Lab 3.1 - FMRI, Stat 214, Spring 2025

April 15, 2025

1 Introduction

Understanding how the human brain processes language is a fundamental challenge at the intersection of neuroscience and artificial intelligence. The rise of functional magnetic resonance imaging (fMRI) has enabled researchers to explore how the brain responds to linguistic stimuli, offering voxel-level recordings of neural activity as individuals engage with natural speech. This lab builds on prior work that leverages machine learning and natural language processing to bridge the gap between linguistic stimuli and corresponding brain responses.

The foundational paper for this lab, "Incorporating Context into Language Encoding Models for fMRI"[1] by Jain and Huth (2018), demonstrates that contextualized word representations—learned via a Long Short-Term Memory (LSTM) language model—can predict brain responses significantly better than traditional word embeddings. Unlike static embeddings such as Word2Vec, which treat each word in isolation, contextual models leverage preceding words to create a dynamic, context-sensitive representation that more closely reflects how humans process language.

In their experiment, Jain and Huth collected fMRI data from subjects listening to over two hours of stories, aligning brain activity with exact word timings. They showed that models incorporating contextual information (i.e., considering temporal structure and narrative flow) achieved higher encoding accuracy across nearly all cortical regions. Their work highlights not only the scientific utility of deep learning representations for modeling the brain but also opens new doors for interpreting brain function through learned embeddings.

In this lab, we aim to replicate and extend aspects of Jain and Huth's methodology by evaluating the predictive power of three types of word embeddings—Bag-of-Words (BoW), Word2Vec (W2V), and GloVe—on fMRI responses from two subjects. Our objective is to determine how well these different embedding methods can predict voxel-level BOLD signals using ridge regression. Specifically, we seek to answer the following questions:

- How well can each embedding method predict neural responses?
- What are the performance differences across brain voxels?
- Which embedding method yields the most reliable and interpretable predictions?

For this purpose, we construct lagged and downsampled embedding matrices aligned with fMRI time points and fit voxel-wise ridge regression models for each embedding type. We evaluate model performance mainly using correlation coefficients(CC), including mean and median CCs, as well as top percentile CCs, across voxels.

2 Data Structure and Exploratory Data Analysis (EDA)

For this lab, we worked with fMRI data and natural language story transcripts to build models that predict brain activity from linguistic input. The data was accessed through the shared directory on the Bridges2 remote server: */ocean/projects/mth240012p/shared/data/*.

This directory contains three main components:

- `subject2/`: fMRI response data for Subject 2
- `subject3/`: fMRI response data for Subject 3
- `raw_text.pkl`: Raw story transcripts and metadata

Each of the `subject2/` and `subject3/` folders contains 101 `.npy` files, one for each story. When loaded, each

file yields a matrix $Y \in R^{T' \times V}$, where T' is the number of fMRI time points (TRs), and V is the number of voxels. While the number of TRs is consistent for a given story across subjects, the number of voxels differs: Subject 2 has 94,251 voxels, and Subject 3 has 95,556. Given this discrepancy, we train and evaluate separate models for each subject.

The `raw_text.pkl` file is a Python dictionary mapping each story name to a `DataSequence` object. These objects encapsulate several fields, including:

- `data`: a list of word tokens in the story
- `data_times`: onset times (in seconds) for each word
- `tr_times`: centered fMRI scan timestamps (one every 2 seconds)
- `split_inds`: indices used to align words into TR-level chunks

These attributes are critical for aligning language input with brain activity data and serve as the foundation for embedding construction and downsampling.

For example, in the story *penpal*, we found approximately 1,590 word tokens and 270 fMRI time points. This confirms that each fMRI scan spans about 5–6 words, consistent with the 2-second TR resolution.

To support data processing and modeling, the provided codebase includes two utility folders: `code/preprocessing.py` and `ridge_utils/`. These contain custom functions for feature generation (`make_delayed`), temporal alignment (`downsample_word_vectors`), and optimized ridge regression routines for high-dimensional voxel-wise prediction.

