BCI: 2 button keyboard

Non-invasive general purpose brain-computer interface

Product Requirements Document

Authors/Contributors: Ethan Grabau

TL;DR:

We are building a non-invasive brain-computer interface that will allow users to interact with their digital world and send messages all private, hands-free, and voice-free. This input will be as fast or faster than keyboard typing.

Background:

With advancements in LLM's, there seems to be a prime opportunity to introduce a novel BCI (Brain Computer Interface) that could become the next mainstream input method for consumers.

What is unique about this moment?

With the advent of LLM's and transformer models, there is an opportunity to make the best test prediction ever that is highly personalized to a user.

There is a theory that "final interfaces" will look like a big "next" button that a user clicks again and again, accurately predicting what the user wants to do, and allowing them to edit the input if it's not what they want. With LLM's that are trained on users' personal context (ie chat history and emails, etc) there is a real opportunity to create something like this final interface. We can start with just text prediction when users are typing.

Why brain interface?

- Typing is good when your hands are available, but users often find themselves either temporarily and permanently impaired (ie one or both hands occupied, or eyes need to be up and available for safety).
- Voice is likely going to become more common with conversation assistants though when there is a lot of back and forth with agents, but many customers are not comfortable speaking out loud.
- The brain-computer interface is a sort of “holy grail” of interface design.

Haven’t others tried this already?

- Yes, but I think they are taking an approach that might be overly challenging. They are trying to decode actual thought, which is probably the real holy grail, but likely unachievable in the short term. (ie low precision due to extensive phonics that need to be detected by a system)

What is novel here?

- I am proposing a system that only relies on a very limited set of inputs. My assumption is that it would be “easy” to build a consumer-friendly hardware device that could get anywhere from 2-4 brain “inputs”. My rationale is that there are even low-cost consumer devices that detect brain-waves for control/input on a binary basis.

Solution Proposal:

We build a keyboard that only has 2-4 buttons. The left side of the keyboard, the right side of the keyboard, maybe a space bar/ enter bar and maybe a clear/backspace button.

We train an LLM / transformer-based model to create the best auto-prediction (fine tuned or trained on a user’s messaging history - and re-trained over time to include new concepts, words, proper nouns, etc). That makes next work prediction highly accurate. Then the L/R side keyboard input along with the LLM predictions make a truly magical, highly accurate, low latency input method.

After we build a two-button keyboard that can do this, and validate that it is highly usable, we then we build a device that can detect those 2-4 inputs from the brain.

Then you get a set of earbuds and soon you are texting your partner, in your voice, and having a two-way conversation back and forth as if it's like mind-reading.

I'm assuming that once a user learns the two-button input method, then it will become highly easy to remember and use without a physical keyboard.

We can always have a configurable software option selection for the user to default to a normal keyboard style to input new words that can't be predicted, so they keyboard can be adaptable.

We can even consider building our own messaging app if required to facilitate robust LLM integration that can do some pre-processing prior to user engagement (if desirable).

After we do this for chat use-cases, we can consider other use cases like chat with LLM agents, and open-ended search and command/control use cases.

Key Principles

The following principles should be followed in the design:

- Highly accurate (assuming people won't keep using if it's not accurate)
- Low latency - as fast or faster than keyboard typing ideally
- Manual backup modes (ie full keyboard) for users to be able to complete the task if prediction is insufficient (which we know it will be eventually)
- Ideally takes into account as much context as possible