Speakerly™: A Voice-based Writing Assistant for Text Composition

Dhruv Kumar[*], **Vipul Raheja**[*], **Alice Kaiser-Schatzlein, Robyn Perry,**
Apurva Joshi, Justin Hugues-Nuger, Samuel Lou, Navid Chowdhury
Grammarly
firstname.lastname@grammarly.com

Abstract

We present Speakerly™, a new real-time voice-based writing assistance system that helps users with text composition across various use cases such as emails, instant messages, and notes. The user can interact with the system through instructions or dictation, and the system generates a well-formatted and coherent document. We describe the system architecture and detail how we address the various challenges while building and deploying such a system at scale. More specifically, our system uses a combination of small, task-specific models as well as pre-trained language models for fast and effective text composition while supporting a variety of input modes for better usability.

1 Introduction

Writing is a multi-step process that involves planning (ideation), translation (composition), and reviewing (revision) (Flower and Hayes, 1981). In the ideation phase, the writer gathers information and organizes their thoughts. The composition step involves articulating the ideas effectively through the use of the right words and arranging them cogently in a draft. During revision, the focus is on grammatical correctness, logical flow of ideas, coherent document structure, and style.

Most current writing assistants have been limited in their ability to provide seamless writing assistance across all the stages, take into account the user context, and be robust to work on diverse real-world use cases at scale (Gero et al., 2022a).

In this work, we introduce Speakerly™, a voice-based end-to-end writing assistance system that works across the different stages of writing, helping users become more efficient with their communication. The user uses the voice interface to articulate their thoughts in natural speech. Our system then creates a polished and ready-to-send first draft while addressing all the intermediate issues,
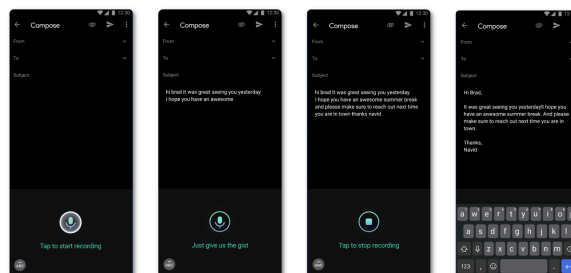

Figure 1: An illustrative example of Speakerly™ for email composition on mobile. A user presses the microphone button at the bottom of their email application and starts speaking naturally (no templatization or structural tailoring of the speech input is needed). Once they stop speaking, Speakerly™ converts the speech into structured, well-formatted, and polished compositions.

such as structure, formatting, appropriate word usage, and document coherence.

We use voice as it is a natural and efficient input modality, allowing users to compose their thoughts quickly and even use the system in eyes-free scenarios while performing tasks such as walking and driving (Kamm, 1995; Cohen and Oviatt, 1995; Ruan et al., 2018). Moreover, with the increased ubiquity of voice-based assistants, such as Alexa and Siri, voice-based interactions have become more common and intuitive for users (Porcheron et al., 2018).

However, using voice has some challenges. First, during the ideation stage, the user typically only has a rough idea of what they want to write. Thus, if the system is unable to handle a lack of structure and slight incoherence in the input, users will end up spending a significant amount of time on fixing the output. Second, different writers can have varied needs requiring the system to handle the demands and constraints of different use cases. For example, short vs. long inputs, instructional vs. dictation inputs, open vs closed-ended inputs, and specific structures and formatting for emails, instant messages, and notes (Table 1). Finally, the

---

[*]Equal contribution by both authors.

system should be reasonably fast so that it can provide a delightful user experience.

Speakerly™ is composed of multiple stages (Section 3) that progressively refine the relatively noisy and unstructured speech from the user and address the aforementioned challenges and requirements. In the remainder of this paper, we describe the technical system architecture and our approaches to address challenges related to modeling, evaluation, inference, and sensitivity.

2 Related Work

Most research in the past has been limited to either a single use case for composition or one particular stage of the writing process. For example, previous works have focused on email writing (Hui et al., 2018), science writing (Gero et al., 2022b), story writing (Clark et al., 2018; Coenen et al., 2021), slogan and metaphor writing (Gero and Chilton, 2019), poetry writing (Chakrabarty et al., 2022), and support comments (Peng et al., 2020), to name a few. Our system, in contrast, can handle various use cases ranging from short instant messages to long notes to open-ended instructions to closed-ended and information-dense dictations.

On the other hand, some writing assistance-focused works disproportionately emphasize specific stages of writing, such as editing and revision (Mallinson et al., 2022; Du et al., 2022; Kim et al., 2022; Schick et al., 2023; Raheja et al., 2023) rather than end-to-end writing assistance. Again, in contrast, our system is much more extensive as it takes in noisy and unstructured speech input and iteratively refines it to produce a final well-formatted output rather than focusing on a single-shot, structured text-to-text transformation.

