Text-based Brain Encoding and Decoding

Prisha 2021101075

1. Introduction

We present the implementation and analysis of brain encoding and decoding models for textual stimuli. The brain encoding problem aims to automatically generate fMRI brain representations given a stimulus, while the brain decoding problem attempts to reconstruct the stimuli given the fMRI brain representation.
We implemented both encoder and decoder models for different brain regions of interest (ROIs) and evaluated their performance using various metrics. We explored how different sentence embedding techniques correlate with brain activity in specific regions and investigated the importance of different parts of speech in predicting brain responses. We developed stacked models that combine information from multiple brain regions to improve decoding performance.

2. Dataset Description

The dataset consists of 627 sentences and corresponding fMRI data recorded when the sentences were presented to two subjects. The fMRI data is provided for four different brain regions of interest (ROIs):

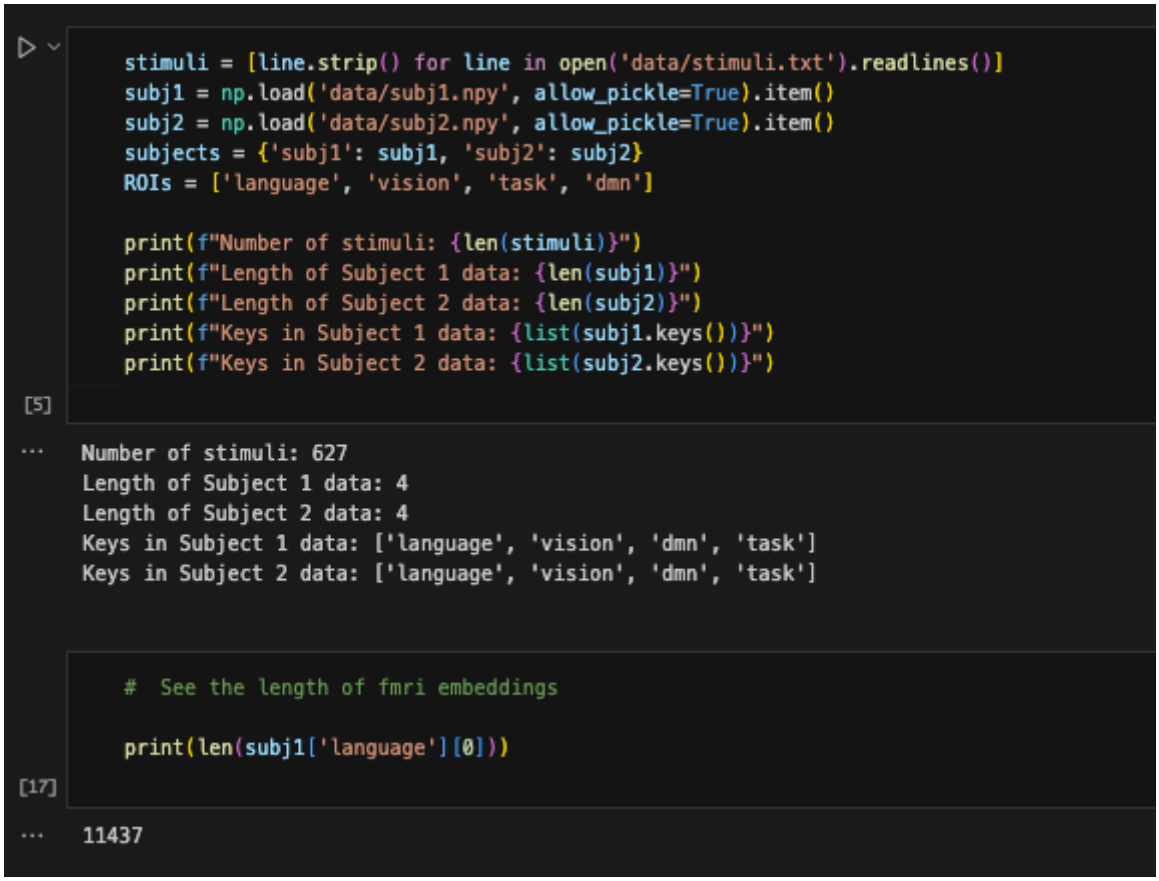
1. **Language**: Related to language processing, understanding, word meaning, and sentence comprehension
2. **Vision**: Related to the processing of visual objects and object recognition

3. **Task Positive**: Related to attention and salience information
4. **Default Mode Network (DMN)**: Linked to semantic processing functionality

The dataset contains three files:

- `stimuli.txt`: Contains 627 sentences, each on a separate line
- `subj1.npy`: fMRI data for Subject 1, stored as a dictionary with ROIs as keys
- `subj2.npy`: fMRI data for Subject 2, stored in the same format.

After loading the data, we verified the dimensions and structure to ensure proper processing:



```
stimuli = [line.strip() for line in open('data/stimuli.txt').readlines()]
subj1 = np.load('data/subj1.npy', allow_pickle=True).item()
subj2 = np.load('data/subj2.npy', allow_pickle=True).item()
subjects = {'subj1': subj1, 'subj2': subj2}
ROIs = ['language', 'vision', 'task', 'dmn']

print(f"Number of stimuli: {len(stimuli)}")
print(f"Length of Subject 1 data: {len(subj1)}")
print(f"Length of Subject 2 data: {len(subj2)}")
print(f"Keys in Subject 1 data: {list(subj1.keys())}")
print(f"Keys in Subject 2 data: {list(subj2.keys())}")
```

[5]

```
Number of stimuli: 627
Length of Subject 1 data: 4
Length of Subject 2 data: 4
Keys in Subject 1 data: ['language', 'vision', 'dmn', 'task']
Keys in Subject 2 data: ['language', 'vision', 'dmn', 'task']
```

```
#  See the length of fmri embeddings

print(len(subj1['language'][0]))
```

[17]

```
11437
```

Code that loads the dataset and verify the dimensions

3. Methodology

3.1 Sentence Representations

For this study, we extracted sentence embeddings using four different models as specified in the assignment:

1. **SBERT1** (Sentence-BERT): `all-MiniLM-L6-v2` - A lightweight and efficient sentence transformer model
2. **SBERT2** (Sentence-BERT): `all-mpnet-base-v2` - A more powerful sentence transformer model
3. **T5**: `t5-small` - A text-to-text transformer model fine-tuned for semantic tasks
4. **CLIP**: `clip-vit-base-patch32` - A vision-language model trained on image-text pairs

