where a is the suggestion made by the agent (an empty string if no suggestion was made), and "accepted" is a binary indicator for whether the user accepted the suggestion. We estimate $\Delta t_{distraction}$ by two saccada times for the task of natural reading (Rayner & Clifton, 2009), which amounts to $\Delta t_{distraction} = 2 \times 30$ ms $= 60$ms. We estimated the per-character writing time and reading time to be 521ms and 40ms respectively, based on an internal dataset in our target use case of EMG handwriting (CTRL-labs at Reality Labs et al., 2024). This leads to $\alpha = \frac{40}{521}, \beta = \frac{60}{521}$.

B Further Details on Computational Experiments

Our PPO and DQN agents are built off the CleanRL implementations (Huang et al., 2022), while our IQL agent is built off the CORL implementation (Tarasov et al., 2022). For PPO and DQN, we use a per-action Q-network architecture that takes as input the state and a single candidate suggestion (or `wait`), and produces the corresponding Q-value (and similarly the action probability for the PPO policy). To obtain the best action, we simply choose the candidate suggestion or `wait` with highest Q-value (or with highest action probability for PPO); note that this requires running $k+1$ forward passes through the model, one per k candidate suggestions plus one for `wait`. Both PPO and DQN used the `distilbert-base-uncased` pre-trained model from HuggingFace[3], where the Q-function and the PPO policy network are implemented as one-layer projection MLPs on top of the distilbert transformer. We did *not* freeze any layers of the transformer; initial experiments found that only training the projection layer of the Q-function network and using distilbert as a frozen feature extractor worsened DQN's performance. IQL used a multi-layer perceptron (MLP) with two hidden layers of size 256 each, and $k+1$ output nodes, one for each of the k candidates and `wait`. To obtain the best action, we take the action corresponding to the output node with maximum value. All RL agents were trained for 250K steps.

The default hyperparameters for the RL agents are as follows.

PPO: learning rate 10^{-6}, number of steps per iteration 20, number of optimization epochs 1, discount factor 0.99. All other hyperparameters were unchanged from the CleanRL defaults.

DQN: learning rate $3 \cdot 10^{-7}$, target network update interval 10000, learning start iteration 25000, discount factor 0.99, model update frequency 10, batch size 32, exploration ϵ linearly decaying from 1.0 to 0.05 over the first half of training. All other hyperparameters were unchanged from the CleanRL defaults.

IQL: learning rate $3 \cdot 10^{-4}$, batch size 256, discount factor 0.99, $\tau = 0.7$. All other hyperparameters were unchanged from the CORL defaults.

C Multi-Word Suggestion Length Normalization

In our multi-word suggestions experiment, we generated multiple LM candidates via beam search. Since the joint likelihood of suggesting two words is always lower than the likelihood of suggesting just the first, i.e., $P(\text{second word} \mid \text{first word}) \times P(\text{first word}) < P(\text{first word})$, we normalized the joint probability of each 2-word suggestion by taking the square root to counteract this effect, similar to Murray & Chiang (2018). We note that in the multi-word suggestions experiment, we considered both 1-word and 2-word suggestions, not just the 2-word ones. Hence, for all suggetions, we universally replace the raw probability from the LM, denoted as P, by the normalized probability, denoted as P'. Concretely, the normalized probability P' for a general m-word suggestion $w_{1:m}$ after the entered partial sentence w_{prev} is

$$P'(w_{1:m} \mid w_{\text{prev}}) = \left(\prod_{i=1}^{m} P\left(w_i \mid w_{1:i-1}, w_{\text{prev}}\right) \right)^{\frac{1}{m}}.$$

[3] https://huggingface.co/distilbert/distilbert-base-uncased

Here, w_i refers to the ith word. We can see that when $m = 1$, we simply have $P'(w_1 \mid w_{\text{prev}}) = P(w_1 \mid w_{\text{prev}})$, and hence no length normalization is applied.

Here is a real example from our LM in the multi-word suggestions experiment. For the context `finally, we have a third val`, the LM generated three candidate completions: `entine's day`, `ue`, and `entine's`. After length normalization, the normalized probabilities were {`entine's day`: 33%, `ue`: 32%, `entine's`: 17%}. This would be impossible without length normalization, as the joint probability of `entine's day` could never exceed the probability of `entine's`, but it is clear that `entine's day` is a better completion for the stated context.

D Why Does Incorporating 2-Word Suggestions Harm the Oracle?

In our initial experiments, we were surprised to find that incorporating 2-word suggestions *harms* the performance of even the oracle agent. Recall that the oracle agent has privileged access to the target sentence the user is trying to write, and therefore will never make a wrong suggestion that gets ignored by the user. Upon probing further, we discovered that this is due to several situations where 2-word candidates take up the slot of the correct 1-word candidate, preventing it from being one of the k candidates seen by the agent. Here, we provide a real example from our experiments.