Voice-based input has been known to optimize people's interaction and has been studied in the past (Williams, 1998) and is well-integrated in virtual assistants (such as Siri and Alexa). It has been used for various tasks such as voice notes (Stifelman et al., 1993), data capturing (Luo et al., 2021), information querying (Schalkwyk et al., 2010) and data exploration (Srinivasan et al., 2020). Such systems can have speech recognition errors that are difficult to recover from and restrict the user's natural speaking behavior (Luo et al., 2020). To tackle these problems, recent works have looked at voice-based text editing (Ghosh, 2020; Fan et al., 2021).

3 System Description

Our system takes natural speech from the user as input and generates a coherent and well-formatted text output. As shown in Fig. 2, the input progressively gets refined and enhanced as it traverses the pipeline, consisting of multiple task-specific models. Each stage can have its own errors. Hence, models across the pipeline are designed with complementary, sometimes overlapping capabilities, which allows them to recover from errors collectively and improve robustness to variation and noise in the input.

The pipeline has three main components: Automatic Speech Recognition (ASR), Normalization, and Comprehension. The ASR module takes raw speech and converts it to text. Then, the normalization module cleans up speech disfluencies, adds punctuation, and applies grammatical error corrections (GEC). Finally, the comprehension module cleans the text of remaining issues, such as incoherent document structure, word choice, formatting, formality, and style, and composes the final output text, handling instruction or dictation, or any other mode of input for a variety of use cases. We now explain these three components in more detail:

3.1 Automatic Speech Recognition (ASR)

The entry point to the system is an ASR component. This stage is responsible for the transcription of the user's spoken input and also handles basic speech recognition errors, such as filler words and background noise [1]. We leverage out-of-the-box ASR solutions and experiment with Speech-to-Text services from Microsoft Azure, Google Cloud, and OpenAI Whisper. In general, Google and Microsoft Azure were at par in terms of supported features, such as support for streaming (real-time recognition), recognition of different dialects, spoken punctuation recognition, vocabulary customization, and price. We also considered OpenAI Whisper since it is open-source and about 70% cheaper. We eventually chose to use the Microsoft Azure Speech-to-Text due to quality considerations (Section 4.1).

---

[1]We tested the system by using muffled voice and playing street sounds in the background, while speaking to the system. We found that while ASR can deal with most ambient and low noises, any loud sound, can prevent it from picking up some words or inserting incorrect words. We deal with such cases in the Normalization step (Section 3.2).
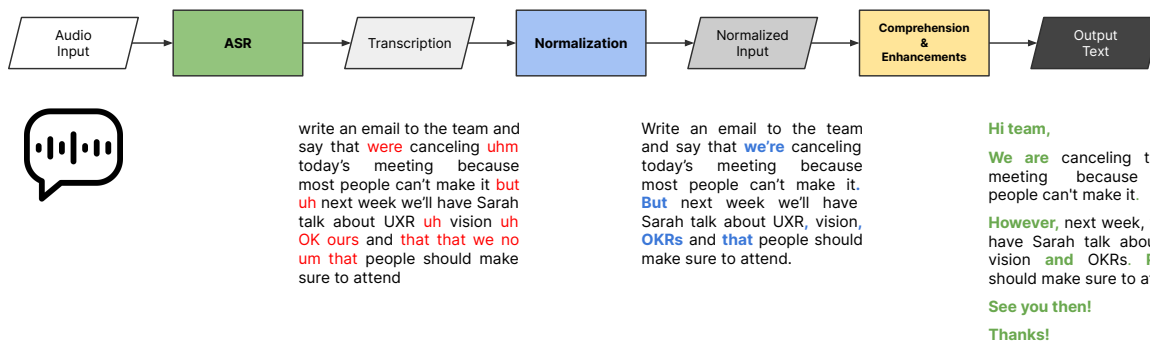
Figure 2: Overview of the system architecture. The ASR system first transcribes the input. Then, the Normalization stage fixes the issues in the transcribed input (shown in red and blue). Finally, the comprehension stage generates a well-formatted and coherent output text with further enhancements.

3.2 Normalization

The transcribed audio input may still contain noise stemming from ASR errors, speech disfluencies, uniqueness of individual elocution, ambiguous word boundaries, background noise, and lack of context, among others. Therefore, we introduce another stage in the pipeline to enrich the speech transcription further and get a cleaner input for the downstream comprehension model(s)[2]. This stage comprises three sub-stages dedicated to addressing specific issues in the transcription: Speech Disfluency Filtering, Punctuation Restoration, and Grammatical Error Correction. We now describe these in more detail.

3.2.1 Speech Disfluency Filtering

One of the numerous issues encountered in speech-based systems pertains to the inherent fluidity of spoken language, characterized by the occurrence of errors and spontaneous self-correction. Speakers, upon recognizing their speech errors, instinctively engage in the process of rectification by means of editing, reformulating, or starting afresh. This instinctual and subconscious phenomenon is a common and integral part of spontaneous human utterance, referred to as disfluencies (Shriberg, 1994), and poses significant challenges to the real-world deployment of speech-based systems.