The sentence embeddings were generated for all 627 stimuli sentences and stored for subsequent encoding and decoding tasks:

```
[7]
def get_t5_embeddings(sentences):
    inputs = t5_tokenizer(sentences, return_tensors='pt', padding=True, truncation=True)
    with torch.no_grad():
        outputs = t5_model(**inputs)
    return outputs.last_hidden_state.mean(dim=1).numpy()

[8]
def get_clip_embeddings(sentences):
    inputs = clip_tokenizer(sentences, return_tensors='pt', padding=True, truncation=True)
    with torch.no_grad():
        outputs = clip_model(**inputs)
    return outputs.last_hidden_state.mean(dim=1).numpy()

[9]
print("Generating Embeddings...")
sbert_embeddings1 = sbert_model1.encode(stimuli, show_progress_bar=True)
sbert_embeddings2 = sbert_model2.encode(stimuli, show_progress_bar=True)
t5_embeddings = get_t5_embeddings(stimuli)
clip_embeddings = get_clip_embeddings(stimuli)

print("Calculating Length of Embeddings...")
sbert_embeddings_len1 = len(sbert_embeddings1)
sbert_embeddings2_len2 = len(sbert_embeddings2)
t5_embeddings_len = len(t5_embeddings)
clip_embeddings_len = len(clip_embeddings)

print(f"Length of SBERT Embeddings: {sbert_embeddings_len1}")
print(f"Length of SBERT Embeddings: {sbert_embeddings2_len2}")
print(f"Length of T5 Embeddings: {t5_embeddings_len}")
print(f"Length of CLIP Embeddings: {clip_embeddings_len}")

# Print Length of each embedding
print(f"Length of SBERT Embeddings: {sbert_embeddings1.shape}")
print(f"Length of SBERT Embeddings: {sbert_embeddings2.shape}")
print(f"Length of T5 Embeddings: {t5_embeddings.shape}")
print(f"Length of CLIP Embeddings: {clip_embeddings.shape}")

...
Generating Embeddings...
Batches: 100%|██████████| 20/20 [00:03<00:00,  5.49it/s]
Batches: 100%|██████████| 20/20 [00:03<00:00,  5.87it/s]
Calculating Length of Embeddings...
Length of SBERT Embeddings: 627
Length of SBERT Embeddings: 627
Length of T5 Embeddings: 627
Length of CLIP Embeddings: 627
Length of SBERT Embeddings: (627, 384)
Length of SBERT Embeddings: (627, 768)
Length of T5 Embeddings: (627, 512)
Length of CLIP Embeddings: (627, 512)
```

Code to extract sentence embedding from different models

3.2 Brain Encoding Pipeline

The brain encoding pipeline predicts fMRI activations from sentence embeddings. For each subject and each brain ROI, we trained separate

encoding models using Ridge regression with the sentence embeddings as input features and the corresponding fMRI activations as targets.

The encoding pipeline consisted of the following steps:

1. **Data Preparation**: Organize sentence embeddings (input) and fMRI data (target) for each subject and ROI
2. **Model Training**: Implement Ridge regression models with cross-validation.
3. **Evaluation**: Calculate performance metrics for each model configuration

Function for the encoder that fits regression line between stimuli and fmri response

Additionally, we conducted an interpretable brain encoding analysis by masking different parts of speech (nouns, verbs, adjectives) to understand their relative importance in predicting brain activations:

```python
def mask_pos(sentences, target_pos):
    """
    Mask the specified part-of-speech (POS) tags in the sentences.
    Args:
        sentences (list of str): List of sentences to mask.
        target_pos (str): The POS tag to mask (e.g., 'NN', 'VB').
    Returns:
        list of str: List of sentences with the specified POS tags masked.
    """
    masked = []
    for sentence in sentences:
        tokens = word_tokenize(sentence)
        tags = pos_tag(tokens)
        new_tokens = [token for token, pos in tags if not pos.startswith(target_pos)]
        masked.append(' '.join(new_tokens))
    return masked
```

```python
# Test example
sentences = [
    "The cat sat on the mat.",
    "He runs quickly through the park.",
    "She enjoys reading books."
]

# Mask all nouns (POS tag starts with 'NN')
masked_nouns = mask_pos(sentences, 'NN')
print("Masked Nouns:", masked_nouns)

# Mask all verbs (POS tag starts with 'VB')
masked_verbs = mask_pos(sentences, 'VB')
print("Masked Verbs:", masked_verbs)
```

```
Masked Nouns: ['The sat on the .', 'He runs quickly through the .', 'She enjoys reading .']
Masked Verbs: ['The cat on the mat .', 'He quickly through the park .', 'She books .']
```

Code that maks noun/adjectives/verbs and verifies the same with some examples

3.3 Brain Decoding Pipeline

The brain decoding pipeline performs the inverse task: predicting sentence embeddings from fMRI activations. Similar to the encoding task, we trained individual decoders for each subject and ROI, as well as stacked decoders that combine information from multiple ROIs.

The decoding pipeline followed these steps:

1. **Data Preparation**: Organize fMRI data (input) and sentence embeddings (target) for each subject and ROI
2. **Individual ROI Decoders**: Train Ridge regression models for each ROI separately
3. **Multi-ROI Decoder**: Develop stacked models that concatenate features from all ROIs
4. **Evaluation**: Calculate performance metrics for each decoder configuration

```python
# ========= BRAIN DECODER MODELS =========

def train_decoder(X, Y, k=5):
    """
    Train a decoder that predicts embeddings from brain activity.
    X: Brain activity data (fMRI data)
    Y: Target embeddings
    """
    kf = KFold(n_splits=k, shuffle=True, random_state=42)
    v2v_scores = []
    pc_scores = []
    best_model = None
    best_score = -1
    best_test_indices = None

    for train_idx, test_idx in kf.split(X):
        # Use Ridge regression for decoding
        model = Ridge(alpha=1.0)
        model.fit(X[train_idx], Y[train_idx])

        # Predict embeddings from test brain activity
        preds = model.predict(X[test_idx])

        # Evaluate
        v2v_score = v2v_accuracy(Y[test_idx], preds)
        pc_score = pearson_corr(Y[test_idx], preds)

        v2v_scores.append(v2v_score)
        pc_scores.append(pc_score)

        # Save the model with best 2V2 accuracy
        if v2v_score > best_score:
            best_score = v2v_score
            best_model = model
            best_test_indices = test_idx

    # Return average scores and the best model
    return np.mean(v2v_scores), np.mean(pc_scores), best_model, best_test_indices
```

```python
# ========= MULTI-ROI DECODER =========

def train_stacked_decoder(subject_data, target_embeddings, k=5):
    """
    Train a stacked decoder that combines information from multiple ROIs.
    """
    # Concatenate all ROI data
    X_combined = np.hstack([subject_data[roi] for roi in ROIs])

    # Train decoder on combined data
    v2v, pc, best_model, best_test_indices = train_decoder(X_combined, target_embeddings, k)

    return v2v, pc, best_model, best_test_indices
```

