Prompt Programming for Large Language Models via Mixed Initiative Interaction in a GUI

Tanya A. Morris
Computer Laboratory
University of Cambridge
tam71@cantab.ac.uk

Alan F. Blackwell
Computer Laboratory
University of Cambridge
afb21@cam.ac.uk

Abstract
Large Language Models (LLMs) now demonstrate many surprising capabilities that previously required special algorithms, for example interactive correction of syntax errors in structured text. However, the problem of how to systematically and reliably access these capabilities of LLMs has led to a new genre of "prompt programming" or "prompt engineering". This paper presents a design case study in which we apply OpenAI's Codex to an interface requiring syntax-constrained textual input, an email client. Via a mixed-initiative interface design, the system provides appropriate suggestions based on the output of the LLM. A user study was undertaken, finding that the incorporation of a LLM resulted in a decrease in perceived workload as well as a 62.5% reduction in errors. This work demonstrates how mixed-initiative interface design can better support attention investment in the use of LLMs, by delivering capabilities that might otherwise require prompt programming in a chat dialogue, but via a relatively conventional GUI.

1. Introduction
This paper describes a design case study exploring a new approach to the integration of predictive Large Language Models (LLMs) into functional user interfaces. At the time of writing, the release of the ChatGPT product, based on the GPT-n family of language models from OpenAI, has resulted in a step-change in popularity and public awareness of LLMs. In previous releases of the GPT-n series, and also in popular generative image models such as DALL-E and Stable Diffusion, substantial attention had been paid to the usability challenges of prompt engineering, or prompt programming. In ChatGPT and other dialog-based LLM deployments, the role of the generative prompt is combined with that of a chatbot dialog system. This obfuscates the distinction between user control of the generative output and Turing Test-style illusion of first person dialog to structure the interaction with the system.

Commentators on human-centric AI such as Ben Shneiderman (Shneiderman, 2020) have suggested that the first-person illusion is unhelpful or even unethical, and also that some people might find it easier to access LLM functionality if a constrained set of generative options were offered via a conventional user interface rather than a dialog prompt (Shneiderman, 2023). If the considerable power of LLMs was previously accessed via prompt *programming*, the substitution of more usable front ends for construction of those prompts can be considered as a new application of end-user programming. That is our motivation for presenting this work to the Psychology of Programming Interest Group.

Our case study is a classic domain for end-user programming research, in which non-programmers must construct and manipulate formally structured text syntax. Our illustrative example is the entry of syntactically-correct email addresses, when done by users who have impairments such that they may be unaware of the formal requirements of correct syntax, or be unable to generate valid examples due to cognitive, perceptual or motor impairments. We use the *Codex* variant of GPT-3 as the underlying language model. We apply principles of *mixed-initiative interaction*, an optimisation approach for design of intelligent user interfaces where AI agents collaborate with users to help them achieve their goals (Eric Horvitz, 1999).

2. Mixed-initiative interaction for user impairments
This application is designed for use by those who are facing cognitive, perceptual or motor impairments, including elderly users (Iancu & Iancu, 2017; Lindberg & De Troyer, 2021). (However, please note

that as an undergraduate student project, there is limited potential to undertake research with vulnerable populations - the evaluation study that is reported later was carried out with other students, as we will explain). For our intended population of users with impairments, in the case of entering email addresses, we address the following issues:

- *Difficulty remembering syntax* - suggestions to complete the email address will be offered to the user if they are paused, and corrections will be suggested if the syntactical rules of email addresses have not been followed.
- *Difficulty concentrating* - suggestions to change the input or move to a more supportive interface will appear if the user is paused for an extended period of time.
- *Difficulty using the keyboard* - suggestions or corrections will be offered if the user makes an error when inputting on the keyboard, and a user will be able to accept automated changes from the system.

Our system design applies key principles of mixed-initiative interaction (Eric Horvitz, 1999) as follows:

1. **Significant added value** where the user is unable to enter a correct email address.
2. **Consider uncertainty about the user's goal** by calculating the probability that the address needs to be corrected.
3. **Considering the status of a user's attention** by only interrupting the user if they are 'paused'.
4. **Inferring ideal action** by calculating the utility of each possible action.
5. **Employing dialog to resolve uncertainties** by offering a prompt with alternative actions.
6. **Allowing efficient direct invocation and termination:** If users want a higher or lower level of support, they can manually move to a more/less supportive interface via a task bar.
7. **Minimising costs of poor guesses about action and timing:** The dialog will timeout if the user does not respond. If the user continues typing, it will disappear.
8. **Providing mechanisms for efficient agent-user collaboration:** The user can accept an email address suggestion, and then edit it themselves.
9. **Employing socially appropriate behaviours for agent-user interaction:** The agent suggestions are polite and helpful.
10. **Maintaining working memory of recent transactions and continue to learn by observing:** The system continues to update the user model based on interaction with the system.

