Langsmith: An Interactive Academic Text Revision System

Takumi Ito[*,1,2] , **Tatsuki Kuribayashi**[*,1,2], **Masatoshi Hidaka**[*,3],
Jun Suzuki[1,4], **and Kentaro Inui**[1,4]
[1]Tohoku University [2]Langsmith Inc. [3]Edge Intelligence Systems Inc. [4]RIKEN
{t-ito, kuribayashi, jun.suzuki, inui}@ecei.tohoku.ac.jp
hidaka@edgeintelligence.jp

Abstract

Despite the current diversity and inclusion initiatives in the academic community, researchers with a non-native command of English still face significant obstacles when writing papers in English. This paper presents the *Langsmith* editor, which assists inexperienced, non-native researchers to write English papers, especially in the natural language processing (NLP) field. Our system can suggest fluent, academic-style sentences to writers based on their rough, incomplete phrases or sentences. The system also encourages interaction between human writers and the computerized revision system. The experimental results demonstrated that Langsmith helps non-native English-speaker students write papers in English. The system is available at https://emnlp-demo.editor.langsmith.co.jp/.

Figure 1: An overview of interactively writing texts with a revision system.

1 Introduction

Currently, diversity and inclusion in the natural language processing (NLP) community are encouraged. In fact, at the latest NLP conference at the time of writing[1], papers were submitted from more than 50 countries. However, one obstacle can limit this diversity: *The papers must be written in English.* Writing papers in English can be a daunting task, especially for inexperienced, non-native speakers. These writers often struggle to put their ideas into words.

To address this problem, we built the *Langsmith* editor, an assistance system for writing NLP papers in English.[2] The main feature in Langsmith is a revision function, which suggests fluent, academic-style sentences based on writers' rough, incomplete drafts.

The drafts might be so rough that it becomes challenging to understand the user's intended meaning to use as inputs. In addition, several potentially plausible revisions can exist for the drafts, especially when the input draft is incomplete.

Based on such difficulties, our system provides two ways for users to customize the revision: the users can (i) request specific revisions, and (ii) select a suitable revision from diverse candidates (Figure 1). In particular, the request stage allows users to specify the parts that require intensive revision.

Our experiments demonstrate the effectiveness of our system. Specifically, students whose first language is Japanese, which differs greatly from English, managed to write better drafts when working with Langsmith.

Langsmith has other assistance features as well, such as text completion with a neural language

---

* The authors contributed equally
[1]The 58th Annual Meeting of the Association for Computational Linguistics
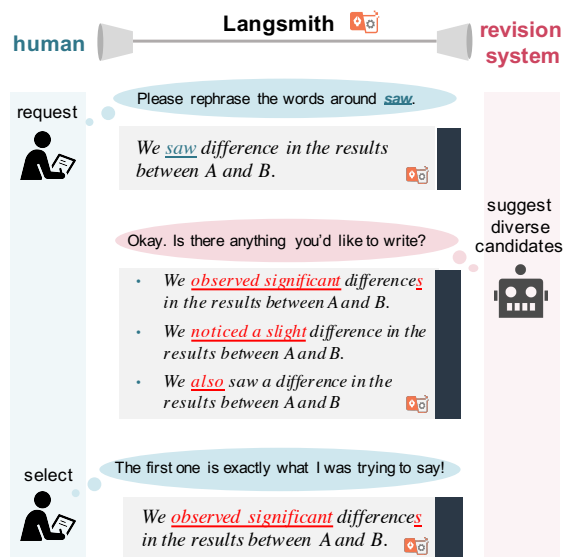[2]See https://www.youtube.com/channel/UCjHeZPe0tT6bWxVVvum1bFQ for the screencast.
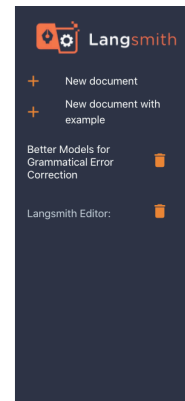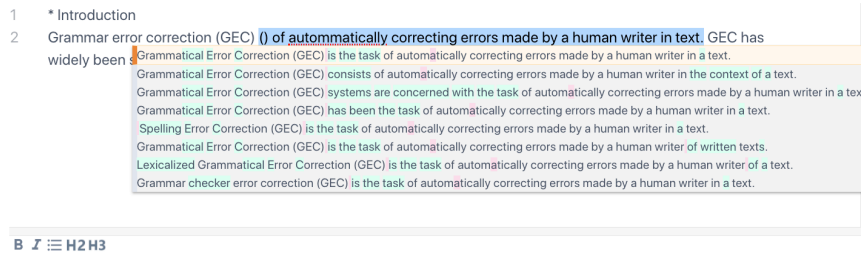
Figure 2: Screenshot of Langsmith. The revision feature suggests various revisions for the input "*Grammar error correction (GEC) () of automatically correcting errors made by a human writer in text.*" The characters highlighted in green are added to the original sentence, and the red points indicate tracked deletions.

