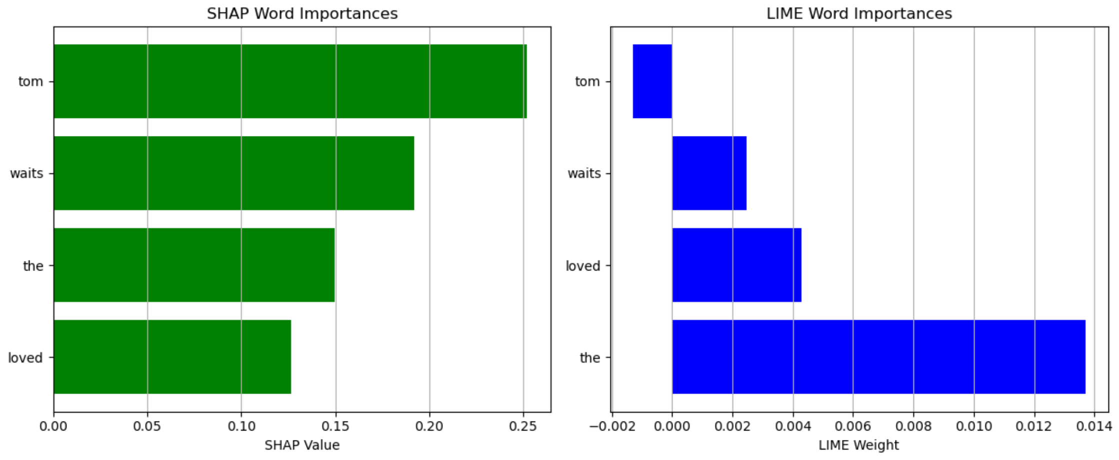


Figure 4: SHAP and LIME results indicating word importance

Overall, while SHAP and LIME demonstrated consistency in the types of words identified, their distinct mechanisms resulted in subtle differences in interpretation, underscoring the value of employing both methods for complementary insights.

We chose to visualize the top 4 voxels where the model performed well. Although each voxel's predicted activation formally sums over thousands of embedding dimensions, in practice just a handful of features dominate. In the SHAP waterfalls for "Avatar" (Fig. 5), voxel 520's response is pulled upward by a feature of magnitude $\approx +0.05$ and pushed downward by another of ≈ -0.05, while voxel 540 exhibits a large negative contribution of ≈ -0.50 alongside smaller positive effects ($\approx +0.05$). Voxels 554 and 555 each display their own distinct sparse patterns of high-magnitude SHAP values, illustrating functional specialization across cortical regions.

The LIME explanations (Fig. 6) echo this sparsity: each voxel's local linear surrogate concentrates nearly all its weight on fewer than ten dimensions. For instance, voxel 520's strongest LIME coefficients are $\approx +0.12$ and ≈ -0.07, whereas voxel 540 highlights a different pair around $\approx +0.04$ and ≈ -0.08. Although SHAP and LIME rank features slightly differently, they consistently identify the same small subset of embedding directions, boosting our confidence in these word-to-voxel associations.

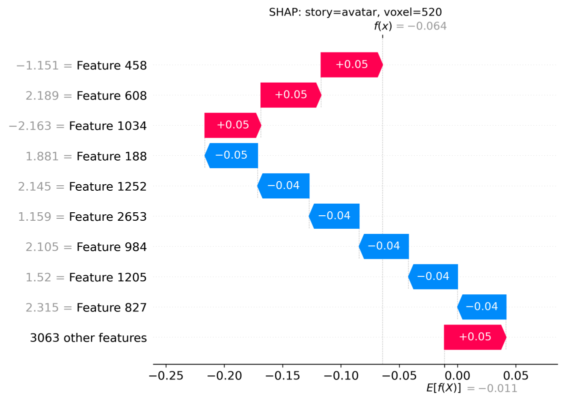
3.4 Stability Check

We packaged our interpretability pipeline into a single function so that it could be applied to any test story. To validate its robustness, we ran the same analysis on the second test story, "Backside of the Storm."