3. Implementation

Implementation is split into three main parts - the error model (including creation of an error model and appropriate Codex prompt), the user model (including the creation of the user model and its integration with the Codex prompt), and the GUI (including the creation of the graphical user interface and its integration with the user model). The back-end access to the Codex API was coded in Python and implemented as a Flask server. The front-end user model and GUI were coded in Javascript React.

3.1. Error Model

A dataset of pairs formatted as [erroneous email address, correct email address] was required to test various Codex prompts, and whether they could correct the erroneous email address. No existing dataset of this format existed, so we applied an error model to the Enron dataset[1] to create these pairs.

To form an appropriate error model, a corpus of spelling errors made by Wikipedia editors was used, which contains both the intended word (correct spelling) as well as the corresponding misspelt version[2]. The Levenshtein distance is used to calculate the 'difference' between strings, finding that the average edit distance of a misspelt word is 1.378 and the probability of an error at any given character is 0.158. The relative likelihood of that error being a substitution is 0.278, of a deletion is 0.428, and of an insertion is 0.294. Users are more likely to miss out a letter, as opposed to inserting an incorrect letter or substituting a letter for a different one.

A corpus of 150 email addresses was extracted from 'To:' fields in the Enron dataset, to which the error model was applied with probabilities of 0.01, 0.03 and 0.05, to explore the extent to which Codex can correct errors in email addresses.

3.2. User Model

The user model continually updates estimates of the attention of the user, estimating whether they are focused on their current task, and the skill level of the user, considering whether the difference between the text they have entered, and the most likely alternative suggested by Codex, implies they need more support. These values were used to form utility estimates that offered assistance when utility reached an appropriate threshold.

3.3. Codex Prompt

As is becoming increasingly well known, LLM behaviour in response to unconstrained text prompts can be unpredictable (Weidinger et al., 2022; Rae et al., 2022). Our approach, inspired by end-user programming, is to use a pre-tested structured prompt, with the input email address inserted into the prompt. This is inspired by Shneiderman's suggestion that a more structured approach can provide greater predictability for users (Shneiderman, 2023).

A variety of prompts to Codex were created, and evaluated using two Codex models: 'Cushman,' and 'Davinci' which is more capable but slower[3]. Other Codex parameters were `Temperature = 0`, to make the model more deterministic; `Stop = "\n"` to ensure that only one line of predicted text was returned; and `Best_of = true` to return only the most probable option.

OpenAI's guidance on prompt design (*Best practices for prompt engineering with OpenAI API | OpenAI Help Center*, n.d.) was followed, with all instructions at the beginning of the prompt. The 'context,' in this case the email address which requires correction, was separated from the instructions using triple quotation marks. Three prompts were tested as follows.

Prompt 1, no examples provided: `Correct the email address below. Incorrect company names or domain names should be corrected. Names which may have been spelt incorrectly should be corrected. Remove invalid symbols. The output should be an email address which is likely what the inputted email was intended to be. Email: """" + errorEmail + """" Correct Email:`

Prompt 2, the email address 'john.doe@hotmail.co.uk' is used to show the effect of individual errors (substitution, insertion, deletion) and how these should be corrected: `Correct the email address below. Incorrect company names or domain names should be corrected. Names which may have been spelt incorrectly should be corrected. Remove invalid symbols. The output should`

[1]https://www.cs.cmu.edu/ enron/
[2]https://www.dcs.bbk.ac.uk/ roger/wikipedia.dat
[3]https://platform.openai.com

```
be an email address which is likely what the inputted email was
intended to be.  Email:  johbn.doe@hotmail.co.uk Correct Email:
john.doe@hotmail.co.uk Email:  johb.doe@hotmail.co.uk Correct Email:
john.doe@hotmail.co.uk Email:  jhn.doe@hotmail.co.uk Correct Email:
john.doe@hotmail.co.uk Email:  """" + errorEmail + """" Correct
Email:
```