model. Furthermore, the communication between the server and the web frontend is achieved via a protocol specialized in writing software called the Text Editing Assistance Smartness Protocol for Natural Language (TEASPN) (Hagiwara et al., 2019). We hope that our system will help the NLP community and researchers, especially those lacking a native command of English.[3]

2 Related work

2.1 Natural language processing for academic writing

Academic writing assistance has gained considerable attention in NLP (Wu et al., 2010; Yimam et al., 2020; Lee and Webster, 2012), and several shared tasks have been organized (Dale and Kilgarriff, 2011; Daudaravičius, 2015). These tasks focus on polishing texts in already published articles or documents near completion. In contrast, this study focuses on revising texts in the earlier stages of writing (e.g., first drafts), where inexperienced, non-native authors might even struggle to convey their ideas accurately.

Ito et al. (2019) introduced a dataset and models for revising early-stage drafts, and the 1-to-N nature of the revisions was pointed out. We tackled this difficulty by designing an overall demonstration system, including a user interface.

2.2 Writing assistance tools

Error checkers. Grammar/spelling checkers are typical writing assistance tools. Some highlight errors (e.g., *Write&Improve*[4]), while others suggest

corrections (e.g., *Grammarly*[5], *LanguageTool*[6], *Ginger*[7], and *LinggleWrite*; see Tsai et al. (2020)) for writers.

Langsmith has a revision feature (Ito et al., 2019), as well as a grammar/spelling checker. The revision feature suggests better versions of poor written phrases or sentences in terms of fluency and style, whereas error checkers are typically designed to correct apparent errors only. In addition, Langsmith is specialized for the NLP domain and enables domain-specific revisions, such as correcting technical terms.

Text completion. Completing a text is another typical feature in writing assistance applications (*WriteAhead*[8], *Write With Transformer*[9], and *Smart Compose*; see Chen, Mia Xu and Lee, Benjamin N. and Bansal, Gagan and Cao, Yuan and Zhang, Shuyuan and Lu, Justin and Tsay, Jackie and Wang, Yinan and Dai, Andrew M. and Chen, Zhifeng and Sohn, Timothy and Wu, Yonghui (2019)). Our system also has a completion feature, which is specialized in academic writing (e.g., completing a text based on a section name).

3 The *Langsmith* editor

3.1 Overview

This section presents Langsmith, a web-based text editor for academic writing assistance (Figure 2). The system has the following three features: (i) text revision, (ii) text completion, and (iii) a gram-

---

[3]This paper was also written using Langsmith.
[4]writeandimprove.com

[5]https://www.grammarly.com
[6]https://languagetool.org
[7]https://www.gingersoftware.com
[8]writeahead.nlpweb.org
[9]https://transformer.huggingface.co

matical/spelling error checker. These features are activated when users select a text span, type a word, or push a special key.

As a case study, this work focuses on paper writing in the NLP domain. Thus, each assistance feature is specialized in the NLP domain. The following sections explain the details of each feature.

