6 Experiments

6.1 Experimental Setup

We sampled target sentences for the user from two open-source datasets, SMS Spam Collection ("ham" labels only) (Almeida et al., 2011) and Reddit Webis-TLDR-17 (Volske et al., 2017). Applying the following filters led to ∼500 and ∼300,000 sentences in each dataset respectively:

- Use only sentences with length ≤ 10 words to save compute.
- Use only sentences with no profanity and whose perplexity is under 150 as computed by GPT2-Large (Radford et al., 2019), to ensure language quality (e.g., no misspellings or slang).
- Remove sentences with characters not in the set "a-z,.'?! " to satisfy the LM input requirements.

For our language model (LM), we trained a small (∼25MB) transformer (Vaswani et al., 2017) on the Wikipedia[2] and Reddit Webis-TLDR-17 corpora. This transformer outputs a probability distribution over the English vocabulary. To reduce down to the k candidates fed into our RL agent, we take the top k outputs and renormalize their probabilities. Unless otherwise specified, our experiments only consider single-word autocompletion (as opposed to multi-word).

For all simulations, we assume the user is the idealized one described in Section 4.

6.2 Computational Experiments

We explored three RL algorithms: PPO, an online policy gradient method (Schulman et al., 2017); DQN, an online value-based method (Mnih et al., 2013); and IQL, an offline method (Kostrikov et al., 2021). We benchmark these algorithms against three baseline agents: oracle, uniform random, and threshold-based. The oracle agent has privileged access to the target sentence and always surfaces correct suggestions when given by the LM; it thus establishes an upper bound on performance. The threshold-based agent surfaces the top suggestion if and only if its probability is above a fixed threshold. Optimization on the validation set led us to use threshold=0.3. For PPO and DQN, we used DistilBERT (Sanh et al., 2019) as the policy architecture and initialization, while IQL used a multi-layer perceptron (MLP) trained from scratch. See Appendix B for more details.

The IQL policy received the suggestion probabilities from the LM as part of its input, while PPO/DQN policies did not. This design choice was made because probabilities from the LM are floating-point numbers and it is not useful to tokenize them as strings to feed into transformers. Future work may consider learning or constructing an input representation for these probabilities, e.g., using an embedding layer (Chen et al., 2021) or predefined activation (Gorishniy et al., 2022).

All RL algorithms were trained for 250K gradient steps with $\gamma = 0.99$. Since IQL is an offline RL algorithm, we trained it on a static dataset of 5,000 trajectories collected by a policy that follows the threshold-based baseline but with a 5% chance of taking a random exploratory action each step.

First, we considered the $k = 1$ setting, where the LM generates one candidate and the action space is $\{suggest, wait\}$. We used the default reward function parameters $\alpha = 40/521$, $\beta = 60/521$. Figure 3 shows the average return and number of saved characters in our two domains. In both cases, IQL performed best among RL techniques (9.37 ± 0.17 average return), but could not improve significantly over the threshold-based agent (9.27 ± 0.18). Interestingly, PPO was overly aggressive: it saved the user the most characters, yet it obtained a lower return due to more incorrect suggestions.

These observations made us question whether our autocomplete problem benefits from sequential decision-making, or if a contextual bandit suffices. To answer this, we reran the RL agents under the same MDP but with $\gamma = 0$. Results are shown in Figure 4 (left two plots). Theoretically, $\gamma = 0.99$ renders the problem harder, yet the solution asymptotically should be better or the same as when $\gamma = 0$. For IQL, the results with both γ values were on par. However, for PPO, we observed better results with $\gamma = 0$, though as we increased the number of training steps to 1M, the difference