Prompt 3, a variety of email addresses were used, each showing how to correct multiple errors in one email address:

```
Correct the email address below.
Incorrect company names or domain names should be corrected.  Names
which may have been spelt incorrectly should be corrected.  Remove
invalid symbols.  The output should be an email address which
is likely what the inputted email was intended to be.  Email:
jpohn.doe@hotail.co.uk Correct Email:  john.doe@hotmail.co.uk Email:
alicesmith10o00@gmai1.com Correct Email:  alicesmith1000@gmail.com
Email:  bjohnsonicloud,com Correct Email:  bjohnson@icloud.com
Email:  """" + errorEmail + """" Correct Email:
```

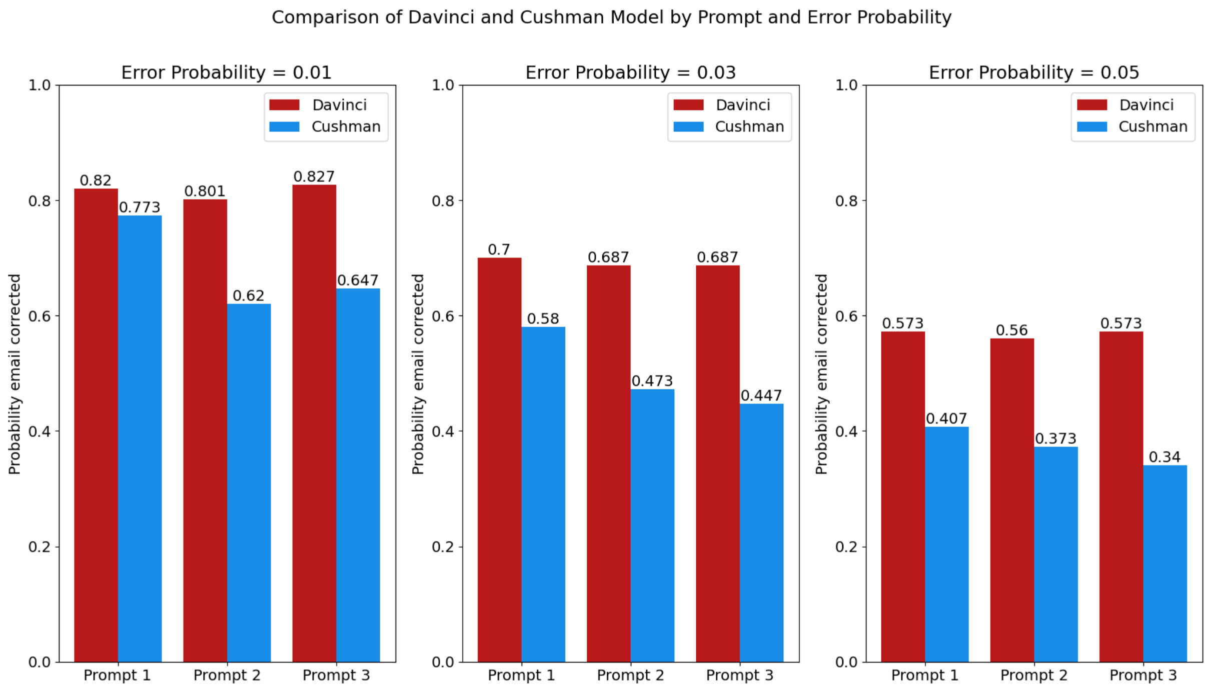


Figure 1 – Probability of an email address being successfully corrected by Codex, for each error probability, model, and prompt.

The results of the prompt evaluation are displayed in Figure 1. The Davinci model behaved similarly for each prompt, whereas the Cushman model performed better using Prompt 1. Davinci outperformed the Cushman model, with on average 17% more email addresses successfully corrected. Since the prompts had similar performance for the Davinci model, Prompt 1 was chosen because it requires fewer tokens to be sent.

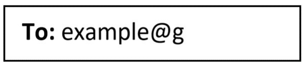
The Codex rate limit had to be considered when calling the API. Various techniques were employed to decrease the load on the API, and to avoid the user waiting while the system appears idle. On 23rd March, OpenAI discontinued support for Codex, including both the Davinci and Cushman models. Fortunately, we had already created a system for user testing which could use the edit distance to find the most similar cached response.

3.4. Interface overview

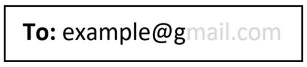
Four alternative input methods were implemented, offering increasing levels of support based on the Codex suggestions:

1. No support - a plain text input box.
2. Autofill - suggestions to complete the email address appear in the box, in a light grey shade.
3. Dropdown - suggestions to complete or correct the email address appear as a dropdown menu.
4. AutoSuggest - suggestions to correct the email appear above the text, with boxes suggesting insertion, deletion or substitution.