3.2 Revision feature

The revision feature, the main feature of Langsmith, suggests better sentences in terms of fluency and style for a given draft sentence (Figure 2). This feature is activated when the user selects a sentence or smaller unit.

Writers sometimes struggle to put their ideas into words. Thus, the input draft for the revision systems can be incomplete, or less informative. Based on such a challenging situation, we examine the REQUEST and SELECT framework to help users discover sentences that better match what the user wanted to write.

REQUEST stage. Langsmith provides two ways for users to request a specific revision, which can prevent unnecessary revisions being provided to the user.

First, users can specify where the system should intensively revise a text.[10] That is, when a part of a sentence is selected, the system intensively rephrases the words around the selected part.[11] Figure 3 demonstrates the change of the revision focus, depending on the selected text span. Note that controlling the revision focus was not explored in the original sentence-level revision task (Ito et al., 2019). This feature is also inspired by Grangier and Auli (2018).

Second, users can insert placeholder symbols, " () ", at specific points in a sentence. The system revises the sentence by replacing the symbol with an appropriate expression regarding its context. The input for the revision in Figure 2 also has the placeholder symbol. Here, for example, the symbol is replaced with "the task." This feature is inspired by Zhu et al. (2019); Donahue et al. (2020); Ito et al. (2019).

SELECT stage. The system provides several revisions (Figure 2). Note that there is typically more

---

(a) Revisions focusing on *This formulation · · · and output.*

(b) Revisions focusing on *promote.*

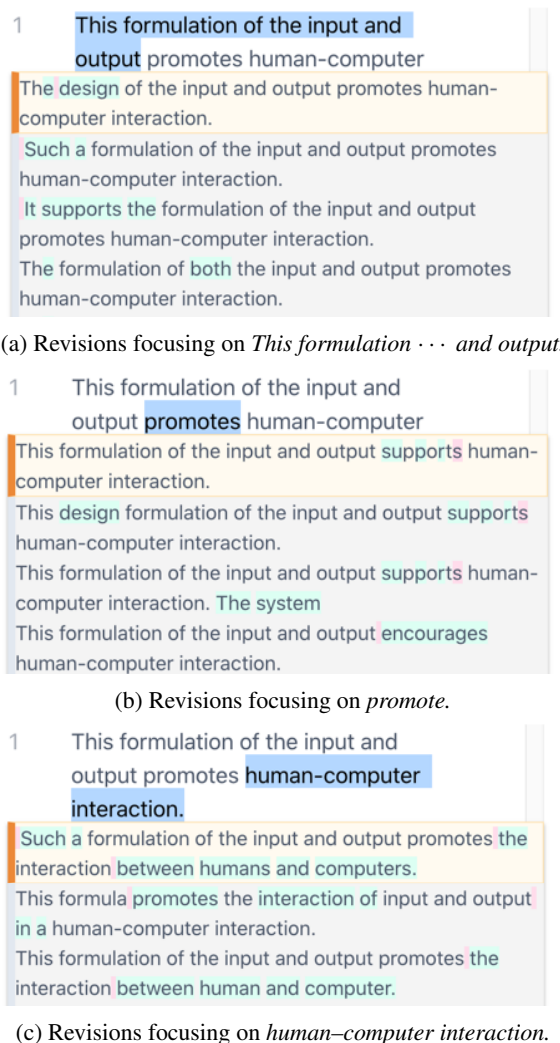(c) Revisions focusing on *human–computer interaction.*

Figure 3: The focus of the revision depends on the parts selected by users.

than one plausible revision in terms of fluency and style, in contrast to correcting surface-level errors (Napoles et al., 2017).

The diversity of the output revisions is encouraged using diverse beam search (Vijayakumar et al., 2018). In addition, these revisions are ordered by a language model that is fine-tuned for NLP papers. That is, revisions with lower perplexity are listed in the upper part of the suggestion box. Furthermore, the revisions are highlighted in colors, which makes it easier to distinguish the characteristics of each revision.