Stacked and Individual decoder that predict sentence embedding given fmri response

3.4 Evaluation Metrics

We used multiple evaluation metrics to assess the performance of our

models:

1. **2V2 Accuracy**: Measures the ability of the model to match the correct pairs of predicted and actual representations.
2. **Pearson Correlation (PC)**: Measures the linear relationship between predicted and actual values.
3. **Rank Accuracy** (for decoders): The median rank of the true sentence among all database sentences based on similarity.
4. **Top-k Accuracy** (for decoders): Percentage of cases where the true sentence is among the top k predicted sentences.
5. **Representational Similarity Analysis (RSA)**: Comparing the representational geometry of our model with human brain data.

```python
def cosine_distance(u, v):
    return 1 - cosine_similarity(u.reshape(1, -1), v.reshape(1, -1))[0][0]

def v2v_accuracy(y_true, y_pred):
    """
    Calculate the 2-vs-2 accuracy as defined in the assignment.
    """
    N = len(y_true)
    correct = 0

    for i in range(N-1):
        for j in range(i+1, N):
            # Calculate the matching score (same indices)
            score_match = (cosine_similarity(y_true[i].reshape(1, -1), y_pred[i].reshape(1, -1))[0][0] +
                           cosine_similarity(y_true[j].reshape(1, -1), y_pred[j].reshape(1, -1))[0][0])

            # Calculate the non-matching score (crossed indices)
            score_non_match = (cosine_similarity(y_true[i].reshape(1, -1), y_pred[j].reshape(1, -1))[0][0] +
                               cosine_similarity(y_true[j].reshape(1, -1), y_pred[i].reshape(1, -1))[0][0])

            # Check if matching score is higher than non-matching
            if score_match > score_non_match:
                correct += 1

    total_pairs = (N * (N - 1)) // 2
    return correct / total_pairs

def pearson_corr(y_true, y_pred):
    """
    Calculate the average Pearson correlation across all dimensions.
    """
    return np.mean([pearsonr(y_true[:, i], y_pred[:, i])[0] for i in range(y_true.shape[1])])

def rank_accuracy(y_true, y_pred, database):
```

```python
def rank_accuracy(y_true, y_pred, database):
    """
    Description:
    For each predicted embedding, this function computes the cosine similarity with all database embeddings.
    It then determines the rank of the true sentence (based on its similarity score) among all database sentences.
    The rank is 1-indexed, meaning the best possible rank is 1 (highest similarity).
    Finally, the function returns the median rank across all samples.
    """
    n_samples = y_true.shape[0]
    ranks = []

    for i in range(n_samples):
        # Calculate cosine similarity between predicted and all database embeddings
        similarities = cosine_similarity(y_pred[i].reshape(1, -1), database)

        # Get the rank of the true sentence (0-indexed)
        true_idx = np.argmax(cosine_similarity(y_true[i].reshape(1, -1), database))
        sorted_indices = np.argsort(-similarities[0])  # Descending order
        rank = np.where(sorted_indices == true_idx)[0][0] + 1  # Add 1 for 1-indexing
        ranks.append(rank)

    return np.median(ranks)

def top_k_accuracy(y_true, y_pred, database, k=5):
    """
    Calculate the percentage of cases where the true sentence is in top-k predictions.
    """
    n_samples = y_true.shape[0]
    top_k_hits = 0

    for i in range(n_samples):
        # Calculate cosine similarity between predicted and all database embeddings
        similarities = cosine_similarity(y_pred[i].reshape(1, -1), database)

        # Get the true sentence index
        true_idx = np.argmax(cosine_similarity(y_true[i].reshape(1, -1), database))

        # Get top-k predictions
        top_k_indices = np.argsort(-similarities[0])[:k]

        if true_idx in top_k_indices:
            top_k_hits += 1

    return top_k_hits / n_samples
```

Implementation of all the metrics as mentioned in the assignment.

4. Results and Analysis

4.1 Brain Encoding Results

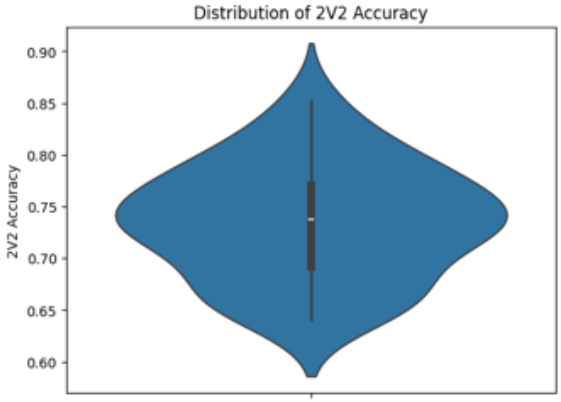
Our encoding models demonstrated varying performance across different ROIs, subjects, and embedding methods. The results provide insights into which brain regions are more predictable from linguistic features and which embedding techniques better capture relevant semantic information.

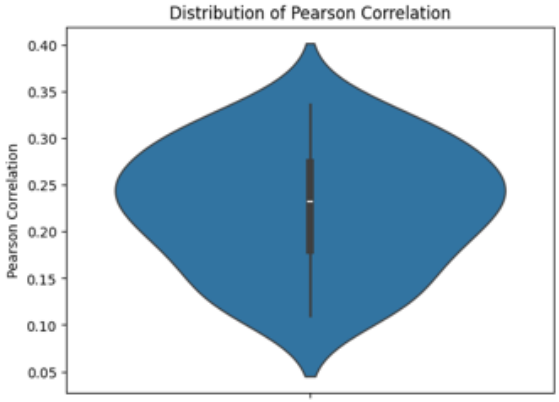
Analysis of Encoding Performance Across Metrics

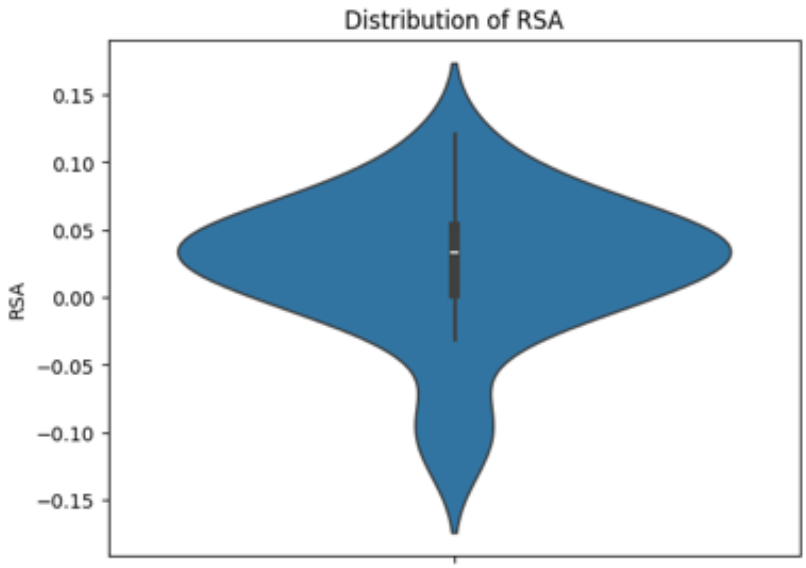
From the graphs below, we observe that the **2V2 accuracy** across all experiments—considering different ROIs, subjects, and sentence embeddings—ranges between **0.6 and 0.9**, with the distribution centered around **0.75**. This indicates that the encoding models are generally effective at predicting brain activity patterns from sentence embeddings.