Specifically, this part of the system focuses on detecting and removing disfluent tokens in the transcribed text and not replacing them with correct hypotheses. We formulate this as a token-level sequence tagging problem and experiment with three models. To categorize the disfluencies,

we use the framework defined in Shriberg (1994), which has three categories: *repetitions* (one or many words are repeated), *replacements* (a disfluent word or phrase is replaced with a fluent one), and *restarts* (initial utterance is completely abandoned and restarted).

Following are the details of the Disfluency Filtering models:

1. **Baseline**: An off-the-shelf model for joint disfluency detection and constituency parsing (Jamshid Lou and Johnson, 2020).

2. **DISF-SB-QA**: RoBERTa (Liu et al., 2019) model, fine-tuned on two publicly available datasets: The Switchboard Corpus (Godfrey et al., 1992) and Disfl-QA (Gupta et al., 2021).

3. **DISF-SB-QA-LD**: DISF-SB-QA model further fine-tuned on an augmented dataset of artificial disfluencies and task-specific data using LARD (Passali et al., 2022).

3.2.2 Punctuation Restoration

Once the disfluencies are removed, the input is still a stream of text without any punctuation or sentence segmentation. Therefore, the next step in the system is to restore punctuation (including capitalization). We experiment with three models that are trained to perform multi-class token classification. Specifically, there are five categories describing the respective token-level edit actions they apply:

- COMMA: Append one of [, ; : -]

- PERIOD: Append .

- QUESTIONMARK: Append one of [? !]

- CAPITALIZATION: Capitalize the word

---

[2]We experimented with handling these issues with the fine-tuned comprehension model (Section 3.3.1) but found that it could not reliably fix all of the issues, further resulting in deterioration in the quality of the generated text.

- NONE: No change

Following are the details of the Punctuation restoration models:

1. **rpunct**[3] is an open-source Python package for punctuation restoration, which uses a BERT-base model trained on Yelp reviews dataset[4]. We use this as our baseline.

2. **PUNCT-COMP**: Fine-tuned DistilBERT (Sanh et al., 2019) model on the same dataset as COMP-FT (described in Section 3.3.1).

3. **PUNCT-COMP-GEC**: Retrained version of PUNCT-COMP model after applying grammatical corrections (Section 3.2.3) to the dataset.

3.2.3 Grammatical Error Correction (GEC)

We use the GECToR system (Omelianchuk et al., 2020) for grammatical error correction. Similar to our models for Disfluency filtering and Punctuation restoration, it is a sequence tagging model using a Transformer-based encoder.

3.3 Comprehension

The output from the normalization step is then fed into the comprehension stage, which transforms the normalized input into a well-structured and coherent output, handling a wide variety of inputs. Table 1 shows the different types of inputs that the comprehension stage can handle. For example, the input can be an instruction or a dictation; an email, an instant message, or a note; open-ended or closed-ended[5]. Moreover, the spoken text can often be incomplete and noisy. Thus, the comprehension model enhances the quality of such text while minimizing meaning change and hallucination.

We experiment with two approaches for the comprehension stage. The first is fine-tuning a lightweight pre-trained model (called COMP-FT), and the second is using a pre-trained LLM out-of-the-box (called COMP-LLM).

3.3.1 COMP-FT

We use Pegasus (Zhang et al., 2020) (770M parameters), a transformer-based encoder-decoder. We limit ourselves to a small model since larger models have a higher latency, and we find that a model of this size can handle a significant portion of inputs. Since smaller models do not work well on open-ended generation, we limit it to closed-ended inputs. Model training details are present in Appendix A.

To fine-tune COMP-FT, we create a dataset containing 28k/1k/1k input-output pairs for training/validation/test sets, respectively. First, we ask human annotators to create 10k instruction-output pairs covering the various instruction-based use cases described earlier. Then, we create dictation-based data by removing the formatting and paraphrasing[6] the outputs from this dataset, and use the resulting text as inputs instead.

Finally, we augment the dataset by applying 25 different augmentations to deal with the issues that were either not handled or were introduced by the earlier stages of the pipeline. We build upon NL-Augmenter (Dhole et al., 2021), an open-source library that contains 117 transformations and 23 filters for a variety of natural language tasks. A selection of the augmentations can be found in Appendix C.

3.3.2 COMP-LLM

We use the *gpt-3.5-turbo* model from the Azure OpenAI platform. Since this model is a chat-based model, the main challenge is to find the right prompt for all our use cases. Further, the text generated by it is prone to verbosity and often contains hallucinations leading to meaning change. The benefit, however, is its ability to handle open-ended inputs such as *"Write a list of items to bring camping"*. Finally, it has higher latency and is more expensive to deploy.

