Text-based Brain Encoding and Decoding

Cognitive Science and AI: Assignment 4

April 10, 2025

1 Instructions for submission

Deadline: 17/04/2025

Maximum marks - 100

- You may do the assignment in Jupyter or Colab notebook or a script that executes the code.
- A report should be submitted that includes all the deliverables. Report and code should be included in a folder specified by Roll Number and Name of the student and submitted in Moodle, adhering to the deadline.
- Include the assignment number, your name and roll number in the notebook/script as well for better identity.
- Late submissions are NOT accepted.
- IMPORTANT: Make sure that the assignment that you submit is your own work. Do not copy any part from any source including your friends, seniors. Any breach of this rule could result in serious actions including an F grade in the course.
- Your marks will depend on the correctness / convincing discussion points. In addition, due consideration will be given to the clarity and details of your answers and the legibility and structure of your code.
- Do not copy or plagiarise, if you're caught for plagiarism or copying, penalties are much higher (including an F grade in the course) than simply omitting that question.

2 Objective

The brain encoding problem aims to automatically generate fMRI brain representations given a stimulus. The brain decoding problem is the inverse problem of reconstructing the stimuli given the fMRI brain representation. In this assignment, your task would be to construct an encoder as well as a decoder for textual stimuli. Details about the dataset, methodology and the tasks are provided in the section below.

3 Dataset

The dataset to be used for the assignment can be downloaded from here. Dataset consists of 627 sentences and the corresponding fMRI, recorded when the sentences were presented to a subject one by one. fMRI is provided for four different brain ROIs listed below:

- **Language**: Related to language processing, understanding, word meaning, and sentence comprehension
- **Vision**: Related to the processing of visual objects, object recognition
- **Task Positive**: Related to attention, salience information
- **Default Mode Network (DMN)**: Linked to the functionality of semantic processing.

Dataset contains three files:

- **stimuli.txt**: It contains 627 sentences, each in one line.
- **subj1.npy**: Contains fMRI data for Subject 1. Stored as a dictionary, with keys as the four brain regions and values as the corresponding fMRI for 627 sentences.
- **subj2.npy**: Contains fMRI data for Subject 2. Stored as a dictionary, with keys as the four brain regions and values as the corresponding fMRI for 627 sentences.

The data in the npy files can be accessed using the following python code:

"data = np.load("subj1.npy").item()"

4 Task - Brain Encoder

4.1 Deep Encoding Models

The first task is to build 4 different encoders one for each brain ROI region using models described in Section-6. This is to be done for each subject. For example, Encoder for Vision area of subject 1 would predict the vision area voxels of subject 1 given the sentence representations.

The encoder could be simply a regression based model like ridge/lasso regression. You should perform a k-fold cross validation, and report the average of the evaluation metric across folds.

4.1.1 Evaluation Metrics

You need to evaluate your encoders using both the evaluations metrics listed below.

Given a subject and a brain region, let N be the number of samples. Let $\{Y_i\}_{i=1}^{N}$ and $\{\hat{Y}_i\}_{i=1}^{N}$ denote the actual and predicted voxel value vectors for the i^{th} sample. Thus, $Y \in \mathbb{R}^{N \times V}$ and $\hat{Y} \in \mathbb{R}^{N \times V}$, where V is the number of voxels in that region.

2V2 Accuracy is computed as follows:

$$\text{2V2Acc} = \frac{1}{\binom{N}{2}} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \mathbb{I}\left[\{\cos D(Y_i, \hat{Y}_i) + \cos D(Y_j, \hat{Y}_j)\} < \{\cos D(Y_i, \hat{Y}_j) + \cos D(Y_j, \hat{Y}_i)\}\right]$$

where $\cos D$ is the cosine distance function, and $\mathbb{I}[c]$ is an indicator function such that $\mathbb{I}[c] = 1$ if the condition c is true, and 0 otherwise. The higher the 2V2 accuracy, the better.