Implementation. We trained a revision model using LightConv (Wu et al., 2019) implemented in Fairseq (Ott et al., 2019). The revision model generates a sentence based on a given input sentence. The model was trained on a slightly modified version of the synthetic training data used in Ito et al.

---

[10]The system performs sentence-level revisions. Hence the users are instructed to select the non-sentence-crossing area.

[11]We allow the system to correct the parts outside the selected span because sometimes the revision for a specific part requires another adjustment for the other parts.
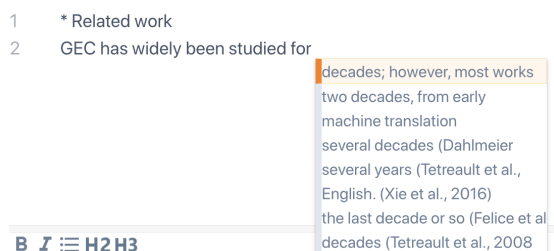
Figure 4: An example of the completion feature. These suggestions are conditioned by the left context, section name (*Related work*) and the paper title (*Better Models for Grammatical Error Correction.*)
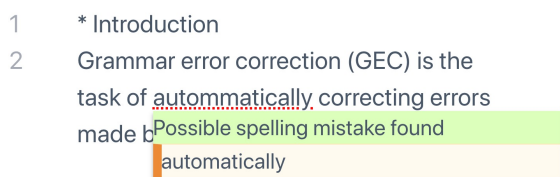

Figure 5: The interface of the error correction feature. Errors are automatically highlighted with a red line. The corrections are suggested when the user hovers over the highlighted words.

(2019). As an example of these modifications, synthetic edit marks were added for a subset of the training data. These marks were attached to a part of the input sentence that has many edits compared to its reference.[12] Thus, the marks can provide a hint for the system to determine where to edit. When using Langsmith, the marks are attached to the span selected by the users. The system is expected to intensively revise the wording in the specified span. Details are in Appendix A.

3.3 Other features

Completion feature. When the user presses the `Tab` key, the completion feature generates plausible preceding phrases from the cursor point (Figure 4). This feature can consider the paper title and section name as well as the text to the left of the cursor.

We used GPT-2 small (117M) (Radford et al., 2019) fine-tuned on the papers collected from the ACL Anthology[13]. Paper titles and section names were concatenated at the beginning of the corresponding paragraphs in the fine-tuning data. De-

tails are in Appendix B.

Error correction feature. We used Language-Tool,[14] an open-source grammatical/spelling error correction tool. Each time the text changes, this feature is called upon. The detected errors are then automatically highlighted with red lines (Figure 5).The corrections are listed when the user hovers over the highlighted words.

3.4 Protocol

Langsmith was developed based on the TEASPN Software Development Kit (Hagiwara et al., 2019).[15] TEASPN defines a set of APIs for writing software (e.g., text editors) to communicate with servers that implement NLP technologies (e.g., revision model). We extended the protocol to convey title and section information in the completion feature. Since Langsmith is a browser-based tool and frequently communicates with a web server running models, we used WebSocket to achieve smooth communication.

4 Experiments and results

We demonstrate the effectiveness of human–machine interactions in revising drafts implemented in our system. We also check whether the REQUEST stage in the revision feature works adequately.

4.1 On the revised draft quality

Settings. We suppose a situation where a person writes a draft in their native language (non-English language), translates it to English, and then revises it further to create an English-language draft. In order to simulate this situation, we first collected Japanese-language version of the abstract sections from eight Japanese peer-reviewed journals.[16] Then, the abstracts were translated into English with an off-the-shelf translation system[17]. We considered the translated abstracts as first drafts. The task is to revise the first drafts. Expert translators created reference final drafts from the Japanese versions of the drafts.[18] We evaluated the quality of the revised versions by comparing them with the corresponding final drafts.

---

[12]Special symbols are attached at the beginning and the end of the specific subsequence.