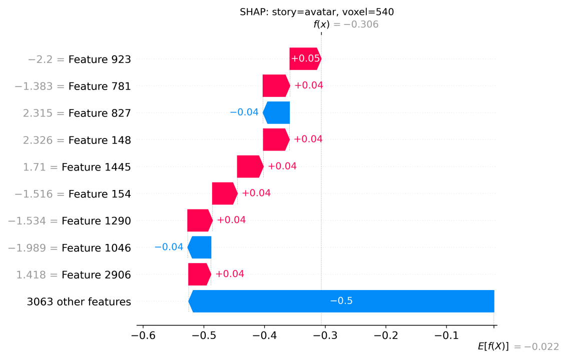
As shown in the SHAP waterfalls (Fig. 7), each voxel remains governed by only a few dominant dimensions. For example, voxel 520 receives a strong negative SHAP contribution of ≈ -0.13 and a matching positive boost of $\approx +0.13$ (and more matching pairs like this), while voxel 540 is driven down by many small contributions and up by $\approx +0.33$ for all the remaining features.

The LIME explanations (Fig. 8) mirror these patterns: the same features that appear as top drivers in SHAP also rank highly under LIME, with comparable sign and relative magnitude.

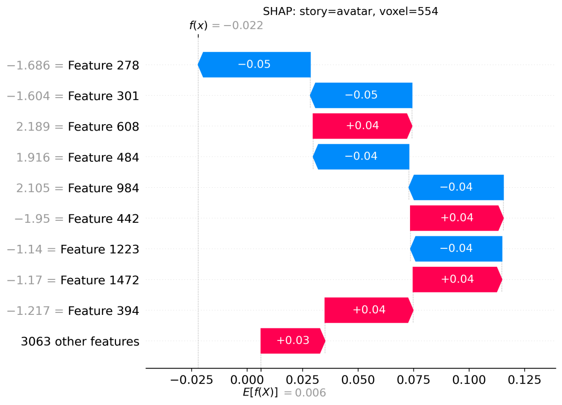
This cross-story agreement across both SHAP and LIME confirms that our feature-to-voxel mappings are not story-specific artifacts, but rather stable, meaningful associations between particular embedding dimensions and neural responses.



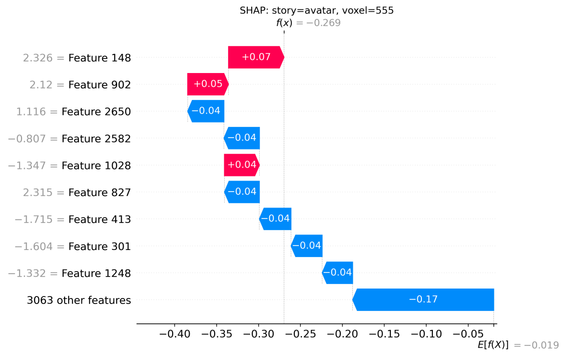
(a) voxel 520



(b) voxel 540

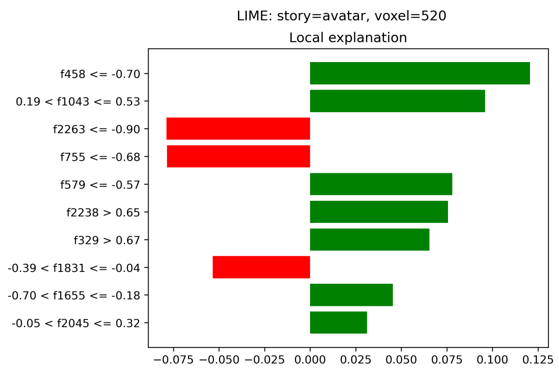


(c) voxel 554

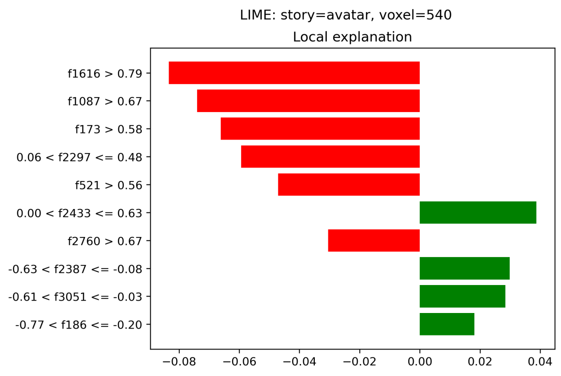


(d) voxel 555

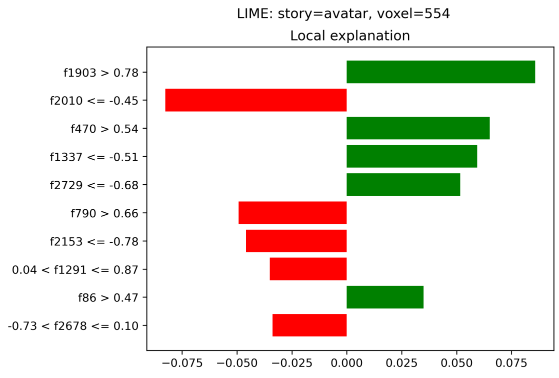
Figure 5: Combined figure showing the top 4 voxels with SHAP waterfall for the story 'Avatar'.