Pearson Correlation (PC) is computed as

$$PC = \frac{1}{N} \sum_{i=1}^{N} \text{corr}\left(Y_i, \hat{Y}_i\right)$$

where $corr$ refers to the correlation function.

4.2 Interpretable Brain Encoding

Your model should provide feature attribution showing which linguistic features drive activation in specific brain regions.

This can be done by masking different types of parts of speech separately (masking verbs, nouns, adjectives), extracting sentence embeddings, and then performing encoding to understand which POS has better results and correlation to which ROI.

4.2.1 Evaluations

In addition to 2V2 Accuracy and Pearson Correlation, you will evaluate:

1. **Information Mapping**: Visualize which linguistic features are encoded in different brain regions.
2. **Representational Similarity Analysis (RSA)** Compare the representational geometry of your model with human brain data.

5 Task - Brain Decoder

5.1 Deep Decoding Models

The second task is to build 4 different decoders one for each brain ROI using models described in Section-6. This needs to be done for each subject. For example, Decoder for Vision area of subject 1 would predict the sentence representations given the vision area voxels of subject 1.

The decoder could be simply a regression based model like ridge/lasso regression or any of your choice. You should perform a k-fold cross validation, and report the average of the evaluation metrics across folds.

5.1.1 Evaluation Metrics

You need to evaluate your decoders using both the evaluation metric listed below.

Given a subject and a brain region, let N be the number of samples. Let $\{Y_i\}_{i=1}^{N}$ and $\{\hat{Y}_i\}_{i=1}^{N}$ denote the actual and predicted sentence vectors for the i^{th} sample. Thus, $Y \in \mathbb{R}^{N \times D}$ and $\hat{Y} \in \mathbb{R}^{N \times D}$, where D is the dimension of each sentence vector.

2V2 Accuracy is computed as follows:

$$\text{2V2Acc} = \frac{1}{\binom{N}{2}} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} \mathbb{I}\left[\{\cos D(Y_i, \hat{Y}_i) + \cos D(Y_j, \hat{Y}_j)\} < \{\cos D(Y_i, \hat{Y}_j) + \cos D(Y_j, \hat{Y}_i)\}\right]$$

where $\cos D$ is the cosine distance function, and $\mathbb{I}[c]$ is an indicator function such that $\mathbb{I}[c] = 1$ if the condition c is true, and 0 otherwise. The higher the 2V2 accuracy, the better.

Pearson Correlation (PC) is computed as

$$PC = \frac{1}{N} \sum_{i=1}^{N} \text{corr}\left(Y_i, \hat{Y}_i\right)$$

where $corr$ refers to the correlation function.

5.2 Multi-ROI Decoder

Design stacked regressors to effectively combine information from multiple brain regions. (ROI) for decoding

5.2.1 Evaluation metrics

In addition to 2V2 Accuracy and Pearson Correlation as defined in the original assignment, you will evaluate:

1. **Rank Accuracy**: Rank all possible sentences and report the median rank of the true sentence.
2. **Top-k Accuracy**: Percentage of cases where the true sentence is among the top k predicted sentences.

5.3 Text recreation from predicted embeddings (Bonus)

Try to recreate the actual sentences from predicted text embeddings.

6 Sentence Representations

You have to extract the sentence embeddings from four models as mentioned below:

1. The **most performant**, and **most light weight** of Sentence Transformers model from SentenceTransformers a.k.a SBERT (2 models)
2. GPT variants or other recent models (e.g., T5) fine-tuned on semantic similarity tasks (1 model)
3. CLIP text representations for encoding (1 model)

7 Deliverables

A complete end-to-end pipeline implementation should be provided in the form of Jupyter Notebook or Google Colab. This notebook should be executable by just one click. While submitting all notebooks should have the outputs rendered (inlined) wherever necessary for a quick evaluation. Only those notebooks that shows the rendered outputs is considered for further evaluation.

A report should discuss in detail, the process of creating a pipeline and end-results + visualizations wherever necessary.