[13]https://www.aclweb.org/anthology

[14]https://github.com/languagetool-org/languagetool/releases/tag/v3.2

[15]https://github.com/teaspn/teaspn-sdk

[16]We used the journals accepted at https://www.anlp.jp/en/index.html.

[17]https://translate.google.co.jp

[18]We used https://www.ulatus.com/.

Condition	BLEURT
HUMAN&MACHINE	**-0.45**
HUMAN-ONLY	-0.51
MACHINE-ONLY	-0.51
First drafts	-0.70

Table 1: Comparison of the revision quality. The scores are averaged over the corresponding revisions. Higher scores indicate that the drafts are closer to the final drafts.

Q.	Strongly agree	Slightly agree	Slightly disagree	Strongly disagree
(I)	87.5	12.5	0	0
(II)	50.0	50.0	0	0
(III)	62.5	31.3	6.3	0
(IV)	12.5	50.0	31.3	6.3
(V)	75.0	12.5	6.3	6.3
(VI)	43.8	43.8	12.5	0

Table 2: Results of the user study about (I)-(VI). The scores denote the percentage of the participants who chose the option.

We compared three versions of revised drafts to evaluate the effectiveness of Langsmith:

- one fully and automatically revised by Langsmith (MACHINE-ONLY revision)
- one revised by a human writer without Langsmith (HUMAN-ONLY revision), and
- one revised by a human writer using assistance features in Langsmith (HUMAN&MACHINE revision).

The following paragraphs explain how we obtained the above three versions of the revisions. Appendix C shows the statistics of the drafts.

MACHINE-ONLY revision. We automatically applied the revision feature to the drafts (each sentence) without the REQUEST and Select stages. For each sentence, the revision with the highest generation probability was selected.[19] We created one MACHINE-ONLY revision for each first draft.

HUMAN-ONLY revision. Human writers revise a given first draft. The writers can only access to the error correction feature. This setting simulates the situations that writers typically face.

HUMAN&MACHINE revision. Human writers revise a given first draft with full access to the Langsmith features.

Human writers. We asked 16 undergraduate and master's students at an NLP laboratory to revise the first drafts in terms of fluency and style. The students were Japanese natives, representatives of the inexperienced researchers in a country where the spoken language is considerably different from English. Each participant revised two different first drafts, one with the HUMAN-ONLY setting and the other one with the HUMAN&MACHINE setting.

Half of the participants first revised a draft with the HUMAN-ONLY setting, and then revised another draft with the HUMAN&MACHINE setting; the other half performed the same task in the opposite order. Ultimately, we collected two HUMAN&MACHINE revisions and two HUMAN-ONLY revisions for each first draft.

Comparison and results. We compared the quality of the three versions of the revised drafts: MACHINE-ONLY revision, HUMAN-ONLY revision, and HUMAN&MACHINE revision. We compared the revised drafts with their corresponding final draft using BLEURT (Sellam et al., 2020), the state-of-the-art automatic evaluation metric for natural language generation tasks. Details of the evaluation procedure is shown in Appendix D. Note that the score is not in the range $[0, 1]$, and a higher score means that the revision is closer to the final draft. Table 1 shows that HUMAN&MACHINE revisions were significantly better[20] than MACHINE-ONLY and HUMAN-ONLY revisions. The results suggest the effectiveness of human–machine interaction achieved in Langsmith. Since this experiment was relatively small in scale and only used an automatic evaluation metric, we will conduct a larger-scale experiment with human evaluations in the future.

4.2 User study

After the experiments outlined in Section 4.1, we asked the participants about the usability of Langsmith. The 16 participants were instructed to evaluate the following statements:

(I) Langsmith was more helpful than the Baseline environment for the revision task.

---

[19]The hyperparameters for decoding revisions were the same as the revision feature in Langsmith. Re-ranking with the language model was also employed.