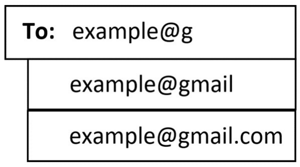
Mockups were created to understand how each of these options would behave when an email address is incomplete (Figure 2) compared to when an email address is erroneous (Figure 3). Note that the AutoSuggest option behaves in the same way as Autofill when there are no immediate errors to correct.



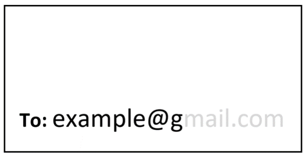
(a) No Support



(b) Autofill

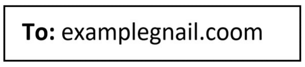


(c) Dropdown

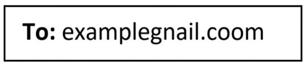


(d) AutoSuggest

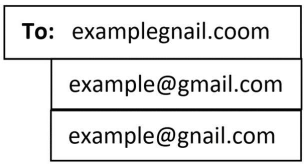
Figure 2 – How each text input option should behave when an email address is not completed.



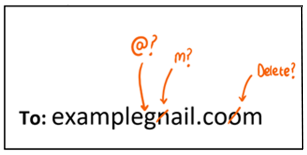
(a) No Support



(b) Autofill



(c) Dropdown



(d) AutoSuggest

Figure 3 – How each text input option should behave when an email address is erroneous.

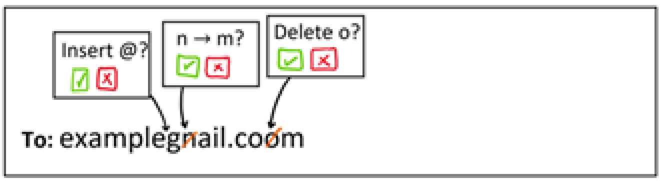


Figure 4 – Mockup including the scribble over errors.

Material UI 'MUI Core' React components were used to create the GUI, offering an open source library[4] to implement interactive elements of the project. The 'react-autocomplete-hint' component was employed to create the ghost text box[5].

[4]https://mui.com/pricing/
[5]https://www.npmjs.com/package/react-autocomplete-hint

3.5. AutoSuggest Interface

Typically, spelling errors are represented with a red line beneath the word. This does not tell the user where the error is or how to correct it, which is important to support an improved understanding of the syntax. Hence the most supportive interface option will point out specific errors (substitution, deletion, insertion), with the user able to accept or reject the system's suggestions.

An initial mockup showed a scribble over the text to be removed or replaced (Figure 4), but hiding the underlying text would make the text challenging to read, particularly for the target user who may already have reduced visual acuity. Instead, we reverted to the typical approach of a line underneath the text.

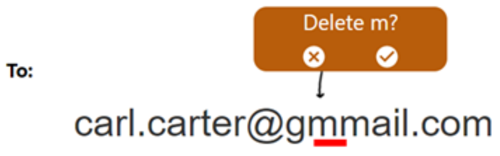
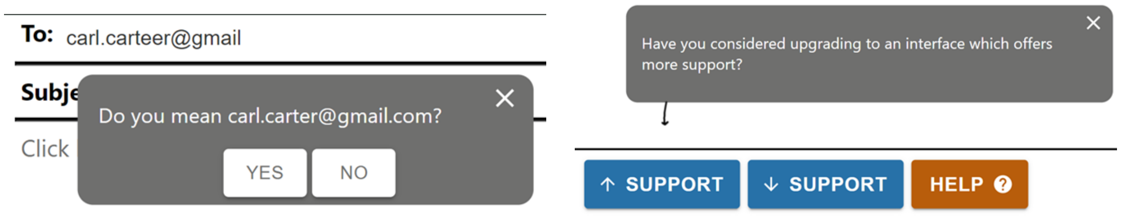


Figure 5 – Example of a suggestion being highlighted.

To support user control, it is important to respect when they reject suggestions by clicking the close button (Wen & Imamizu, 2022). A list of 'rejected' suggestions was formed to ensure the same suggestion was not made again.

Each page has a 'send' button, used in testing to submit the email and clear the input box. There are also 'increase support' and 'decrease support' buttons on each page, which enable a user to move between different interfaces. If the system deems a user needs help, but has no suggestions to change their current input (shown in Figure 6a), they may be prompted to move to a more supportive interface (shown in Figure 6b). The pages are summarised in Figure 7.