3.3.3 Hybrid Approach

Since both COMP-FT and COMP-LLM are effective at different use cases, we combine both models into a hybrid approach. Outputs requiring more open-ended generation and having low scope for sensitivity issues are passed to COMP-LLM, whereas shorter inputs and those which require more factual consistency are processed by COMP-FT. The last column in Table 1 shows which model processes the different inputs. We train a binary classifier, a fine-tuned DistilBERT (Sanh et al., 2019) model, to decide whether the system should use COMP-FT or COMP-LLM. This model

---

Example	Input Type	Content Type	Intent Type	Model
Hey John, are you coming to the meeting later today?	Dictation	Closed-ended	Messaging	COMP-FT
Email Sam, we met with Joe today, meeting went well, follow-up with him next week.	Dictation	Open-ended	Email	COMP-FT
After more than 50 years The Eagles are heading on the road for what they say will be their "final" tour. On Thursday the legendary band announced "The Long Goodbye" tour that is set to kick off September 7 in New York.	Dictation	Closed-ended	Notes	COMP-FT
Send an email to Joe. Let him know that fundraiser is a go, and it will be happening next Wednesday at 8:00. PM.	Instruction	Closed-ended	Email	COMP-FT
Pick up groceries at 5 pm tomorrow.	Instruction	Closed-ended	Notes	COMP-FT
Write a thoughtful birthday wish for Jim. He is one of my oldest friends. He is turning 31. Make the message witty.	Instruction	Open-ended	Messaging	COMP-LLM
Write a blog post on AI from the perspective of a 30-year-old adult.	Instruction	Open-ended	Notes	COMP-LLM

Table 1: Different types of inputs (i.e. normalization outputs) handled by our system, along with their characteristics.

System	WER (%)↓	WRR (%)↑
Microsoft Azure[1]	**3.37**	**97.13**
Google Speech-to-Text[2]	4.55	96.79
OpenAI Whisper[3]	4.43	96.83

Table 2: Performance comparison of different ASR solutions. WER indicates Word Error Rate, and WRR indicates Word Recognition Rate.

System	CCPE-M	Meetings
Baseline	59.2 / **75.3** / 66.3	76.5 / 51.2 / 61.3
Disf-SB-QA	**83.5** / 55.0 / 66.3	87.4 / 82.2 / 84.7
Disf-SB-QA-LD	78.7 / 68.4 / **73.2**	**97.3** / **89.5** / **93.2**

Table 3: Performance comparison of different Disfluency Filtering models (Precision / Recall / F1).

was trained using a manually created dataset containing 1000 examples. The classifier is applied to the output text of the normalization stage.

4 Evaluation

4.1 ASR

In order to evaluate the quality of the various ASR systems, we collected a dataset of 1000 voice inputs by releasing the system to a small set of internal users, who were asked to use the system for their composition needs. Expert annotators then transcribed these voice recordings, and the ASR systems were evaluated using the standard ASR metrics of Word Error Rate (WER) and Word Recognition Rate (WRR). Table 2 shows the performance comparison of these different ASR systems on this set. We found that Microsoft Azure Speech-to-Text achieved the best performance, which determined our choice of ASR system for Speakerly™.

4.2 Speech Disfluency Filtering

Since the Disfluency Filtering models are sequence tagging models, we use Precision/Recall/F1 as the evaluation metrics on two evaluation datasets. First is the CCPE-M dataset (Radlinski et al., 2019), a corpus consisting of dialogues between two paid crowd-workers using a Wizard-of-Oz based, Coached Conversational Preference Elicitation (CCPE) methodology. We also collect and annotate (via crowdsourcing) an internal dataset sourced from the transcripts of company-wide, internal Zoom meetings, which were then annotated for the disfluency filtering task by expert annotators. Table 3 summarizes the results of the three models on the two evaluation sets. We observed that DISF-SB-QA-LD was the best-performing model, owing largely to the task-specific data augmentation.

4.3 Punctuation Restoration

Since the Punctuation Restoration models are also sequence tagging models, we evaluate them using Precision/Recall/F1 metrics on the same test set as the COMP-FT model (Section 3.3.1). Table 4 details the results of the three models on the test set for all the punctuation label groups. We also report metrics for sentence boundary detection, which is

a combination of the `PERIOD` and `QUESTIONMARK` labels. We observe that PUNCT-COMP-GEC was the best-performing model in most categories.

4.4 Comprehension

For COMP-FT, we evaluate various models between 240M and 1.3B parameters on our test set (Section 3.3.1) using BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005) and BLEURT (Sellam et al., 2020) as evaluation metrics. The Pegasus (Zhang et al., 2020) model outperforms the other models (Table 6). However, we find that automated metrics were neither suitable nor reliable for evaluation, as they largely focus on n-gram-based overlap with references. Thus, we use human evaluations to measure the quality of our comprehension models.

We conduct extensive human annotation studies to gather insight into the quality of the output generated by the comprehension models. First, we compare COMP-FT and COMP-LLM on various closed-ended composition scenarios using 1200 examples. We restrict this dataset to closed-ended use cases since COMP-FT does not work well for open-ended use cases. For each example, we ask seven annotators to provide a binary judgment on fluency, coherence, naturalness, and coverage (descriptions to annotators provided in Appendix B) and decide the final judgment by majority voting. We also measure inter-annotator agreement using a simple percent agreement, as well as Cohen's κ (McHugh, 2012).

