Adaptive Language Modeling and Predictive Text Optimization for a Brain–Computer Interface Keyboard

In this project, students will design and implement an adaptive language model (LM) to enhance typing efficiency and contextual prediction in a brain–computer interface (BCI)–controlled on-screen keyboard.
The LM will predict probable letters or words based on prior input and dynamically adjust the visual interface—such as key size, highlighting, or flashing order—to guide user attention toward the most likely selections.
This system simulates how a real BCI keyboard assists individuals with limited motor function by combining probabilistic modeling, contextual learning, and user interface adaptation.

Learning Objectives

By completing this project, students will:

1. Understand the principles of probabilistic and neural language modeling (e.g., n-gram and RNN-based models).
2. Implement an adaptive prediction engine that updates unigram and bigram probabilities in real time.
3. Develop algorithms to assign probability scores to each key and dynamically reweight them based on linguistic context.
4. Design a user interface that visually emphasizes high-probability keys and adapts to user input patterns.
5. Evaluate performance using metrics such as typing speed, prediction accuracy, and information transfer rate (ITR).

Technical Tasks

1. **Model Development:**
    - Implement a probabilistic (n-gram) language model using Python.
    - Train the model on an English text corpus (e.g., movie subtitles, news articles).
    - Initialize statistical priors (base word or letter frequencies).

2. **Adaptive Updating:**
    - Implement real-time updates to unigram and bigram probabilities after each user selection.
    - Enable the LM to increase the probability of contextually relevant letters (e.g., "e" after "th").
3. **Interface Integration:**
    - Build a simple GUI keyboard using Python (Tkinter, PyQt, or web-based front end).
    - Dynamically reweight key appearance (e.g., enlarge, brighten, or reorder high-probability keys).
4. **Performance Evaluation:**
    - Measure improvements in typing speed, prediction accuracy, and cognitive workload.
    - Compare static and adaptive LM performance.

Tools and Technologies

- Programming Language: Python
- Libraries: NLTK or spaCy (for text preprocessing), NumPy/Pandas (for data handling), PyQt/Tkinter (for GUI), Matplotlib (for visualization)
- Optional: TensorFlow or PyTorch for neural language modeling (advanced students)

Deliverables

1. A working adaptive on-screen keyboard application.
2. A report explaining the LM design, update logic, and user interface adaptation strategy.
3. A short presentation demonstrating the system and comparing performance metrics.

Expected Outcomes

By the end of this project, students will have a working prototype of an adaptive predictive text system that demonstrates how statistical learning and real-time interface adaptation can significantly improve communication efficiency in assistive technologies. The project provides hands-on experience with machine learning, human–computer interaction, and accessibility-centered design — key skills in modern intelligent system development.