Context-Enhanced Feature Construction Using Sentences

1. Motivation

The original EEG P300 dataset contains brain responses for individual characters only.

It does not include:
- -> Word structure
- -> Sentence structure
- -> Language context
- -> Sequential dependencies between characters

However, in real language:
- -> Characters are not independent
- -> The probability of a character depends on previous characters

To address this limitation, we introduce language context by combining:
- -> EEG-based features
- -> Contextual probabilities from a language model (LSTM)

This creates a hybrid feature representation that improves character prediction.

2. High-Level Overview

After completing EEG preprocessing, we perform the following steps:

2.1 Generate 400 sentences
2.2 Process each sentence character by character
2.3 For each character:
  Attach an EEG chunk (30 windows)
  Attach a context probability window (1 window)
2.4 Combine both into a single feature vector

Final result: Each character is represented by 31 windows instead of 30

3. Sentence Generation

3.1 We generate 400 sentences, for example:
  THE QUICK DOG JUMPED OVER
  BLUE WATER FLOWED DOWN GENTLY
  EIGHT CHILDREN RAN THROUGH MEADOW
  FIND THE DARK PATH AHEAD

3.2 These sentences:
  Are not used during language model training (I mean the LSTM model not the CNN model, It will later be used to train CNN)
  Are used only for evaluation and feature construction

3.3 EEG Feature Source (This point is mentioned here only to remind you of the result of the previous document.)

From the previous preprocessing document: EEG data is stored in a container with 36 buckets
Each bucket corresponds to one P300 character: A-Z, 1-9, _
Each bucket contains 4–8 EEG chunks
Each chunk represents one full character trial
EEG Chunk Structure
  EEG Chunk
  ├── Window 1 → (78 × 64)
  ├── Window 2 → (78 × 64)
  ├── ...
  └── Window 30 → (78 × 64)

4. Language Model (Context LSTM Model)

Model Type:
  -> LSTM-based Character Prediction Model
  -> Trained on book text data
  -> Learns how characters follow each other

Goal
  Given a prefix (incomplete sentence), predict:
  "What is the probability of each possible next character?"

5. How the Language Model Is Trained (Simple Explanation)

The model learns by reading text character by character.

Training process:
1. Take a short sequence of characters (example: "HELLO EV")
2. Ask the model to predict the next character
3. Compare prediction with the real next character
4. Adjust weights to reduce error
5. Repeat for many epochs and many sequences

Over time, the model learns:
1. Common letter patterns
2. Word structure
3. Language rules

6. Language Model Prediction Examples

Example 1
  Input: "T"
  Output: probabilities (simplified): {
                H → 0.956
                I → 0.009
                A → 0.004
                ...
              }
  Prediction: T → TH

Example 2
  Input: "ORANG"
  Output: probabilities (simplified): {
                E → 0.791
                A → 0.005
                B → 0.002
                ...
              }
  Prediction: ORANG → ORANGE

7. Language Model Evaluation

To evaluate the model:
  Use the 400 generated sentences
  The model has never seen them before

Testing Procedure

For each sentence:
  Start with the first character
  Predict the next character
  Extend the prefix
  Repeat until sentence ends

Example:
  Input: T
  Input: TH
  Input: THE
  Input: THE Q
  ...

Evaluation Rule:
  Prediction is considered correct if: The true next character appears in the top 3 probabilities

Result:
  Next-character prediction accuracy: 64.95%

8. Context-Enhanced Feature Construction

Now we combine EEG data and language context.

For all 400 sentences:
1. Loop through sentences
2. Loop through characters
3. Skip spaces
4. For each character:
  Select EEG chunk
  Generate context probabilities
  Convert probabilities to (78 × 64)
  Append to EEG windows
5. Store result

Each processed entry contains:
  Character
  Prefix Sentence
  Position in sentence
  EEG + context feature vector (31 window)

9. Detailed Construction of the Context Probability Window and EEG Chunk Selection

This section explains in detail how:
  The language model output is produced and formatted
  The probability output is converted into a (78 × 64) window
  EEG chunks are selected from the character-specific buckets

9.1 Language Model Output Format (JSON Probabilities)
  For a given prefix (incomplete sentence), the LSTM language model outputs
  a probability distribution over the full P300 character set.