[2] https://huggingface.co/datasets/wikipedia

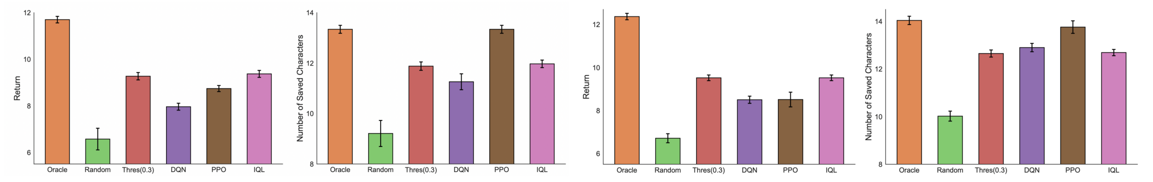
Figure 3: Average return and number of characters saved, over 5 independent runs (including training, for RL agents). Error bars depict 95% confidence interval. *Left two:* SMS dataset, 50 sentences in evaluation set. *Right two:* Reddit Webis-TLDR-17 dataset, 400 sentences in evaluation set.

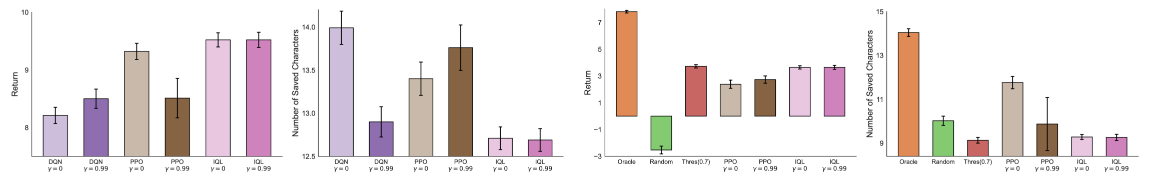
Figure 4: *Left two plots:* Analysis of how varying γ affects RL agents. *Right two plots:* Results of rerunning experiments with $\alpha = 0.4$, as suggested by our theory (Section 5). All plots use the Reddit Webis-TLDR-17 dataset and show the same metrics (y-axis) as in Figure 3. In the right two plots, DQN results are omitted because they were significantly worse than PPO and IQL, and we changed the threshold-based agent to use threshold 0.7 based on re-tuning with the updated α.

disappeared (not shown in plots). Our observations support the hypothesis that with our current setup, sequential decision-making may not be helpful.

Next, we incorporated our theoretical takeaways from Section 5 and set $\alpha = 0.4$ to see if this provides better opportunity for RL. Results are shown in Figure 4 (right two plots). Indeed, our theoretical analysis transferred to PPO: we see that the PPO agent trained under $\gamma = 0.99$ performs better than $\gamma = 0$. Because this difference was not significant after 250K steps, we extended the learning to 1M steps and confirmed the returns differed significantly (not shown in plots): (3.14 ± 0.27) for $\gamma = 0.99$ vs. (2.65 ± 0.16) for $\gamma = 0$. However, like in Figure 3, PPO could not reach the average return of IQL and the threshold-based agent. We believe this gap in performance is due to the fact that the PPO policy does not receive suggestion probabilities from the LM as part of its input.

Our final computational experiment revolved around *multi-word suggestions*, drawing on the intuition that longer suggestions have more opportunity to improve text entry speed. We increased our candidate pool to $k = 5$ and allowed LM candidates to be 1 or 2 words. Since the joint likelihood of suggesting two words is always lower than the likelihood of suggesting just the first, we normalized the joint probability of 2-word suggestions, similar to Murray & Chiang (2018). See Appendix C for details. Unfortunately, our initial experiments indicated that allowing multi-word suggestions degraded the return for both oracle and threshold-based agents. Surprisingly, it also reduced the oracle agent's number of saved characters. The reason is that 2-word candidates can take up the slot of the correct 1-word candidate, preventing it from being seen by the agent. See Appendix D for details. Given these findings, we didn't train RL agents (which are GPU-intensive) in this setting.