[20]We applied a bootstrap hypothesis test (Koehn, 2004), and the score of HUMAN&MACHINE was significantly higher than the HUMAN-ONLY and MACHINE-ONLY scores ($p < 0.05$).

Feature	percentage
revision	100
completion	31.3
correction	62.5

Table 3: Results of the user study about helpful features. The scores denote the percentage of the participants who chose the feature (multiple choice question).

(II) Comparing the text written by the two environments, the text written with Langsmith was better.
(III) The feature of specifying where to intensively revise was helpful.
(IV) The placeholder feature in the revision feature was helpful.
(V) Providing more than one output from the revision feature was helpful.
(VI) Providing more than one output from the completion feature was helpful.

The participants evaluated the statements (I)-(VI) on a four-point scale: (a) strongly agree, (b) slightly agree, (c) slightly disagree, and (d) strongly disagree. In addition, the participants answered whether each feature was helpful in writing.

Results. Tables 2 and 3 show the results of our user study. From the responses to (I) and (II), we observed that the users were satisfied with the writing experience with Langsmith. The responses to (III), (IV), and (V) support the idea that our REQUEST and SELECT stages are helpful. Here, using the place holders was relatively not helpful. The responses to (VI) also suggest that showing several candidates does not bother the users. Table 3 displays the result of whether each feature was helpful in writing. The result indicates that the revision feature was the most useful for creating drafts using the implemented features.

4.3 Sanity check of the REQUEST stage

Finally, we checked the validity of our method to control the revision based on the selected part of the sentence (Figure 3).

Settings. We randomly collected 1,000 sentences from the first drafts created with the translation system. In each sentence with T tokens $x = (w_1, \cdots, w_T)$, we randomly inserted edit marks to specify a certain span $s = (i, j)$ in x ($1 \leq i < j \leq T, 1 \leq j - i \leq 5$). Specifically, special to-

kens were inserted before w_i and after w_j in x. We denote the input sentence with these edit marks as x^{edit}. We then obtained 10-best outputs of the revision system $(y_1^{\text{edit}}, \cdots, y_{10}^{\text{edit}})$ for each x^{edit}. Here, these output sentences were generated through the diverse beam search with the same settings as the revision feature in Langsmith. We calculated the following score for each input sentence and its revisions:

$$r = |\{y_k^{\text{edit}} \mid x_{i:j} \in \text{ngram}(y_k^{\text{edit}}), 1 \leq k \leq 10\}|$$

where $x_{i:j}$ denotes the subsequence $(w_i, \cdots, w_j)$ in x. The function ngram$(\cdot)$ returns a set of all the n-grams of a given sequence. A lower r indicates that the subsequence specified with the edit marks are more frequently rephrased.

We also obtained a score r' for each x. r' was calculated using the input without the edit marks x and its 10-best outputs y_k. We compared r and r' for each x.

Results. We observed that r frequently[21] had lower values than r'. That is, a certain subsequence was more rephrased by the revision system when it had the edit marks than when it did not. These results validate our approach of controlling the revision focus, which is implemented in the REQUEST stage of the revision feature.

5 Conclusions

We have presented Langsmith, an academic writing assistance system. Langsmith provides a writing environment, in which human writers use several assistance features to improve the quality of texts. Our experiments suggest that our system is useful for inexperienced, non-native writers in revising English-language papers. We are aware that our experimental settings were not fully well-designed (e.g., we had only Japanese participants, and no human evaluation). We will evaluate Langsmith in more sophisticated settings. We hope that our system contributes to breaking language barriers in the academic community.

---

[21]We conducted the one-side sign test. The difference is significant with $p \leq 0.05$.

References

Chen, Mia Xu and Lee, Benjamin N. and Bansal, Gagan and Cao, Yuan and Zhang, Shuyuan and Lu, Justin and Tsay, Jackie and Wang, Yinan and Dai, Andrew M. and Chen, Zhifeng and Sohn, Timothy and Wu, Yonghui. 2019. Gmail smart compose: Real-time assisted writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Miningg (KDD'19)*, page 2287–2295.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation (ENLG 2011)*, pages 242–249.