  Character Vocabulary
    The vocabulary contains 36 symbols:
    A-Z, 1-9, _

  Model Output
    The output is a JSON-like dictionary:
    {
      "A": p(A),
      "B": p(B),
      "C": p(C),
      ...
      "_": p(_)
    }

    Each value represents the probability that this character is the next one, given the sentence prefix.

    Example output for prefix "THE QUICK DO":
    {
      "G": 0.87,
      "S": 0.04,
      "N": 0.03,
      ...
    }

    This probability distribution encodes language context but does not yet match the EEG window format.

9.2 Ordering and Vectorization of Probabilities
  To align the language model output with EEG data, the probabilities are first converted into a fixed-order vector.

  Step 1 — Vocabulary Ordering
    The characters are ordered according to a predefined list: [A, B, C, ..., Z, 1, 2, ..., 9, _]

  Step 2 — Probability Vector Creation
    The JSON dictionary is mapped into a numeric vector: [p(A), p(B), p(C), ..., p(_)]
    Result: Probability Vector Shape = (36,)
    This guarantees:
      Fixed ordering
      Fixed dimensionality
      Consistency across all samples

9.3 Expanding the Probability Vector to Match EEG Window Shape
  EEG windows have shape: 78 × 64
  To match this format, the probability vector is expanded as follows:

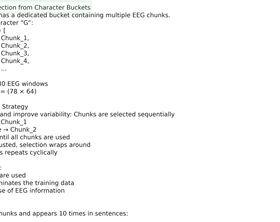
  Horizontal Expansion
    The 36 probabilities are copied to form:
    [p(A), p(B), p(C), ..., p(_), p(A), p(B), p(C), ..., p(_)]

    Then the vector is padded with zeros to reach the required length:
    [p(A), p(B), p(C), ..., p(_), p(A), p(B), p(C), ..., p(_), 0, 0, 0, 0]

    Result: Probability vector shape = (78,)

  Vertical Expansion
    The resulting probability vector of length 78 is repeated 64 times.
    This repetition creates a 2D matrix where:
      Each column represents one EEG channel
      Each column contains the same probability values across time

  Final result:
    Probability window shape = (78 × 64)



9.4 EEG Chunk Selection from Character Buckets
  Each character has a dedicated bucket containing multiple EEG chunks.
  Example for character "G":
    Bucket["G"] = [
                  Chunk_1,
                  Chunk_2,
                  Chunk_3,
                  Chunk_4,
                  ...
                ]
    Each chunk: 30 EEG windows
    Each window = (78 × 64)

  Chunk Selection Strategy
    To avoid bias and improve variability: Chunks are selected sequentially
      First use → Chunk_1
      Second use → Chunk_2
      Continue until all chunks are used
      Once exhausted, selection wraps around
      The process repeats cyclically

  This guarantees:
    All EEG trials are used
    No chunk dominates the training data
    Balanced reuse of EEG information

  Example
    If "G" has 4 chunks and appears 10 times in sentences:
    Usage order:
    G_1 → G_2 → G_3 → G_4 → G_1 → G_2 → G_3 → G_4 → G_1 → G_2

9.5 Final Feature Construction for One Character
  For character "G":
    Select one EEG chunk → 30 windows
    Generate probability matrix → 1 window
    Append probability window to EEG chunk

  Final feature tensor:
    Total windows = 31
    Window shape = (78 × 64)

  So: Feature("G") = 31 × 78 × 64

  EEG Chunk for "G"
  ├── EEG Window 1 (78 × 64)
  ├── EEG Window 2 (78 × 64)
  ├── ...
  ├── EEG Window 30 (78 × 64)
  └── Context Window (78 × 64) ← from LSTM probabilities

10. Final Data Structure (Visualization)

Processed Data
  │
  ├── Sentence 1
  │   ├── T → 31 windows
  │   ├── H → 31 windows
  │   ├── E → 31 windows
  │   └── ...
  │
  ├── Sentence 2
  │   ├── B → 31 windows
  │   ├── L → 31 windows
  │   └── ...
  │
  └── Sentence 400