6.3 User Study

In addition to our computational experiments, we ran a user study to understand how well our reward function aligns with real user behavior. Our goals were to 1) estimate real values for the reward function parameters, α and β, that capture the cognitive load of the user, and 2) explore the possibility of "accumulated fatigue" (Paas et al., 2003) for the users, meaning that as the number of surfaced suggestions increases, the user is less likely to accept suggestions. To address these goals, we ran a study where we asked $N = 9$ users to enter 42 sentences on a keyboard, both with and

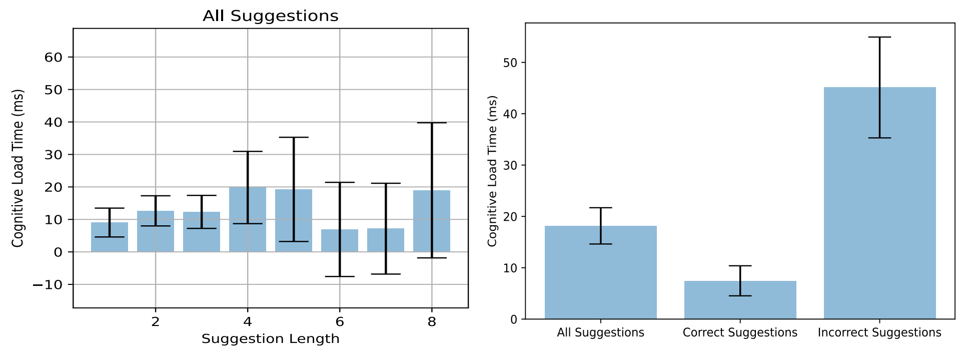

Figure 5: User study results (Section 6.3). Error bars depict 95% confidence interval. *Left:* Average cognitive load versus suggestion length. The cognitive load did not grow significantly with suggestion length. *Right:* Average cognitive load versus suggestion correctness. There is a significant difference between the user's cognitive load when considering correct versus incorrect suggestions.

without autocomplete suggestions. Figure 1 (right) shows an example of the study interface. All $N = 9$ users work with computers as a regular part of their profession.

We obtained 3968 instances where a user entered the same key given the same context (letters typed so far) both with and without autocomplete suggestions. To measure the cognitive load due to the suggestions, we looked at the time difference between how long it took the user to enter the next key in both cases. Figure 5 (left) shows this cognitive load measure as a function of suggestion length. Contrary to our initial estimate of $\alpha = 40/521$, the time users spend on suggestions did not grow with suggestion length, implying that α should be set to 0. The average cognitive load for users was 21.29 ± 3.61ms, which falls within the 20-30ms of a single saccade time for reading (Rayner & Clifton, 2009). This observation suggests that the user only has to saccade once, not twice, after reading a suggestion. This is likely because initially, their eyes are already focused where the suggestion appears, which can in return explain the widespread adoption of inline suggestions.

Furthermore, Figure 5 (right) shows that the cognitive load of looking at a suggestion is highly dependent on suggestion correctness. Specifically, users experienced cognitive loads of 9.18 ± 3.05ms and 50.49 ± 9.67ms for correct and incorrect suggestions respectively. This observation suggests that β is dependent on suggestion correctness, as opposed to our initial fixed estimate of $\beta = 60/521$.

Finally, we probed the presence of "accumulated fatigue", as defined earlier. See Figure 7 in Appendix E for results. The data did not show any evidence of declined acceptance rate based on the cumulative number of past suggestions, or the cumulative number of past incorrect suggestion.

Based on these findings, we re-ran our experiments on the Reddit Webis-TLDR-17 dataset with $\alpha = 0$, and $\beta = 10/521$ if the suggestion is correct and $50/521$ if it is wrong. With these changes, we found that the optimal threshold for the threshold-based agent was 0: always surface the top LM candidate, regardless of its probability. This is because by reducing α and β, we reduced the penalty for incorrect suggestions. This result implies that the sequential decision-making formulation does *not* improve text entry speed of an idealized user when α, β are chosen based on real user behavior. Yet in reality, users do not prefer always-on suggestions, meaning that there is something more than text entry speed that is important to the user (Quinn & Zhai, 2016).

7 Conclusion and Future Work

In this paper, we studied the problem of generating inline autocomplete suggestions via sequential decision-making. This formulation allowed us to apply RL methods to solve the problem, and define a reward function that captures user cognitive load through text entry speed. Our experimental findings suggest that sequential decision-making and RL do *not* improve text entry speed for an