Table 5 shows the human evaluation results for the two models on the four metrics and the corresponding inter-annotator agreement scores. We find that outputs generated by COMP-LLM are more fluent than those from COMP-FT. This result is expected since LLMs are known to generate highly fluent text. Further, outputs generated by COMP-FT are marginally better than those from COMP-LLM on coherence and naturalness. Finally, we find that outputs generated by COMP-LLM have much more meaning-change than those from COMP-FT, highlighting a known problem of hallucination in LLMs. Overall, we find that in closed-ended inputs, the text generated by COMP-FT is overall of higher quality than that generated by COMP-LLM.

Table 5 also shows that Cohen's κ scores were higher for both models on Fluency and Coverage, indicating that annotators were more aligned on

these criteria than they were on Coherence and Naturalness. This confirms our understanding that grammar and the presence or absence of information are more objective, whereas Coherence and Naturalness are more subjective and may vary based on context (for example, a short message may be unnatural but perfectly acceptable as a quick reply). Even though these categories had lower κ scores, they are still in a range that is considered fair agreement.

4.4.1 Sensitivity Evaluations

Current text generation systems have been shown to contain bias and behave differently to sensitive text (Bender et al., 2021; Welbl et al., 2021; Hovy and Prabhumoye, 2021). Therefore, we conduct an iterative sensitivity review of our end-to-end pipeline. We prepare a dataset of 800 sensitive examples to test the generation quality on offensive and non-inclusive language, bias, meaning change, and sensitive domains (such as medical advice and self-harm). We reviewed the generated outputs for the sensitive inputs and after manual reviewing, made the following changes to mitigate the identified risks:

1. Apply dictionary-based filtering for offensive words and a sensitivity classifier[7] after both the normalization and comprehension stages.

2. Retrain COMP-FT on an improved dataset containing examples to handle sensitive text better, improved co-reference resolution, and diversity-based augmentations. For COMP-LLM, we evaluate prompts on their ability to handle sensitive text.

3. Adjust the classifier of the hybrid model to send more sensitive data to COMP-FT instead of COMP-LLM.

Overall, we find that COMP-FT is much better at handling sensitive text compared to COMP-LLM.

4.5 Inference

We deploy our service on Amazon ECS using the `g5.2xlarge` instances. To increase the throughput while reducing overall latency, we enable our service to scale horizontally as well as run multiple inference workers per instance. We conduct

---

[7]DistilBERT (Sanh et al., 2019) trained on DeTexD dataset (Yavnyi et al., 2023)

Model	Sentence	Comma	Period	Question
rpunct	91.0 / 88.3 / 89.6	72.5 / 42.8 / 53.8	83.0 / 91.4 / 87.0	78.4 / 81.0 / 79.7
Punct-Comp	**94.3 / 92.7 / 93.5**	80.2 / 78.5 / 79.3	90.6 / 87.6 / 89.1	**89.0** / 83.5 / **86.2**
Punct-Comp-GEC	**94.3** / 92.1 / 93.2	**80.7 / 93.9 / 86.8**	**93.9 / 92.5 / 93.2**	78.6 / **92.6** / 85.0

Table 4: Performance comparison of different Punctuation Restoration models (Precision / Recall / F1).

System	Fluency (%)↑	Coherence (%)↑	Naturalness (%)↑	Coverage (%)↑
COMP-FT	66.49 (0.61/83.47)	**88.47** (0.31/83.35)	**76.93** (0.43/78.80)	**83.31** (0.51/85.32)
COMP-LLM	**68.93** (0.57/82.40)	86.02 (0.35/82.98)	75.98 (0.45/78.82)	68.25 (0.55/80.56)

Table 5: Human Evaluation of different comprehension models on Fluency, Coherence, Naturalness, and Coverage. The number in brackets shows Cohen's κ and inter-annotator agreement scores, respectively.

load testing to evaluate the infrastructure costs required for deploying the system. We find that we can successfully serve a constant traffic of 1 request per second using the COMP-FT model on a single `g5.2xlarge` instance while maintaining a p90 latency of 3 seconds. To achieve the same latency and throughput requirements for COMP-LLM, we need to scale the number of instances to 30. With a hybrid system that routes the request to either COMP-FT or COMP-LLM, we can reduce the number of instances to 10.

5 Conclusion

In this paper, we presented Speakerly™, a real-time voice-based writing assistant for text composition. It provides a low barrier to entry into the writing process, where a user can interact naturally, either using dictation, instructions, or unstructured thoughts. In turn, it generates a high-quality first draft with low latency, thus, providing them with a simple and efficient way to articulate their thoughts into ready-to-send emails, messages, or notes. We present comprehensive technical details of the different stages of the pipeline and experiments which guided our decisions while deploying the system to our users.