Upon inspection, we observed that `raw_text.pkl` includes 109 stories, while both `subject2/` and `subject3/` include only 101 stories. The additional 8 stories were excluded since they lacked corresponding response matrices, making them unusable for modeling. After filtering, we retained the 101 matched stories and split them into training and test sets using a 70:30 ratio:

- **Training set:** 70 stories
- **Testing set:** 31 stories

This split was used consistently across all embedding types (e.g., Bag-of-Words, Word2Vec, GloVe). The training set was used to generate embeddings and tune hyperparameters (e.g., ridge regression alphas), while the test set was reserved for evaluating models per embedding type.

3 Embeddings

An embedding is a numerical vector representation of text that allows linguistic input to be used in predictive models. Since the fMRI response Y is numerical, we convert each story's sequence of words into embedding matrices to serve as the input matrix X. Based on the train-test split described in the EDA section, we generate embeddings using three methods: Bag-of-Words (BoW), Word2Vec (W2V), and GloVe. For each subject, we generate embedding matrices for both train and test stories for each of the three methods. This results in a total of 12 dictionaries (3 embedding types $\times$ 2 data splits $\times$ 2 subjects).

3.1 Bag-of-Words (BoW)

The Bag-of-Words method represents each word in a story as a high-dimensional one-hot vector, where each vector has a dimension equal to the vocabulary size (i.e., the total number of unique words across all training stories). Each element in a one-hot vector corresponds to a word in the vocabulary: a value of 1 indicates the presence of the word at that position, while all other elements are 0.

3.1.1 Creating vector per word

To ensure consistency across stories, we first compile a vocabulary of all unique words in the training set. For each story, we then convert its sequence of words into a matrix of one-hot vectors, where each row corresponds to a word token and each column corresponds to a vocabulary word. This results in a BoW matrix of shape $(T \times V)$ for each story, where T is the number of words in that story and V is the vocabulary size. We store these matrices in a dictionary keyed by story name.

3.1.2 Downsampling and Trimming for Alignment

Since our goal is to use the embedding matrices to predict the fMRI responses, we need the embedding matrix X to have the `same number of rows` as the fMRI response matrix Y, where each row corresponds to a TR (temporal resolution unit of the fMRI scan). Initially, the BoW matrix has one row per word token, which is much more fine-grained than the TR-level sampling used in Y.

To align the two, we downsample the embedding matrix so that it matches the number of TRs in the response matrix. Conceptually, this means converting fast, high-resolution text (every word) into slow, low-resolution brain signals (every TR). We achieve this using the provided `downsample_word_vectors` function, which applies Lanczos interpolation. This method computes a weighted average of the nearby word vectors for each TR time point, using a smooth filter. This helps reduce noise and resamples the word-level features to match the slower timing of the fMRI data.

The function takes in the original embedding matrix (with one row per word token), the word timing information (`data_times`), and the TR timing information (`tr_times`), and outputs a matrix with *one row per TR*.

After interpolation, we further trim the first 5 seconds and the last 10 seconds of the resulting matrix. This step is consistent with fMRI preprocessing conventions, where the initial and final segments of the scan may not reflect meaningful neural activity due to scanner warm-up or post-story silence. Once trimmed, the embedding matrix X is now temporally aligned with Y.

3.1.3 Lagged Feature Construction

To account for the fact that the BOLD fMRI signal lags behind the stimulus, we construct lagged versions of the downsampled embedding matrices using the provided `make_delayed` function, with delays ranging from 1 to 4 TRs. The idea is that if a word is heard at time t, the associated neural response may appear several seconds later, typically 2–3 TRs afterward.

This lagging process works by augmenting each row (each TR) with the embedding vectors from the previous 4 TRs. That is, when estimating the brain response at a given TR, we do not use the information from that TR itself, but instead use the features from the `four preceding TRs`. This is like *"pushing down"* the embedding matrix so that each row is a concatenation of the past 4 rows.

As a result, the number of columns in the embedding matrix increases by a factor of 4. For example, if the original embedding had d features per TR, the lagged version will have $4d$ features. The final lagged embedding matrices are now temporally aligned with Y and ready to be used for ridge regression modeling.