Recall that our 2-word suggestions experiment used $k = 5$, i.e., the agent picks from the top-5 candidate completions from the LM with the highest probabilities (together with `wait`). For the context `i am gr`, where the user's target sentence is `i am great, how are you?`, the five candidate completions from our LM, in order of descending probability, are: `ateful`, `ateful for`, `ateful to`, `eat for`, and `eat with`. None of these matches the remaining target sentence, so the oracle agent does not surface any suggestion to the user, and instead chooses to `wait` with a reward of 0. However, the LM's sixth completion turned out to be `eat`, which matches the remaining target sentence. Hence, if we were still in the 1-word suggestion scenario, the unmatched 2-word suggestions (`ateful for`, `ateful to`, `eat for`, and `eat with`) would not be present, and the oracle agent would have received `eat` as one of its $k = 5$ options. The oracle then would have surfaced this suggestion to the user and received positive reward.

E Cognitive Load User Study - Extended Results

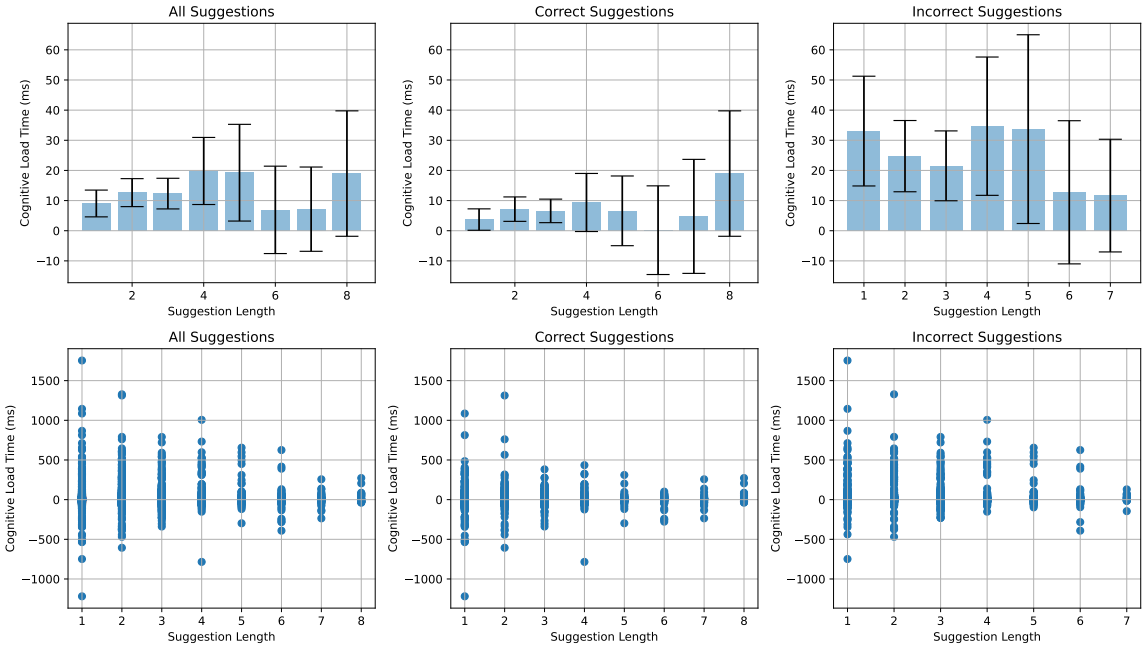

Figure 6: Detailed results of the cognitive load user study based on $N = 9$ subjects. The top row shows the average and 95% confidence intervals, while the bottom row shows all the actual datapoints. The three columns from left to right represent the average cognitive load across: 1) all suggestions, 2) correct suggestions only, and 3) incorrect suggestions only.

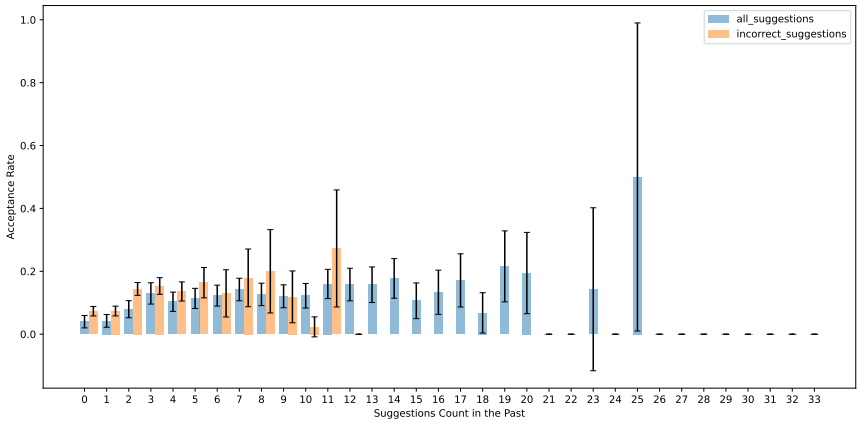

Figure 7: User acceptance rate as a function of the total number of past a) suggestions (blue) and b) incorrect suggestions (orange). The data did not show any evidence of declined suggestion acceptance rate based on the cumulative number of past suggestions, or the cumulative number of past incorrect suggestion. Therefore, we did not find evidence for the accumulated fatigue hypothesis. Error bars depict 95% confidence interval.