Do you mean carl.carter@gmail.com?

Have you considered upgrading to an interface which offers more support?

(a) A pop-up prompting a user to consider changing their input. *(b) A pop-up prompting a user to consider moving to a more supportive interface.*

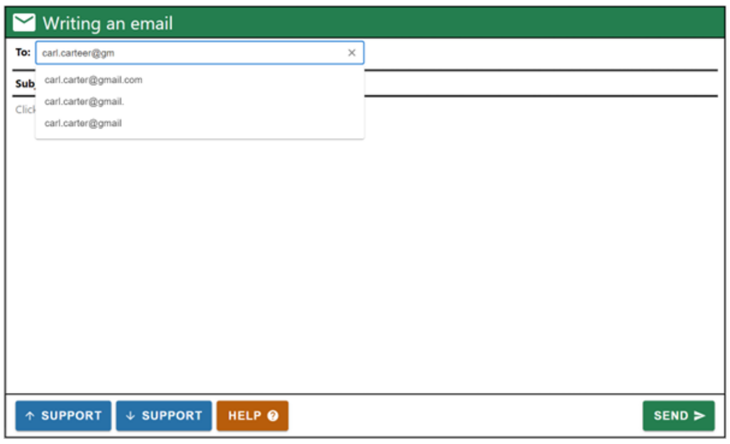
Figure 6 – Two types of pop-ups used in the system.

The previous example showed an email address with a suspected error, but each interface behaves differently if the email address is correct but incomplete (Figure 8).

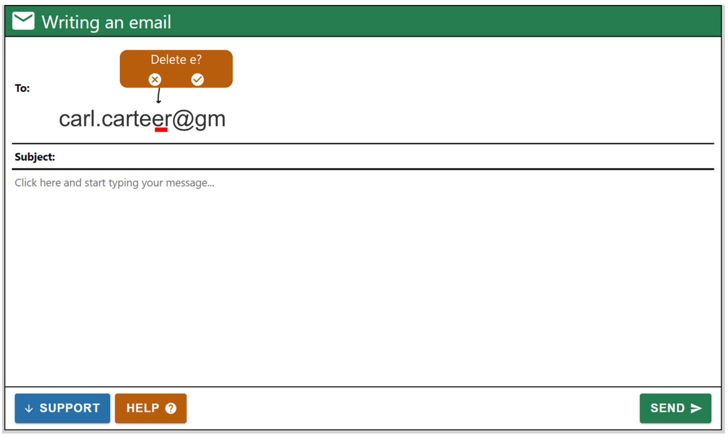
4. User Study

Because this student project did not offer scope to work with the target special population of older users with significant impairments, actual participants were Cambridge undergraduates, with errors forced into their typing stream to simulate the experience of the target users. A typical error rate for typing is 1.17%, with a standard deviation of 1.43% (Dhakal, Feit, Kristensson, & Oulasvirta, 2018). To approximate the skill level of someone who struggles to use a keyboard, we forced errors at 2 s.d. from the mean 4%.

A trade-off between the expected time required to complete the study and the volume of data expected to be retrieved was considered. While asking a user to input a higher number of email addresses into the system would provide more data to analyse, it would also increase the time required by the user, decreasing the likelihood of a participant agreeing to the study. 15 email addresses were randomly selected from the Enron dataset, with the study having an expected completion time of 15 minutes per

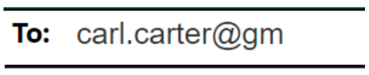


(a) Dropdown input box, where suggestions to correct and complete the email address appear in the dropdown menu, with the most specific option at the top.

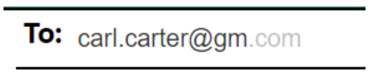


(b) AutoSuggest input box, where errors are pointed out individually.

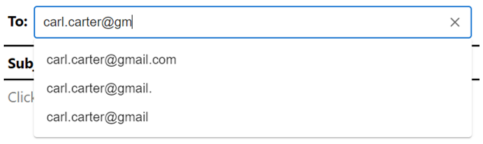
Figure 7 – Each page of the interface.



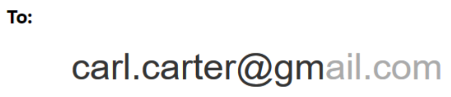
(a) The interface with no help.



(b) The Autofill interface displays 'ghost text' of Codex's prediction for the rest of the email.