The **Pearson correlation** distribution peaks around **0.25**, which is a meaningful value, showing that the predicted brain responses are well-aligned with the true fMRI responses.

Similarly, the **RSA (Representational Similarity Analysis)** values range from **-0.15 to 0.15**, with most values greater than zero. This positive skew suggests that there is a noticeable similarity between the representational structure of the predicted brain responses and the original stimulus embeddings, indicating successful encoding at a representational level.



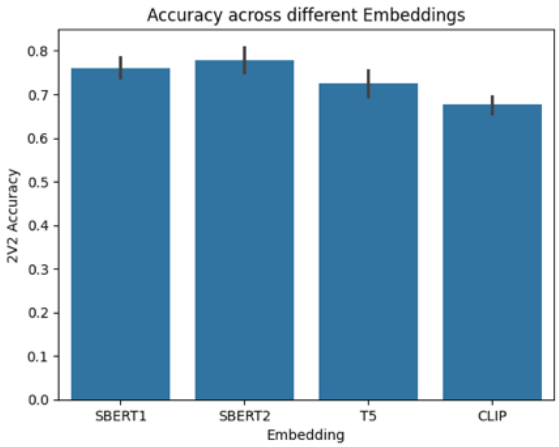


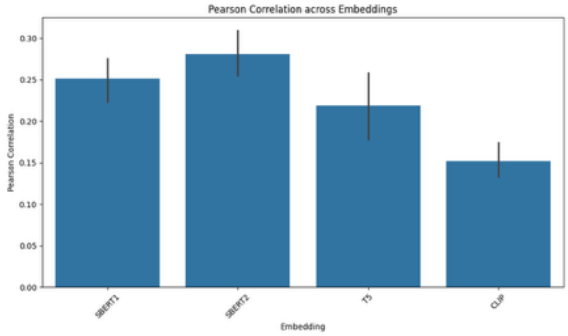


Performance Comparison Across Different Sentence Embeddings

The plots compare the performance of different sentence embeddings (SBERT1, SBERT2, T5, and CLIP) on two metrics: **2V2 accuracy** and **Pearson correlation**.

- **SBERT2** achieves the highest performance across both metrics, suggesting it encodes semantic information in a way that best aligns with brain activity patterns.
- **SBERT1** follows closely, while **T5** shows moderate performance.
- **CLIP** performs the worst among the four embeddings, indicating that its representation may not align well with neural patterns in this context.

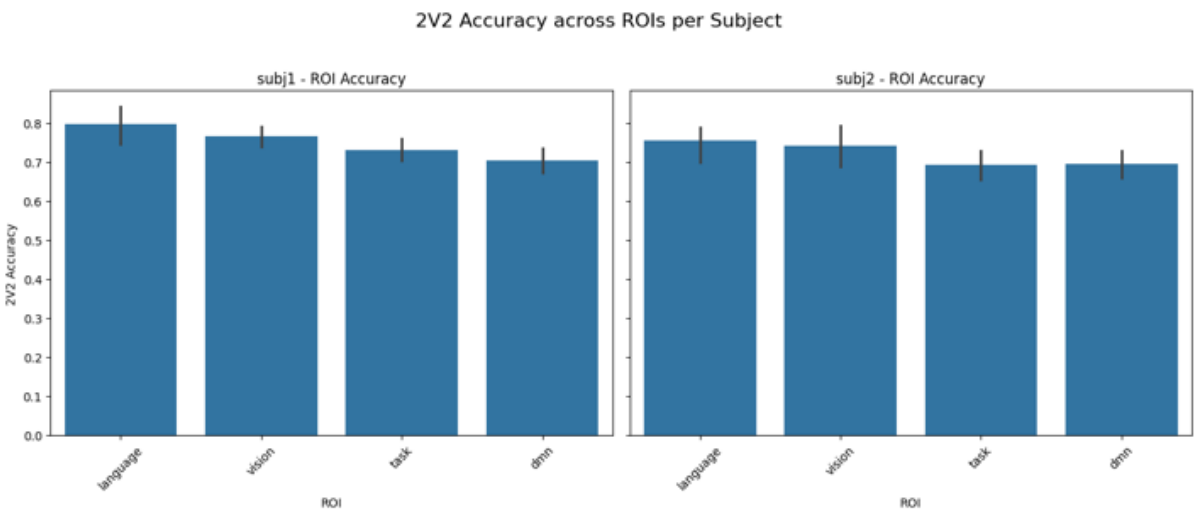


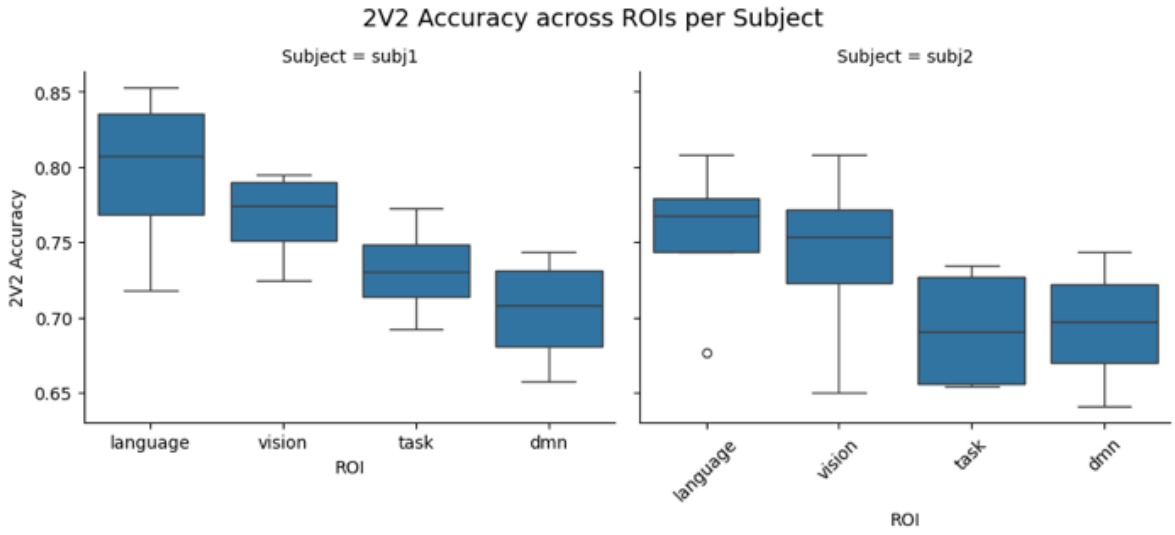


2V2 Accuracy across ROIs for Each Subject

This plot shows 2V2 accuracy across four brain regions (ROIs) for two subjects.