3.2 Word2Vec

Word2Vec (W2V) is a pre-trained word embedding model that maps words into continuous vectors, capturing semantic similarity and contextual relationships between words. We use the publicly available Google News Word2Vec model, which was trained on approximately 100 billion words. The model acts like a dictionary: each key is a word and the corresponding value is a dense 300-dimensional vector.

This is fundamentally different from Bag-of-Words (BoW), which represents each word as a `sparse one-hot vector` with no sense of similarity or context between words. In contrast, Word2Vec embeddings `capture richer semantic relationships`—similar words have similar vectors.

Before creating embeddings, we check which words in our training set are included in the W2V vocabulary. About `5%` of the unique words in our training stories are not found in the W2V model. Since this is a relatively small fraction, we assign a zero vector to those words during embedding construction.

We then repeat the same procedure used in the BoW pipeline:

- Generate an embedding matrix per story by mapping each word token to its corresponding W2V vector (or a zero vector if not found).
- Downsample the word-level matrix to TR-level resolution using the `downsample_word_vectors` function.
- Apply lagged feature construction using `make_delayed` with delays $[1, 2, 3, 4]$, so each row includes information from the four previous TRs.

The final embedding matrix for each story has shape $(T', 1200)$, where T' is the number of TRs in the story and $1200 = 300 \times 4$ is the lagged feature dimension.

3.3 GloVe

GloVe (Global Vectors for Word Representation) is another widely used pre-trained word embedding model, developed by researchers at Stanford in 2014. Unlike Word2Vec, which learns word meanings by predicting a word based on nearby words, GloVe learns by looking at *how often words appear together* across the entire dataset. This helps GloVe capture both local context and overall patterns in language, allowing it to represent the relationships between words more effectively.

We use the publicly available `glove.6B.300d.txt` model, which provides 300-dimensional embeddings for a large vocabulary trained on 6 billion tokens from Wikipedia and Gigaword. Since the GloVe file is distributed as a plain text file, we *manually convert it into a dictionary* where the keys are words and the values are 300-dimensional float vectors.

As with Word2Vec, only a small number of words from our training data are missing in the model, and we assign zero vectors to those cases.

After this, we follow the same embedding pipeline as in Word2Vec: creating an embedding matrix per story, downsampling it to TR resolution, and applying lagged feature construction with delays $[1, 2, 3, 4]$. The resulting embedding matrices have the same shape as in W2V: $(T', 1200)$, where T' is the number of TRs for each story.

3.4 Comparison of Embedding Methods

Each embedding method we explored—Bag-of-Words (BoW), Word2Vec (W2V), and GloVe—has distinct strengths and limitations in the context of modeling fMRI responses to language.

BoW represents each word as a sparse one-hot vector, meaning that even semantically related words like "king" and "queen" have completely unrelated vectors. This lack of contextual or semantic structure limits BoW's ability to generalize. Moreover, BoW results in a very high-dimensional feature space: the number of columns in the matrix grows with the size of the training vocabulary, which changes depending on the dataset. This leads to computational inefficiencies and sparsity, which can be problematic for matrix operations like inversion in regression models. However, BoW is `highly interpretable`. For instance, if the coefficient in the 4th row and 5th column of the fitted β matrix is relatively large, we can directly conclude that the 4th word from lag 1 has a strong predictive influence on the response at the 5th voxel. This level of interpretability is a key advantage of BoW.

W2V and GloVe, in contrast, map each word to a fixed-length dense vector (300 dimensions), regardless of the dataset. These vectors are pre-trained on large corpora and encode semantic structure where words that appear in similar contexts are placed close together in the embedding space. As a result, models using these embeddings can generalize better across stories and capture deeper linguistic relationships. For example, both models famously support analogical reasoning: vector arithmetic such as `king - man + woman = queen` holds due to the way these embeddings capture semantic and gender relationships. However, the dimensions of W2V and GloVe embeddings are not individually interpretable, so we cannot attribute a single coefficient back to a specific word or linguistic feature in the way we can with BoW.