Limitations

While we design Speakerly™to handle the various challenges that can occur in real-world spoken input, there are instances where the system can generate output that does not reflect what the user wanted to say or generate sensitive text. In such cases, the user can either ask the system to regenerate the output, speak again, or manually edit the generated output. Since manually editing the system can be tedious, we plan to integrate a text editing step in the pipeline. Furthermore, our system currently cannot generate very long outputs (greater than 512 tokens). Currently, for most open-ended inputs, we rely on an external LLM, which can be costly and have high latency. Moving forward, we intend to look at other smaller models that can generate high-quality outputs for such texts. Lastly, since we use external ASR systems, which can be limited in their ability to deal with different accents, our system's ability can be limited by it (even though we do have augmentations to mimic such inputs). Finally, we only tested this system for English.

Ethics Statement

During the data annotation and model evaluation processes, all human evaluators' identities were anonymized to respect their privacy rights. All human evaluators received a fair wage higher than the minimum wage based on the number of data points they evaluated.

Although we implement ways to mitigate risks associated with sensitive texts, there can still be instances where the model can generate some sensitive output or cause meaning change and hallucination, especially for open-ended inputs. We do give users options to give feedback and report such issues, which we plan to keep improving the system on (using signals such as social factors, for instance, (Kulkarni and Raheja, 2023)).

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow.

Tuhin Chakrabarty, Vishakh Padmakumar, and He He. 2022. Help me write a poem: Instruction tuning as a vehicle for collaborative poetry writing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6848–6863, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A Smith. 2018. Creative writing with a machine in the loop: Case studies on slogans and stories. In *23rd International Conference on Intelligent User Interfaces*, pages 329–340.

Andy Coenen, Luke Davis, Daphne Ippolito, Emily Reif, and Ann Yuan. 2021. Wordcraft: a human-ai collaborative editor for story writing. *arXiv preprint arXiv:2107.07430*.

P R Cohen and S L Oviatt. 1995. The role of voice input for human-machine communication. *Proceedings of the National Academy of Sciences*, 92(22):9921–9927.

Kaustubh D Dhole, Varun Gangal, Sebastian Gehrmann, Aadesh Gupta, Zhenhao Li, Saad Mahamood, Abinaya Mahendiran, Simon Mille, Ashish Shrivastava, Samson Tan, et al. 2021. Nl-augmenter: A framework for task-sensitive natural language augmentation. *arXiv preprint arXiv:2112.02721*.

Wanyu Du, Zae Myung Kim, Vipul Raheja, Dhruv Kumar, and Dongyeop Kang. 2022. Read, revise, repeat: A system demonstration for human-in-the-loop iterative text revision. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, pages 96–108, Dublin, Ireland. Association for Computational Linguistics.

Jiayue Fan, Chenning Xu, Chun Yu, and Yuanchun Shi. 2021. Just speak it: Minimize cognitive load for eyes-free text editing with a smart voice assistant. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, pages 910–921.

Linda Flower and John R. Hayes. 1981. A cognitive process theory of writing. *College Composition and Communication*, 32(4):365–387.

Katy Gero, Alex Calderwood, Charlotte Li, and Lydia Chilton. 2022a. A design space for writing support tools using a cognitive process model of writing. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, pages 11–24, Dublin, Ireland. Association for Computational Linguistics.

Katy Gero, Vivian Liu, and Lydia Chilton. 2022b. Sparks: Inspiration for science writing using language models. In *Proceedings of the First Workshop on Intelligent and Interactive Writing Assistants (In2Writing 2022)*, pages 83–84, Dublin, Ireland. Association for Computational Linguistics.

Katy Ilonka Gero and Lydia B Chilton. 2019. Metaphoria: An algorithmic companion for metaphor creation. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–12.

Debjyoti Ghosh. 2020. *Voice-based Interactions for Editing Text On The Go*. Ph.D. thesis, National University of Singapore (Singapore).

J.J. Godfrey, E.C. Holliman, and J. McDaniel. 1992. Switchboard: telephone speech corpus for research and development. In *[Proceedings] ICASSP-92: 1992 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 517–520 vol.1.

Aditya Gupta, Jiacheng Xu, Shyam Upadhyay, Diyi Yang, and Manaal Faruqui. 2021. Disfl-QA: A benchmark dataset for understanding disfluencies in question answering. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3309–3319, Online. Association for Computational Linguistics.

Dirk Hovy and Shrimai Prabhumoye. 2021. Five sources of bias in natural language processing. *Language and Linguistics Compass*, 15(8):e12432.

J. Edward Hu, Abhinav Singh, Nils Holzenberger, Matt Post, and Benjamin Van Durme. 2019. Large-scale, diverse, paraphrastic bitexts via sampling and clustering. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 44–54, Hong Kong, China. Association for Computational Linguistics.

Julie S Hui, Darren Gergle, and Elizabeth M Gerber. 2018. Introassist: A tool to support writing introductory help requests. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Paria Jamshid Lou and Mark Johnson. 2020. Improving disfluency detection by self-training a self-attentive model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3754–3763, Online. Association for Computational Linguistics.