- **Language regions** consistently show the highest accuracy, indicating strong alignment with semantic embeddings.
- Accuracy decreases across **vision**, **task**, and **DMN** ROIs, suggesting weaker semantic representation in those areas.
- The trend is consistent across both subjects, reinforcing the role of language ROIs in capturing semantic content.

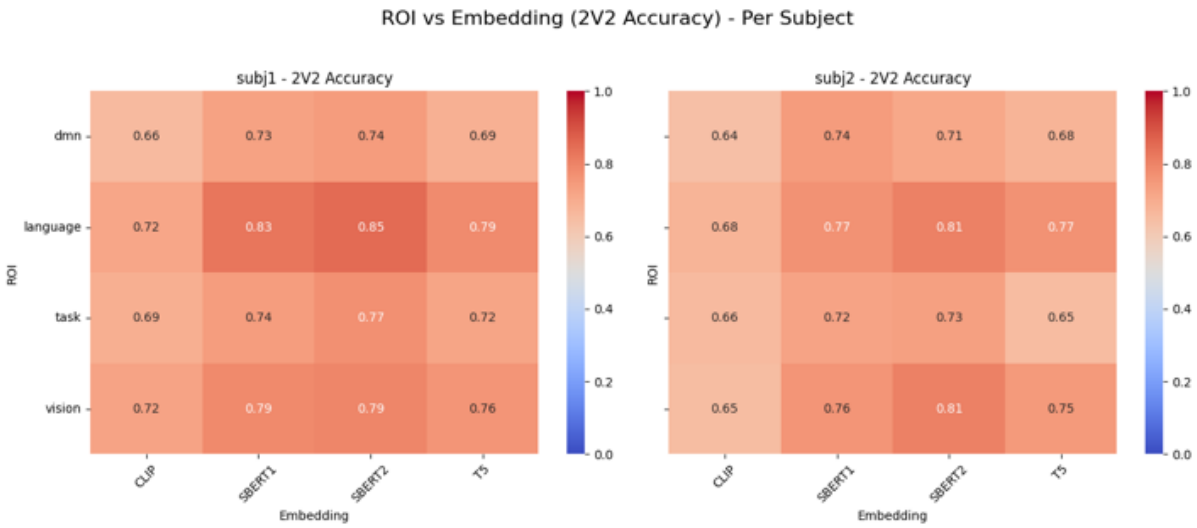




2V2 Accuracy by ROI and Embedding Model per Subject

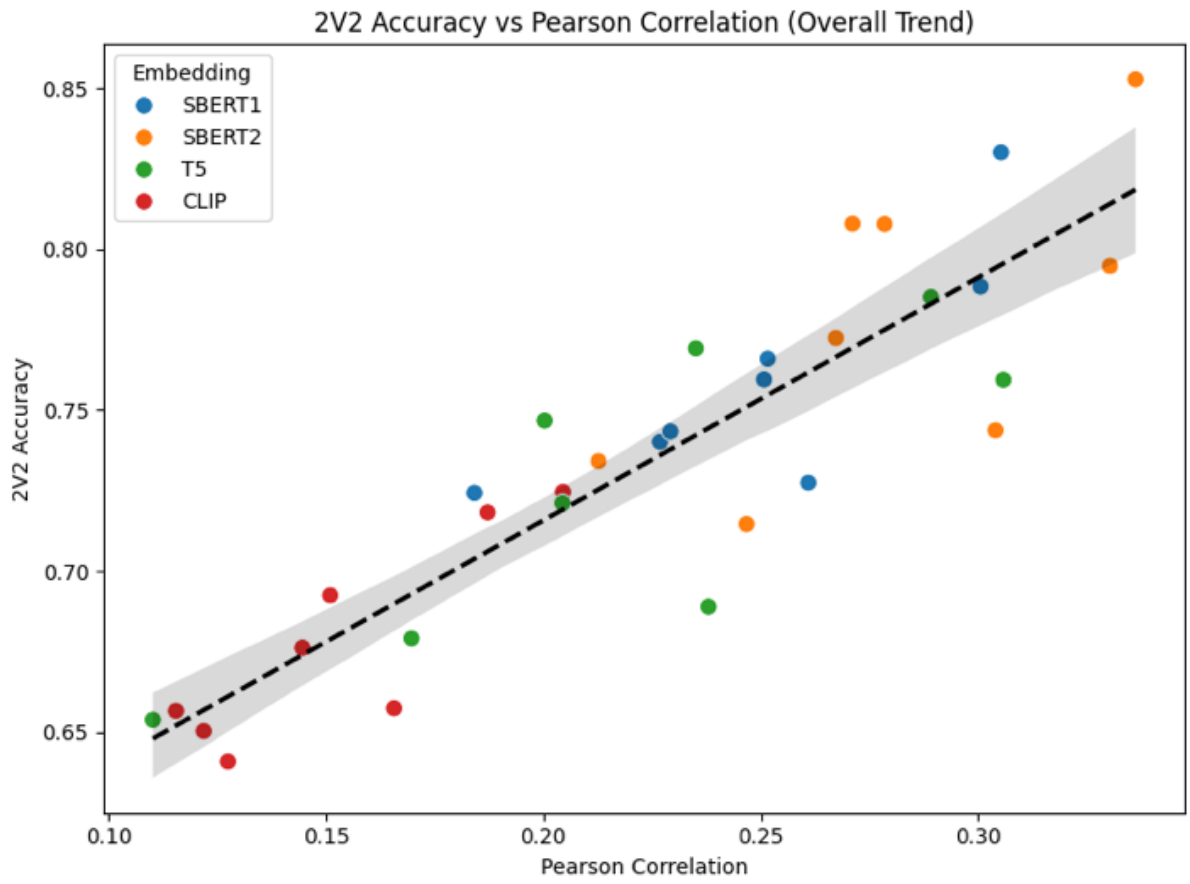
Heatmap shows how well each brain region (ROI) aligns with different semantic embedding models for both subjects.