(c) The Dropdown interface displays possible completions.



(d) The AutoSuggest interface also displays 'ghost text'.

Figure 8 – Each text input option interacting with a partially complete email address.

user. Having email addresses ending only with ‘enron.com’ could introduce bias and is not representative of the intended use, so each email address was given a 50% chance of being altered to take a different domain name (from the top domain names in the UK[6]).

The study has 5 sections:

1. Control: Navigation disabled, use only the interface with no help (including no help pop-ups).
2. Navigation disabled, use only the interface with autofill (with help pop-ups).
3. Navigation disabled, use only the interface with dropdown (with help pop-ups).
4. Navigation disabled, use only the interface with the auto-suggestions (with help pop-ups).
5. Intended use of system: Navigation enabled, maneuver between interface options if desired (with help pop-ups).

Users were warned that errors would be forced into their typing stream, but were asked to treat these in the same way as if they had made the error naturally.

The order of sections 1-4 was randomised for each participant, but section 5 was always last. The order of the email addresses was also randomised. In each section, users were asked to input 3 email addresses, which were shown in large font on a piece of paper. Once the user was ready, the email address was removed from sight, they typed it from memory, then clicked submit. Following each section, users completed the NASA TLX questionnaire (Hart, 1986).

At the end of the study, participants were asked to provide pairwise comparisons of the TLX factors, to be used in constructing a weighted ranking. The were then asked open questions about the support offered by the system.

A pilot study found that lack of errors led the system to believe the user did not need help. To increase the likelihood of participants needing help, we increased the likelihood of forced errors to 6%, and encouraged users to defer error correction until they had stopped typing.

14 participants were recruited, who were all Cambridge undergraduates from a variety of subjects. The study was approved by the Ethics Committee of the Cambridge Computer Lab. It took each participant approximately 17 minutes to complete.

5. Results

Written responses to the open questions were grouped, with the main themes summarised in Table 1. A majority of participants agreed that the system was easy to use, and that they would use a similar system again (Figure 9). Crucially, 71% of participants believed that the suggestions from the LLM added value to the interface.

[6]https://email-verify.my-addr.com/list-of-most-popular-email-domains.php

Sentiment	Theme	Times Mentioned	
Positive	Dropdown was best	4	14
	Helpful	7	
	Easy to use	3	
Negative	Would like a shortcut to select dropdown/buttons	3	11
	Help appeared too late	2	
	Hindrance or too busy	5	
	Suggestions inaccurate	1	

Table 1 – Common themes in response to ‘opinion of the interface’.

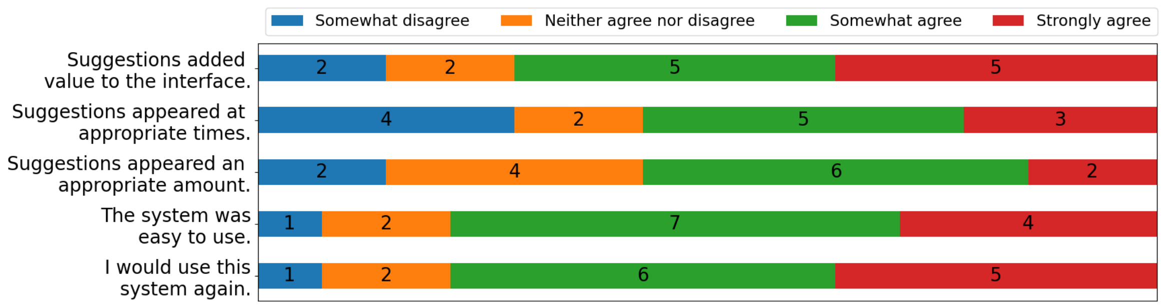
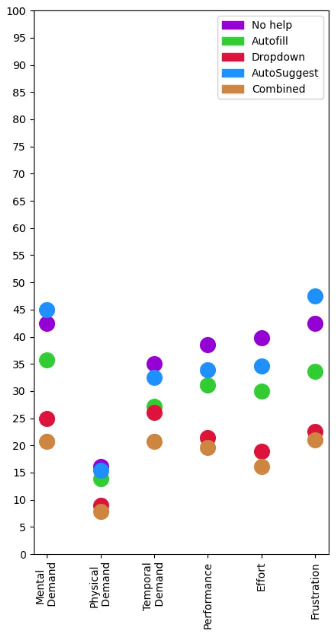


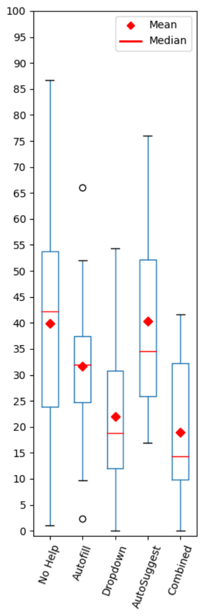
Figure 9 – Summary of final evaluation.