C Kamm. 1995. User interfaces for voice applications. *Proceedings of the National Academy of Sciences*, 92(22):10031–10037.

Zae Myung Kim, Wanyu Du, Vipul Raheja, Dhruv Kumar, and Dongyeop Kang. 2022. Improving iterative text revision by learning where to edit from other revision tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9986–9999, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Vivek Kulkarni and Vipul Raheja. 2023. Writing assistants should model social factors of language.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yuhan Luo, Young-Ho Kim, Bongshin Lee, Naeemul Hassan, and Eun Kyoung Choe. 2021. Foodscrap: Promoting rich data capture and reflective food journaling through speech input. In *Designing Interactive Systems Conference 2021*, pages 606–618.

Yuhan Luo, Bongshin Lee, and Eun Kyoung Choe. 2020. Tandemtrack: shaping consistent exercise experience by complementing a mobile app with a smart speaker. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. EdiT5: Semi-autoregressive text editing with t5 warm-start. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2126–2138, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Mary L McHugh. 2012. Interrater reliability: the kappa statistic. *Biochemia medica*, 22(3):276–282.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Tatiana Passali, Thanassis Mavropoulos, Grigorios Tsoumakas, Georgios Meditskos, and Stefanos Vrochidis. 2022. LARD: Large-scale artificial disfluency generation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2327–2336, Marseille, France. European Language Resources Association.

Zhenhui Peng, Qingyu Guo, Ka Wing Tsang, and Xiaojuan Ma. 2020. Exploring the effects of technological writing assistance for support providers in online mental health community. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–15.

Martin Porcheron, Joel E. Fischer, Stuart Reeves, and Sarah Sharples. 2018. Voice interfaces in everyday life. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, page 1–12, New York, NY, USA. Association for Computing Machinery.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Filip Radlinski, Krisztian Balog, Bill Byrne, and Karthik Krishnamoorthi. 2019. Coached conversational preference elicitation: A case study in understanding movie preferences. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 353–360, Stockholm, Sweden. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Vipul Raheja, Dhruv Kumar, Ryan Koo, and Dongyeop Kang. 2023. Coedit: Text editing by task-specific instruction tuning. *arXiv preprint arXiv:2305.09857*.

Sherry Ruan, Jacob O. Wobbrock, Kenny Liou, Andrew Ng, and James A. Landay. 2018. Comparing speech

and keyboard text entry for short messages in two languages on touchscreen phones. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 1(4).

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Johan Schalkwyk, Doug Beeferman, Françoise Beaufays, Bill Byrne, Ciprian Chelba, Mike Cohen, Maryam Kamvar, and Brian Strope. 2010. "your word is my command": Google search by voice: A case study. *Advances in Speech Recognition: Mobile Environments, Call Centers and Clinics*, pages 61–90.

Timo Schick, Jane A. Yu, Zhengbao Jiang, Fabio Petroni, Patrick Lewis, Gautier Izacard, Qingfei You, Christoforos Nalmpantis, Edouard Grave, and Sebastian Riedel. 2023. PEER: A collaborative language model. In *The Eleventh International Conference on Learning Representations*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

E.E. Shriber. 1994. *Preliminaries to a Theory of Speech Disfluencies*. University of California, Berkeley.

Arjun Srinivasan, Bongshin Lee, Nathalie Henry Riche, Steven M Drucker, and Ken Hinckley. 2020. Inchorus: Designing consistent multimodal interactions for data visualization on tablet devices. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–13.

Lisa J Stifelman, Barry Arons, Chris Schmandt, and Eric A Hulteen. 1993. Voicenotes: A speech interface for a hand-held voice notetaker. In *Proceedings of the INTERACT'93 and CHI'93 conference on Human factors in computing systems*, pages 179–186.

Johannes Welbl, Amelia Glaese, Jonathan Uesato, Sumanth Dathathri, John Mellor, Lisa Anne Hendricks, Kirsty Anderson, Pushmeet Kohli, Ben Coppin, and Po-Sen Huang. 2021. Challenges in detoxifying language models. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2447–2469, Punta Cana, Dominican Republic. Association for Computational Linguistics.

James R Williams. 1998. Guidelines for the use of multimedia in instruction. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 42, pages 1447–1451. SAGE Publications Sage CA: Los Angeles, CA.

Serhii Yavnyi, Oleksii Sliusarenko, Jade Razzaghi, Olena Nahorna, Yichen Mo, Knar Hovakimyan, and Artem Chernodub. 2023. DeTexD: A benchmark dataset for delicate text detection. In *The 7th Workshop on Online Abuse and Harms (WOAH)*, pages 14–28, Toronto, Canada. Association for Computational Linguistics.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, et al. 2020. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

A COMP-FT Training Details

For COMP-FT, we evaluate nine models - ProphetNet (Qi et al., 2020), GPT2 (Radford et al., 2019), Megatron-GPT2 (Shoeybi et al., 2019), BART-large (Lewis et al., 2020), Pegasus-Base (Zhang et al., 2020), BigBird-Pegasus-A (Zaheer et al., 2020), Pegasus-Large (Zhang et al., 2020), T5-Large (Raffel et al., 2020), and GPT3-Neo (Black et al., 2021), between 240M and 1.3B parameters on our test set containing 1k examples. Table 6 shows the models, their respective parameters and their performance using BLEU, ROUGE-1, ROUGE-2, ROUGE-L, METEOR, and BLEURT metrics. We find that overall the Pegasus family of models performs better than the other models.