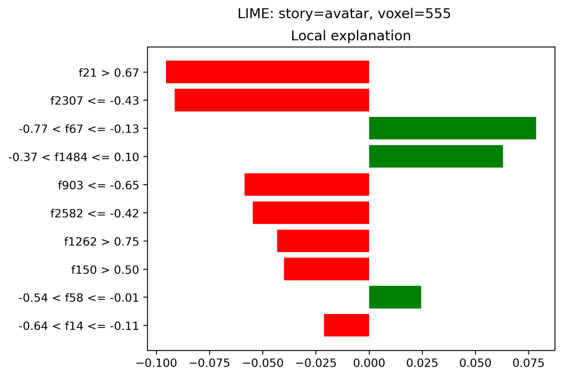
(a) voxel 520



(b) voxel 540

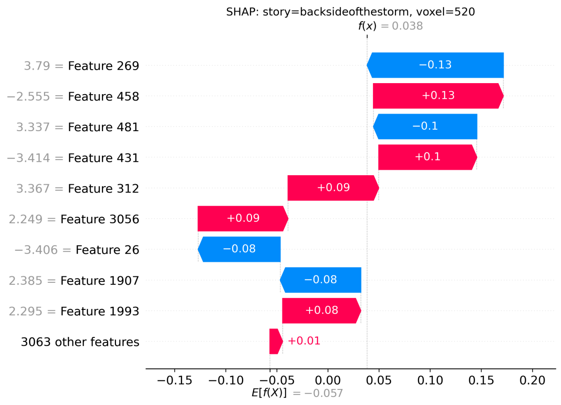


(c) voxel 554

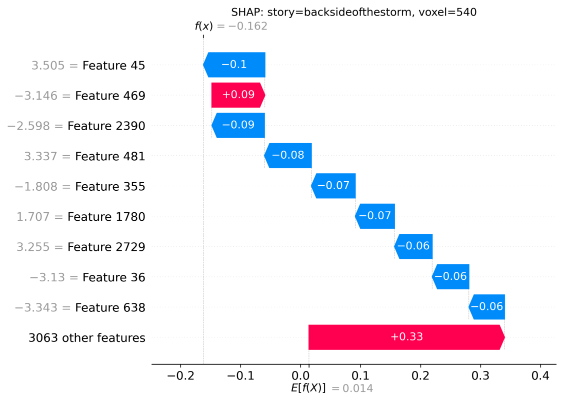


(d) voxel 555

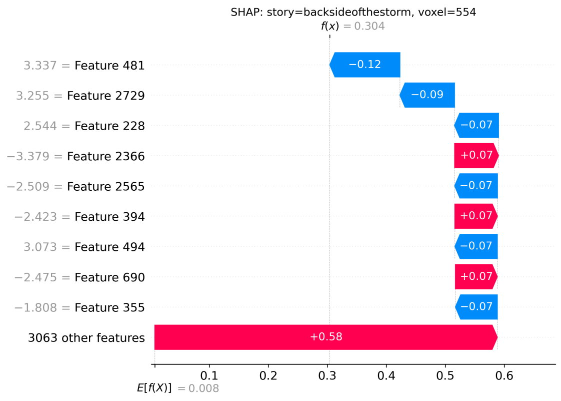
Figure 6: Combined figure showing the top 4 voxels for Lime for the story 'Avatar'.