The NASA TLX responses for each section of the study, as in Figure 10a, shows that the novel Auto-Suggest interface required the most mental demand and caused the most frustration. As expected, the interface which offered no help provided the worst performance rating yet required the highest effort. The overall system, where users could navigate freely, had the lowest demand in all categories, coupled with the best performance.

Users were asked to rank which factors have more impact on workload, and these ranks were used to create a weighted overall score (Figure 10b). The interface with no help has the greatest variability, suggesting some users appreciate extra support while others prefer to correct manually. The interface with autofill, which is most similar to existing products, had the smallest range.



(a) Comparisons of average perceived demand for each interface, including ‘combined’ (whole system).



(b) The mean, median and outliers for overall TLX of each interface option.

Figure 10 – Analysis of raw (a) and weighted (b) TLX scores for each interface.

To evaluate task performance, the average edit distance of entered email addresses from the correct address is displayed in Figure 11. The Autofill and Dropdown interfaces result in improved error rate. The AutoSuggest interface had more uncorrected errors than the interface with no help at all. It was unfamilar to users, with feedback suggesting that there were too many pop-ups to focus. The combined system, where users were free to manoeuvre between interfaces and customise their level of support, provided the best results.

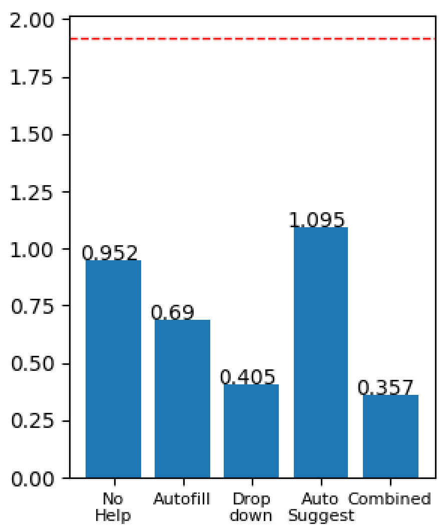


Figure 11 – The average edit distance of inputted emails for each interface, against the edit distance if no errors were corrected.

6. Conclusions

This project points to the positive impact that incorporating LLMs into 'everyday' interfaces can have. The adaptive combination of these supportive interfaces, allowing user choice and guided by a mixed-initiative utility model, showed not only an improvement in performance, but a decrease in perceived workload.

Despite the potential benefit of LLMs in a supportive interface, further issues must be considered, including risk of data leaks due to the training data of the LLM and risk of misinformation where the LLM may not be correct (Weidinger et al., 2022, 2021). These risks can be mitigated via careful prompt engineering. There is a further risk of intrinsic bias in the models (Bommasani et al., 2022; Weidinger et al., 2021), and it would be important to train a model with diverse email addresses, including names from a wide variety of cultures.

If LLMs are to be incorporated into supportive systems designed for people with impairments or elderly users, members of those populations should be included in discourse surrounding AI (Stypinska, 2022). Older people often self-stereotype, with lower self-efficacy leading them to doubt their ability to benefit from technology (Stypinska, 2022; Kottl, Gallistl, Rohner, & Ayalon, 2021). If AI is used to provide support in a mixed-initiative setting, such users should be able to make informed decisions about whether this support could benefit them (Stypinska, 2022). Instead of viewing AI as a tool which older users do not want to interact with, future research could turn to designing AI interfaces which are accessible for all.

7. Future Work

Further exploration, where LLMs are incorporated into other types of interfaces, could show whether these results are replicated in other contexts. In particular, the defined syntax required to type URLs into a browser would be applicable to a similar project. Other examples of defined syntax which could be explored include academic citations, phone numbers or credit card numbers.

Some participants in our limited user study did not want to interact with the help offered by the system, reporting that they do not in general enjoy 'helper' systems, because they find it quicker to correct errors manually themselves. Future work could explore whether these attitudes that we observed among students at the University of Cambridge are also reported by some members of the target design population of older and impaired users.