In summary, BoW excels in interpretability but suffers from sparsity and high dimensionality, while W2V and GloVe offer semantically compact representations at the expense of interpretability.

4 Modeling & Evaluation

To model brain responses recorded via fMRI, we define a function that maps language stimuli to voxel activity. Given the limited size and noisy nature of fMRI data, generic non-linear approximators perform poorly as encoding models. Instead, we use the linear model:

$$\hat{R} = g(S)\beta$$

where $\hat{R} \in R^{T \times V}$ is the predicted voxel activity, S is a continuous string of words in natural language, and $g(S)$ is the embeddings we get. We use Ridge regression to estimate β. To evaluate model performance for

a single voxel, we will compute the Pearson correlation coefficients between real response R and predicted response $\hat{R}$. Then we repeat it for all voxels.

4.1 Model Fitting - Ridge Regression

We stacked embeddings of all stories in our train set to get the train design matrix, and response file to get response matrix. We start by manually setting regularization parameter α to be 1 for all voxels and compare mean correlation coefficient for 3 different embeddings (Table 1).

Embedding	**Mean CC**
Bag-of-Words	0.00216
GloVe	0.00592
Word2Vec	0.00499

Table 1: Mean Correlation Coefficient for Different Embeddings

We can see that when alphas are poorly set, the mean CC across voxels are very low. Besides, the mean CC for bag-of-words embedding is significantly lower than other two embeddings.

4.2 Modeling and Cross-Validation Procedure

4.2.1 W2V & GloVe

To evaluate and tune our ridge regression models, we used a 5-fold cross-validation (CV) procedure on the 70 training stories. In each fold, we split the data into 56 training stories (80%) and 14 validation stories (20%). This was repeated five times, allowing each story to appear in the validation set exactly once.

For each fold, we computed a matrix of correlation coefficients (CCs) across a range of regularization strengths (ridge penalties), using the `ridge_corr` function. Initially, we tested 10 logarithmically spaced α values from 1 to 1000. However, we observed that the highest mean CCs tended to occur at the upper end of this range, suggesting that stronger regularization might improve performance. As a result, we expanded the α range to span from 10 to 10000.

After completing cross-validation, we averaged the correlation matrices across the five folds to obtain a mean performance *for each alpha and voxel*. For each voxel, we selected the alpha value that yielded the highest mean CC across folds. This gave us a vector of best α values (one per voxel) to use in final model training.

Using these tuned alpha values, we fit the model using all 70 training stories and evaluated it on the test stories. This yielded a vector of test correlation coefficients across all voxels. To summarize overall performance, we reported several metrics: mean test CC, median test CC, and the top 1st and 5th percentiles of test CCs across voxels.

4.2.2 BoW

The delayed matrix of BoW embedding has significantly more columns than the other two embeddings (41440 compared to 1200), therefore, to reduce computational cost, we implemented a 3-fold CV for BoW compared to 5-fold CV for the other two embeddings. Additionally, we selected only 2 candidate α values for BoW compared to 10 for the other two embeddings.

Due to the large size of the delayed matrix, the algorithm failed to converge and raised a `LinAlgError` during the SVD decomposition step in the ridge_corr_pred function. To address this issue, we used a try-except block to catch the error and switch to `scipy.sparse.linalg.svds`, which reduces the computational load while still approximating the SVD decomposition of the whole matrix (41440 columns) effectively.

4.2.3 Results (CC) Comparison

Table 2 and 3 show the results of test CCs for the each subject. While Word2Vec and GloVe both outperformed Bag-of-Words across all evaluation metrics, their performance was remarkably similar. For Subject 2, GloVe slightly edged out Word2Vec, whereas for Subject 3, Word2Vec performed marginally better. This suggests that the differences between these two embeddings are subtle and may depend on the individual subject's brain response patterns rather than the embeddings themselves.