- **Language ROIs** achieve the highest 2V2 accuracy, especially with **SBERT2**, indicating strong semantic correspondence.
- **SBERT models** outperform CLIP and T5 across most ROIs and both subjects.
- Accuracy patterns are consistent across subjects, highlighting the robustness of SBERT-based embeddings in modeling brain responses.



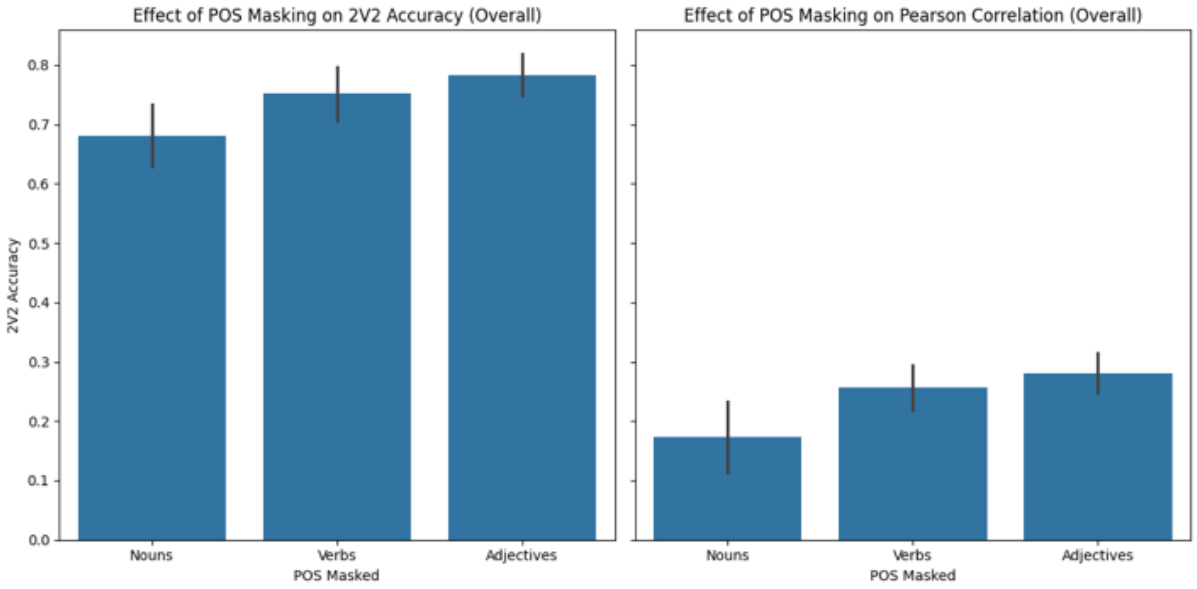
2V2 Accuracy vs Pearson Correlation

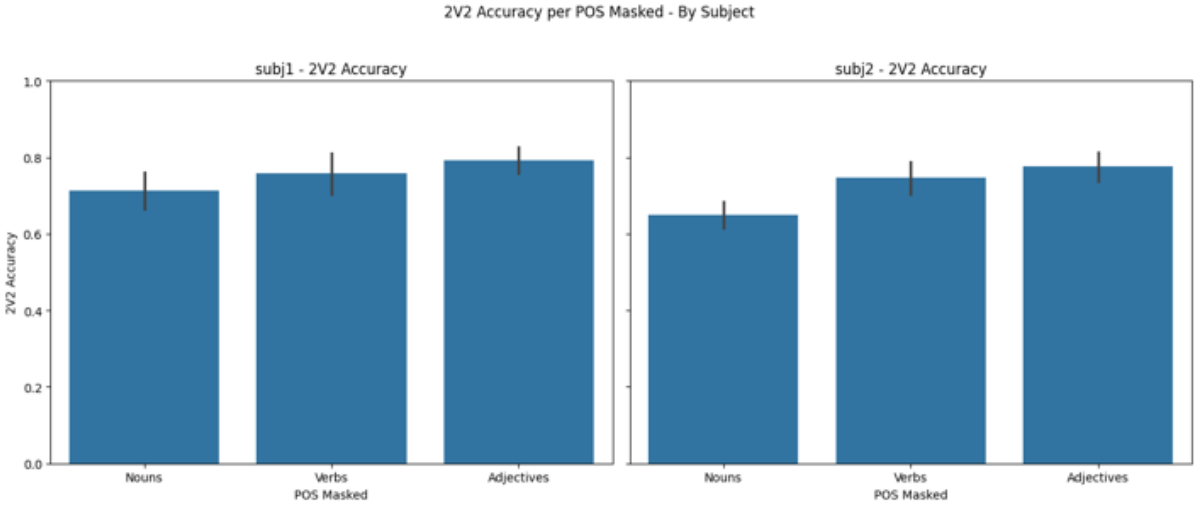
We use `pearsonr` to calculate the correlation between `2V2 Accuracy` and `Pearson Correlation`. The computed value is `0.891`, indicating a strong positive correlation. This suggests that as `2V2 Accuracy` increases, `Pearson Correlation` also tends to increase.

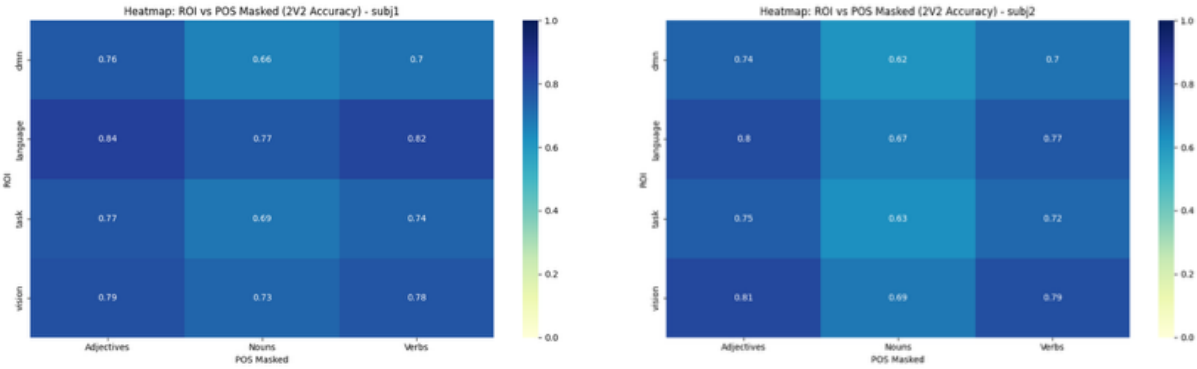


4.2 POS Masking Analysis

To understand the contribution of different parts of speech (POS) to brain activations, we conducted masking experiments where specific POS categories (nouns, verbs, adjectives) were removed from the sentences before generating embeddings:







Key findings from the POS masking analysis:

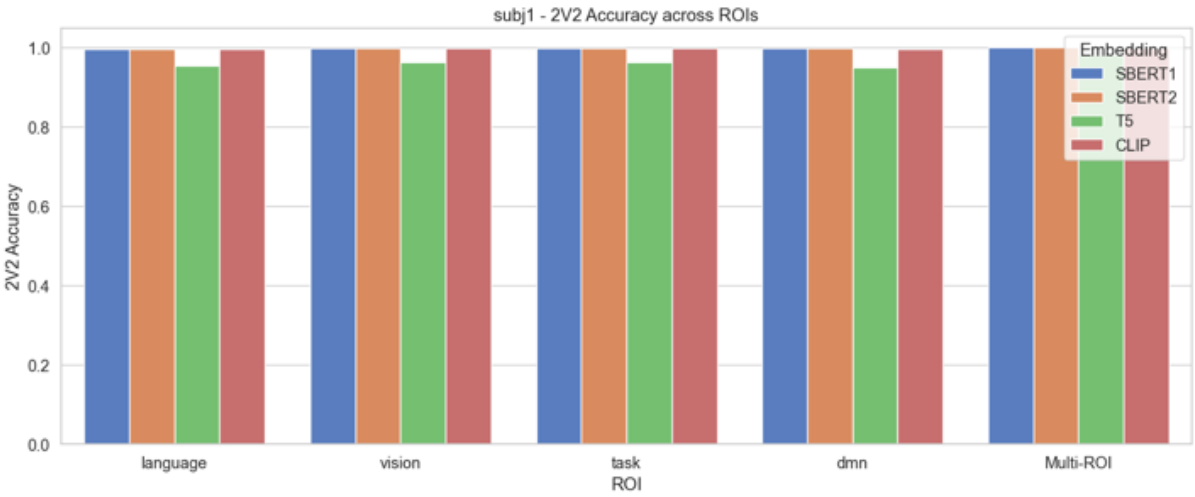
- **Nouns** had the most significant impact on encoding performance when masked, resulting in the largest drop in both 2V2 accuracy and Pearson correlation across all ROIs. This indicates that nouns carry crucial semantic information that strongly correlates with brain activity.
- **Verbs** showed moderate importance, particularly in the Language and Task Positive ROIs, suggesting their role in action-related processing.
- **Adjectives** had the least impact when masked, though they still contributed to overall performance.
- The effect of masking was most pronounced in the Language ROI, demonstrating its sensitivity to semantic content.

4.3 Brain Decoding Results

Our decoding models aimed to reconstruct sentence embeddings from fMRI activations. We evaluated individual ROI decoders as well as multi-ROI stacked decoders:

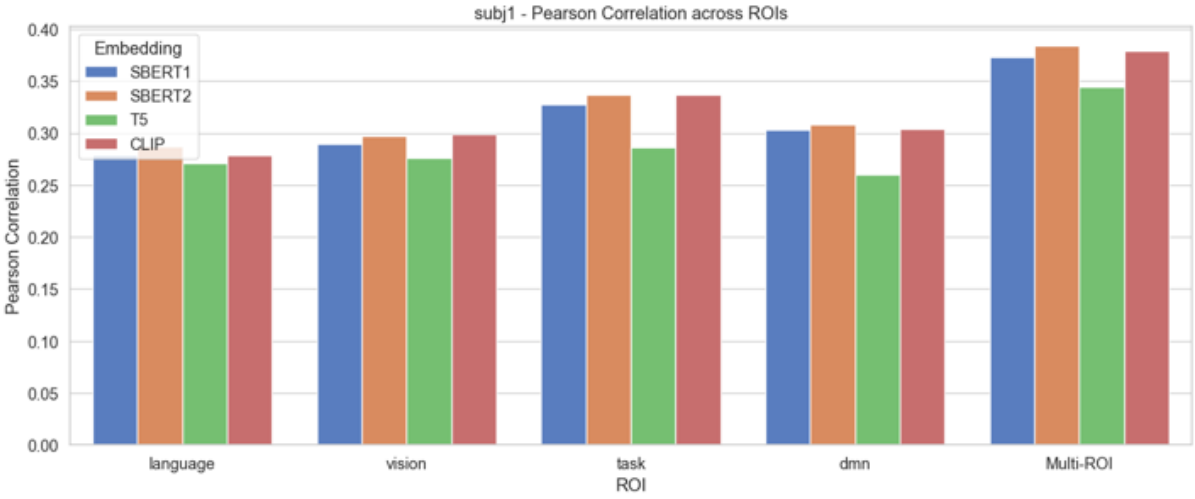
2V2 Accuracy Across ROIs

This plot shows the average 2V2 classification accuracy across different ROIs for subject 1, grouped by embedding type. SBERT-based embeddings perform best overall, with language and vision ROIs yielding higher accuracies.



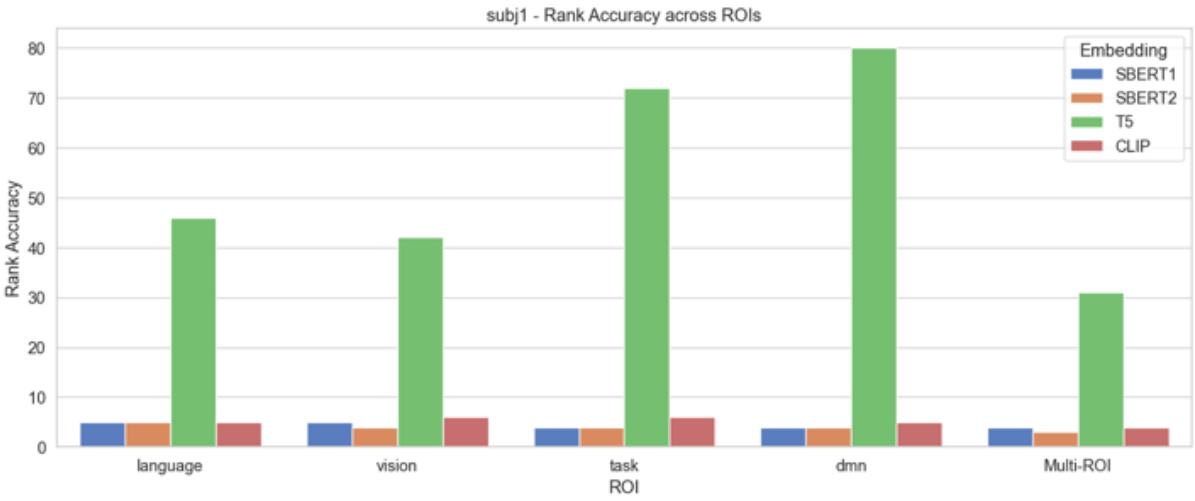
Pearson Correlation Across ROIs

This bar graph displays the Pearson correlation between predicted and true neural responses for each ROI and embedding type. SBERT generally exhibit stronger correlations, particularly in multi-ROI setups.