8. References

Best practices for prompt engineering with OpenAI API | OpenAI Help Center. (n.d.). Retrieved from https://help.openai.com/en/articles/6654000-best-practices-for-prompt-engineering-with-openai-api

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., ... Liang, P. (2022, July). *On the Opportunities and Risks of Foundation Models.* arXiv. Retrieved 2023-04-04, from http://arxiv.org/abs/2108.07258 (arXiv:2108.07258 [cs])

Dhakal, V., Feit, A. M., Kristensson, P. O., & Oulasvirta, A. (2018). *Observations on Typing from 136 Million Keystrokes.* Retrieved 2023-01-20, from https://dl.acm.org/doi/epdf/10.1145/3173574.3174220 doi: 10.1145/3173574.3174220

Eric Horvitz. (1999). *Principles of mixed-initiative user interfaces.* Retrieved 2022-09-03, from https://dl.acm.org/doi/epdf/10.1145/302979.303030 doi: 10.1145/302979.303030

Hart, S. G. (1986, January). *NASA Task Load Index (TLX).* Retrieved 2023-03-03, from https://ntrs.nasa.gov/citations/20000021487 (NTRS Author Affiliations: NASA Ames Research Center NTRS Document ID: 20000021487 NTRS Research Center: Ames Research Center (ARC))

Iancu, I., & Iancu, B. (2017, September). Elderly in the Digital Era. Theoretical Perspectives on Assistive Technologies. *Technologies*, *5*(3), 60. Retrieved 2023-04-13, from https://www.mdpi.com/2227-7080/5/3/60 doi: 10.3390/technologies5030060

Kottl, H., Gallistl, V., Rohner, R., & Ayalon, L. (2021, September). *â[U+0080][U+009C]But at the age of 85? Forget it!â[U+0080][U+009D]: Internalized ageism, a barrier to technology use | Elsevier Enhanced Reader.*

Lindberg, R. S. N., & De Troyer, O. (2021, November). Towards an Up to Date list of Design Guidelines for Elderly Users. In *CHI Greece 2021: 1st International Conference of the ACM Greek SIGCHI Chapter* (pp. 1–7). Online (Athens, Greece) Greece: ACM. Retrieved 2023-02-01, from https://dl.acm.org/doi/10.1145/3489410.3489418 doi: 10.1145/3489410.3489418

Rae, J. W., Borgeaud, S., Cai, T., Millican, K., Hoffmann, J., Song, F., ... Irving, G. (2022, January). *Scaling Language Models: Methods, Analysis & Insights from Training Gopher.* arXiv. Retrieved 2023-04-17, from http://arxiv.org/abs/2112.11446 (arXiv:2112.11446 [cs])

Shneiderman, B. (2020, April). Human-Centered Artificial Intelligence: Reliable, Safe & Trustworthy. *International Journal of Humanâ[U+0080][U+0093]Computer Interaction*, *36*(6), 495–504. Retrieved 2023-04-23, from https://www.tandfonline.com/doi/full/10.1080/10447318.2020.1741118 doi: 10.1080/10447318.2020.1741118

Shneiderman, B. (2023, June). *97th NOTE on Human-Centered AI.*

Stypinska, J. (2022, October). AI ageism: a critical roadmap for studying age discrimination and exclusion in digitalized societies. *AI & SOCIETY*. Retrieved 2023-04-28, from https://doi.org/10.1007/s00146-022-01553-5 doi: 10.1007/s00146-022-01553-5

Weidinger, L., Mellor, J., Rauh, M., Griffin, C., Uesato, J., Huang, P.-S., ... Gabriel, I. (2021, December). *Ethical and social risks of harm from Language Models.* arXiv. Retrieved 2023-04-07, from http://arxiv.org/abs/2112.04359

Weidinger, L., Uesato, J., Rauh, M., Griffin, C., Huang, P.-S., Mellor, J., ... Gabriel, I. (2022, June). Taxonomy of Risks posed by Language Models. In *2022 ACM Conference on Fairness, Accountability, and Transparency* (pp. 214–229). New York, NY, USA: Association for Computing Machinery. Retrieved 2023-04-14, from https://dl.acm.org/doi/10.1145/3531146.3533088 doi: 10.1145/3531146.3533088

Wen, W., & Imamizu, H. (2022, April). The sense of agency in perception, behaviour and humanâ[U+0080][U+0093]machine interactions. *Nature Reviews Psychology*, *1*(4), 211–222. Retrieved 2023-04-25, from https://www.nature.com/articles/s44159-022-00030-6 doi: 10.1038/s44159-022-00030-6