For our system, we, therefore, fine-tune Pegasus on our training set composed of 28k input-output pairs handling different use cases. For fine-tuning the model, we use a single NVIDIA V100 GPU for 30 epochs using a learning rate of $1e - 4$ and a batch size of 16. It takes around 16 hours for the model to train. For all our experiments, we use the maximum token sequence length of 512 on both the encoder and decoder.

B Human Evaluation for comprehension model metrics description

While conducting the human evaluation for the comprehension models, we ask the animators to make a binary decision on Fluency, Coherence, Naturalness and Coverage. Below we provide the definitions we provided to the annotators.

Fluency The generated output should be correct with respect to grammar and word choice, including spelling. It should have no datelines, headers, system-internal formatting, capitalization errors, or ungrammatical sentences (e.g., fragments, missing components) that make the text difficult to read.

Coherence The generated output should be well structured and well organized. It should not just be a heap of related information or a collection of sentences but should build from sentence to sentence to a well-organized, naturally flowing, coherent body of information.

Naturalness The generated output should use natural phrasing and maintain the appropriate tone and level of formality given its content (e.g., the implied relationship between sender and recipient, the topic, etc.).

Coverage The generated output should adequately verbalize the information present in the input. Coverage of all details of the most significant details is desired in the generated output.

C Augmentations for training data

Our system consists of a pipeline of ML models that progressively refines the input at each stage. However, some stages may introduce new errors or fail to fix the errors that they were supposed to fix. The comprehension model is the last stage of the pipeline, and it must address the remaining issues or the new issues introduced by the earlier stages of the pipeline. Therefore, to introduce these capabilities in the comprehension model, we add augmentations to the training dataset of the comprehension model.

While preparing the training dataset for fine-tuning the COMP-FT model, we generate new training examples by adding augmentations to the input and output of the initial dataset prepared by human annotators. Table 7 shows some of the augmentations we apply. It consists of three columns, showing the augmentation type, the issue it addresses, and its definition. We have four categories in the types of issues we address:

ASR issues: These are issues that were caused by the ASR system, such as incorrectly transcribing a word with its homophone, i.e., similar sounding word.

Normalization issues: These are issues that were caused due to issues in the normalization stages, such as missing inserting the correct

punctuations or not removing the filler words.

User input issues: These are issues that were present in the user speech and were not handled by the earlier models in the pipeline, such as repetition of information or incomplete information in the input.

Sensitivity issues: These are issues that we found during our sensitivity reviews, such as the model behaving differently if a non-western name is present in the input.

Model	Size	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	BLEURT
ProphetNet	240M	40.62	70.21	48.51	62.78	57.92	0.04
GPT2	345M	46.68	73.23	56.14	68.41	64.45	0.13
Megatron-GPT2	345M	43.71	72.07	50.8	65.26	61.77	0.12
BART-large	406M	49.63	72.25	53.72	67.38	66.47	0.21
Pegasus-Base	568M	**55.98**	73.3	57.47	73.05	68.95	0.25
BigBird-Pegasus-A	576M	45.92	72.6	52.33	66.37	63.1	**0.49**
Pegasus-Large	770M	53.06	**79.33**	**62.93**	**74.87**	**69.71**	0.26
T5-Large	770M	54.23	78.81	60.03	72.27	68.82	0.24
GPT3-Neo	1.3B	45.58	73.2	52.8	67.03	63.13	0.13

Table 6: Performance comparison of different fine-tuned Comprehension models on automated metrics.

Augmentation	Issue	Definition
Homophones	ASR	Swap random words in the input with their homophones.
Filler words addition	Normalization	Randomly insert filler words such as uh, um, etc. in the input.
Removing periods and commas	Normalization	Randomly joining sentences by removing the period punctuation in the input. This also helps in making the model learn to generate not very-long sentences.
Content repetition	Noisy User Input	Repeat random words, phrases and sentences in the input so that the model learns to remove repeated information.
Random word removal/addition	Noisy User Input	Randomly add or remove certain words/phrases (non-entity) in the input so that the model can learn to deal with such noises.
Sentence shuffle	Noisy User Input	Change the order of certain sentences in the input so that the model can learn to deal with incoherent input.
Gender-neutral rewrite	Sensitivity	Rewrite both the inputs and outputs to a gender-neutral version so that the model does not behave differently for such cases.
Name and date change	Sensitivity	Randomly modify names to non-western names in both input and output so that the model does not behave differently for such cases.

Table 7: A subset of the augmentations used to add more examples to the training data.