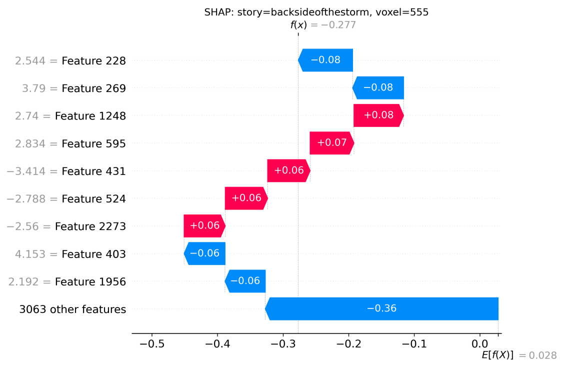
(a) voxel 520



(b) voxel 540

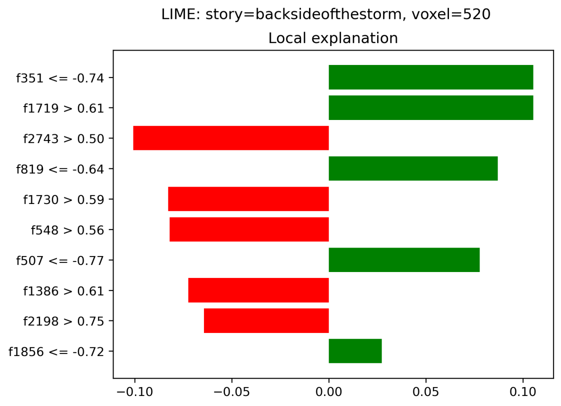


(c) voxel 554

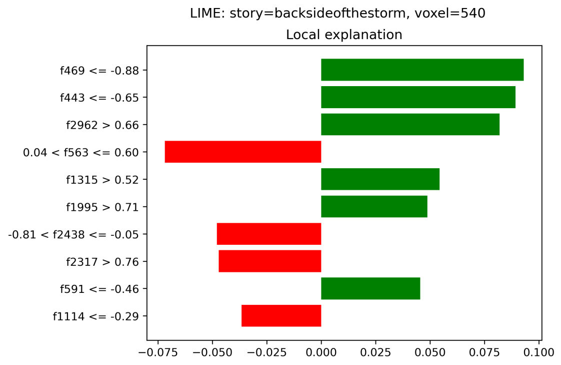


(d) voxel 555

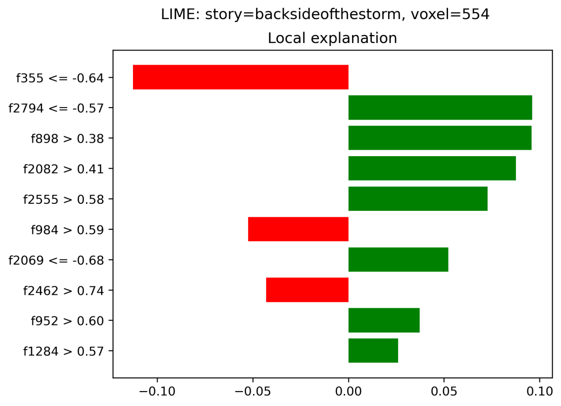
Figure 7: Combined figure showing the top 4 voxels with SHAP waterfall for the story 'Backside of the storm'.



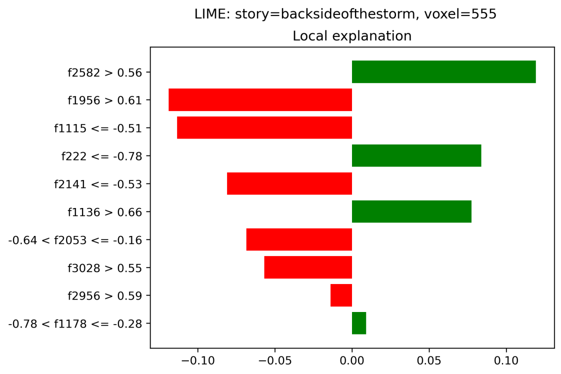
(a) voxel 520



(b) voxel 540



(c) voxel 554



(d) voxel 555

Figure 8: Combined figure showing the top 4 voxels for Lime for the story 'Backside of the storm'.

4 Bibliography

References

[1] Google Research. *BERT-Base Uncased Model.* Accessed: 2025-05-07. 2018. URL: `https://huggingface.co/google-bert/bert-base-uncased`.

A Academic honesty

A.1 Statement

We hereby make the following statements: We personally designed and conducted all data analysis processes presented in this report. We have written and produced all text and graphs included in this report. Additionally, we have incorporated and documented all procedures into our workflow, ensuring that the results are fully reproducible. We have properly attributed all references to others' work.

We strongly believe that integrity in academic research is of utmost importance. First and foremost, we must take responsibility for the results of our research. Since science thrives on collaboration, our findings will serve as the foundation for future research. Dishonest or non-reproducible research can lead to a cascade of erroneous results. Furthermore, plagiarism contributes nothing new to the scientific community and is disrespectful to the original authors, as it misappropriates credit for their work. Therefore, we are committed to maintaining honesty, transparency, and reproducibility in all aspects of our research.

A.2 LLM Usage

Coding

The code portion of this report only uses chatGPT to help refine formatting and annotation writing. We ensure that data exploration, preprocessing, model selection, comparison, interpretation, and tuning are entirely manual by the team members.

Writing

The writing portion of this report only uses chatGPT to help with the layout of the images and the rephrasing of very few sentences. We make sure that the vast majority of the content is brainstormed, discussed, organized, and typed by us in person.