Embedding	Mean CC	Median CC	Top 1 Percentile CC	Top 5 Percentile CC
Bag-of-Words	0.00421	0.00372	0.03648	0.02444
GloVe	0.01369	0.01094	0.07089	0.04824
Word2Vec	0.01298	0.01027	0.07103	0.04782

Table 2: Cross Validation Results for Subject 2 (Test CC)

Embedding	Mean CC	Median CC	Top 1 Percentile CC	Top 5 Percentile CC
Bag-of-Words	0.00490	0.00398	0.04508	0.02705
GloVe	0.02029	0.01636	0.09529	0.06236
Word2Vec	0.02081	0.01686	0.09457	0.06346

Table 3: Cross Validation Results for Subject 3 (Test CC)

Given these results, we conclude that both GloVe and Word2Vec are viable options. Their similarity in performance reflects overlapping properties in how they capture semantic similarity, despite differences in how the embeddings were generated.

When averaging results across both subjects, GloVe had a slightly higher overall performance in terms of mean, median, and top percentile CCs, so we select GloVe as the best-performing model for our final analysis.

4.3 Model Evaluation

We chose GloVe as our best embedding. The histogram of CC for GloVe (Figure 1) reveals a right-skewed distribution, where the vast majority of voxels have CCs close to zero, with a slight spread into the negative range. A long right tail indicates a small subset of voxels achieves relatively high prediction accuracy, while the majority remain poorly predicted.

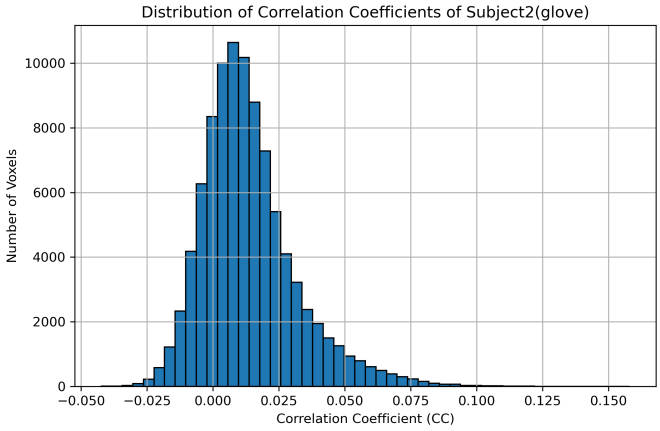


Figure 1: Distribution of CC for Subject 2 using the GloVe embedding.

4.4 Model Performance

Only a small subset of voxels is well-modeled, while the majority shows very small correlation. According to the NeurIPS 2018 paper, GloVe embeddings—like other static word embeddings—lack contextualization and treat each word independently. While GloVe captures semantic similarity through co-occurrence, it cannot represent temporal or syntactic context, which is a critical component of natural language understanding in the brain. This explains the observed histogram: most voxels show weak or no predictability with GloVe because context-independent features fail to align with brain areas sensitive to discourse structure, sentence-level meaning, or narrative coherence. However, the long right tail of the distribution indicates that some regions—likely those involved in basic lexical-semantic processing are still effectively modeled by GloVe.

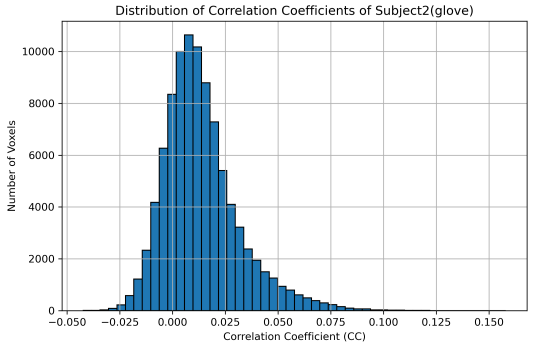
A reasonable interpretation criterion under the PCS framework is we can only interpret a small proportion of well predicted voxels(for example, voxel with CC above a threshold like 0.05). Besides, we choose Ridge regression and Glove to make the model computable. And we will conduct a stability check in the next section.