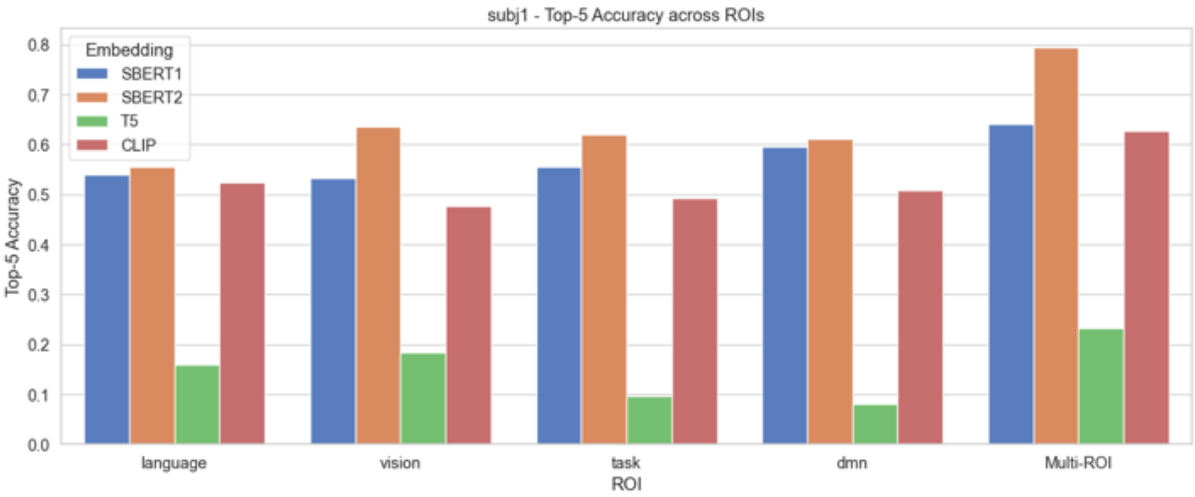
Rank Accuracy Across ROIs

This plot presents the rank of correct matches among all candidates (lower is better) for each embedding across ROIs. T5 performs notably worse than language-based models, especially in task and dmn ROIs.



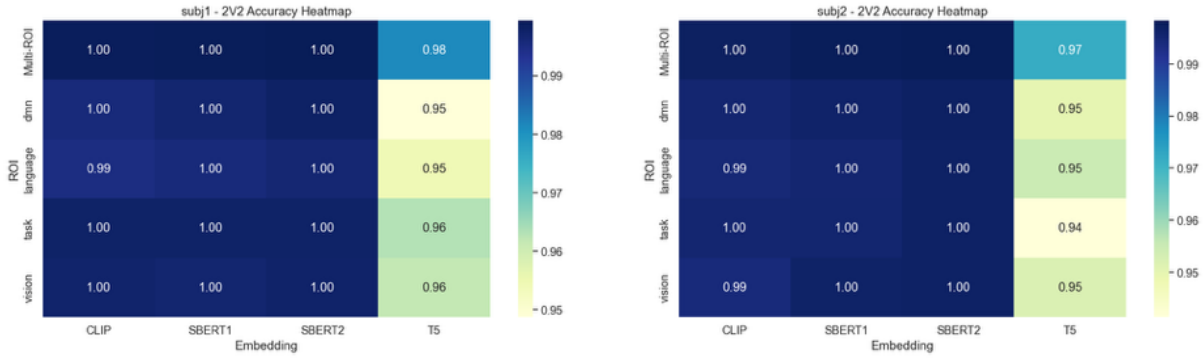
Top-5 Accuracy Across ROIs

This plot shows the proportion of cases where the correct label is within the top-5 predictions. SBERT2 embeddings lead in most ROIs, especially in multi-ROI combinations, highlighting their broader semantic capture capability.



2V2 Accuracy Heatmap Across ROIs and Embeddings

This heatmap visualizes 2V2 classification accuracy scores across various ROIs for subject 1, grouped by embedding type. Language-based embeddings (SBERT1 and SBERT2) achieve near-perfect performance across all ROIs. T5 shows slightly lower accuracies, particularly in language and DMN regions, while CLIP performs comparably to SBERT models in most ROIs but slightly underperforms in the language region.



Key observations from the decoding results:

- The **Language ROI** provided the most informative signals for decoding sentence embeddings, aligning with its role in language processing.
- **SBERT2** embeddings were most accurately decoded, followed by SBERT1, indicating that these representations better capture brain-relevant linguistic features.
- **Top-5 accuracy** showed promising results, with the best models able to identify the correct sentence among the top 5 candidates at a rate significantly above chance.

4.4 Multi-ROI Decoder Performance

The stacked decoder that combined information from all four ROIs demonstrated substantial improvements over individual ROI decoders: **Multi-ROI stacked decoders** consistently outperformed individual ROI decoders, highlighting the benefits of integrating information from multiple brain regions.

Bonus: Text Recreation from Predicted Embeddings

As a bonus task, we attempted to reconstruct the original stimulus sentences from the predicted embeddings generated by our decoders. This involved finding the closest sentence in our database to each predicted embedding based on cosine similarity:

These results suggest that different brain regions encode complementary

aspects of linguistic information, and integrating these signals provides a more comprehensive representation of the semantic content.

```
Recreating text for subj1 using SBERT2 embeddings:
Sample 1:
  Original: An accordion is a portable musical instrument with two keyboards.
  Predicted: Accordions produce sound with bellows that blow air through reeds.
  Similarity: 0.5972

Sample 2:
  Original: Some apartments are for single people, others for families.
  Predicted: An apartment may have one or more rooms, as well as a kitchen and a bathroom.
  Similarity: 0.6161

Sample 3:
  Original: The apartment building can have a garage, a laundry facility or extra storage space.
  Predicted: An apartment may have one or more rooms, as well as a kitchen and a bathroom.
  Similarity: 0.4563

Sample 4:
  Original: Apples have thin skin, a crisp, sweet pulp and seeds inside.
  Predicted: Apples have thin skin, a crisp, sweet pulp and seeds inside.
  Similarity: 0.5179
```