4.5 Stability analysis

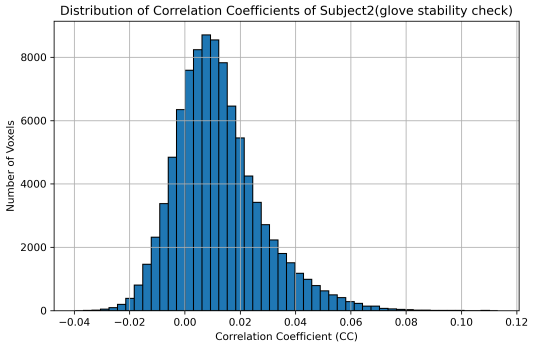
Previously, we fit the Ridge regression model on subject 2. To perform a stability analysis, we reshuffled all stories and fit GloVe embeddings on a new train/test split. Besides, we also fit the model on subject 3. Based on results shown in Table 4 and Figure 2, subject 3 shows better performance than subject 2 in general, but their distributions' shape are quite similar. Train/test set variation has a minor impact on overall model performance. The consistency of CC distribution shapes across different splits and different subjects suggests that although GloVe does not exhibit powerful predictability, it shows stability and robustness to perturbations.

Data	Mean CC	Median CC	Top 1 Percentile CC	Top 5 Percentile CC
Subject 2 (split 1)	0.01369	0.01094	0.07089	0.04824
Subject 2 (split 2)	0.01217	0.01017	0.05948	0.04194
Subject 3 (split 1)	0.02029	0.01636	0.09529	0.06236
Subject 3 (split 2)	0.01917	0.01567	0.08326	0.05830

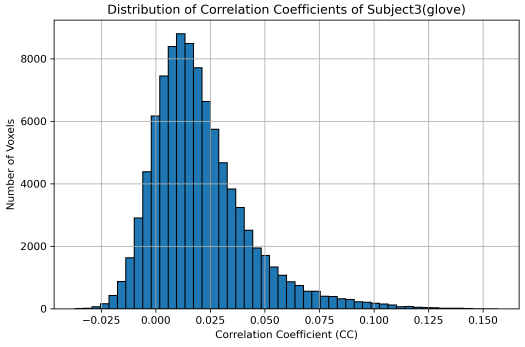
Table 4: Stability Check Results (GloVe Embeddings)



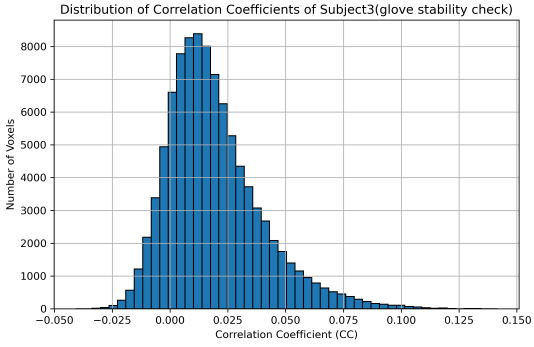
(a) Subject 2 (original split)



(b) Subject 2 (new split)



(c) Subject 3 (original split)



(d) Subject 3 (new split)

Figure 2: Distribution of CC across different subjects and different test stories

5 Conclusion

In this lab, we evaluated three types of static word embeddings—Bag-of-Words, Word2Vec, and GloVe—for predicting fMRI brain responses to naturalistic story listening. While both Word2Vec and GloVe outperformed Bag-of-Words slightly in terms of mean correlation coefficients (CCs), all methods produced generally low CCs, with most values clustered near zero and only a small subset of voxels showing any predictive signal. This suggests that static embeddings may be insufficient to capture the full complexity of linguistic processing in the brain, particularly due to their inability to model context, syntax, and temporal structure. Future work could explore the use of contextualized language models such as BERT or GPT, which dynamically adjust word representations based on context and may align more closely with how the brain encodes language.

6 Bibliography

References

[1] Shailee Jain and Alexander Huth. Incorporating context into language encoding models for fmri. *Advances in neural information processing systems*, 31, 2018.

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.