Vidas Daudaravičius. 2015. Automated evaluation of scientific writing: AESW shared task proposal. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2015)*, pages 56–63.

Chris Donahue, Mina Lee, and Percy Liang. 2020. Enabling language models to fill in the blanks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL 2020)*, pages 2492–2501. Association for Computational Linguistics.

David Grangier and Michael Auli. 2018. QuickEdit: Editing text & translations by crossing words out. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2018)*, pages 272–282.

Masato Hagiwara, Takumi Ito, Tatsuki Kuribayashi, Jun Suzuki, and Kentaro Inui. 2019. TEASPN: Framework and protocol for integrated writing assistance environments. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing: System Demonstrations (EMNLP-IJCNLP 2019)*, pages 229–234.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The Curious Case of Neural Text Degeneration. In *Proceedings of the 8th International Conference on Learning Representations (ICLR 2020)*.

Takumi Ito, Tatsuki Kuribayashi, Hayato Kobayashi, Ana Brassard, Masato Hagiwara, Jun Suzuki, and Kentaro Inui. 2019. Diamonds in the rough: Generating fluent sentences from early-stage drafts for academic writing assistance. In *Proceedings of the 12th International Conference on Natural Language Generation (INLG 2019)*, pages 40–53.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP 2004)*, pages 388–395.

John Lee and Jonathan Webster. 2012. A corpus of textual revisions in second language writing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 248–252.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2017)*, pages 229–234.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: System Demonstrations (NAACL 2019)*, pages 48–53.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

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 (ACL 2020)*, pages 7881–7892.

Chung-Ting Tsai, Jhih-Jie Chen, Ching-Yu Yang, and Jason S. Chang. 2020. LinggleWrite: a coaching system for essay writing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations (ACL 2020)*, pages 127–133.

Ashwin K. Vijayakumar, Michael Cogswell, Ramprasaath R. Selvaraju, Qing Sun, Stefan Lee, David J. Crandall, and Dhruv Batra. 2018. Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, pages 7371–7379.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. HuggingFace's Transformers: State-of-the-art Natural Language Processing. *arXiv preprint arXiv:1910.03771*.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019. Pay Less Attention with Lightweight and Dynamic Convolutions. In *Proceedings of the 7th International Conference on Learning Representations (ICLR 2019)*.

Jian-Cheng Wu, Yu-Chia Chang, Teruko Mitamura, and Jason S. Chang. 2010. Automatic Collocation Suggestion in Academic Writing. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics Conference Short Papers (ACL 2010)*, pages 115–119.

Seid Muhie Yimam, Gopalakrishnan Venkatesh, John Lee, and Chris Biemann. 2020. Automatic compilation of resources for academic writing and evaluating with informal word identification and paraphrasing system. In *Proceedings of the 12th Language Resources and Evaluation Conference (LREC 2020)*, pages 5896–5904.

Wanrong Zhu, Zhiting Hu, and Eric Xing. 2019. Text Infilling. *arXiv preprint arXiv:1901.00158*.

A Details on revision model

Data. We trained the revision model using the slightly modified version of the synthetic training data introduced in Ito et al. (2019). They created several types of synthetic training data with several noising methods; (i) heuristic noising method, (i) grammatical error generation, (iii) style removal, and (iv) entailed sentence generation. We used the data created by the heuristic noising method, style removal, and the entailed sentence generation for training the revision model. Note that we did not use the data generated by the grammatical error generation because grammatical error correction feature was implemented separately from the revision feature in Langsmith.

We attached the edit marks to the subpart of the training data generated by the style removal method. Let $x_{1:N} = (x_1, x_2, \cdots, x_N)$ and $y_{1:T} = (y_1, y_1, \cdots, y_M)$ be an input sentence with N tokens and its revision with M tokens, respectively. Here x was the synthetic draft sentence generated by the style removal method from y. The training dataset consists of the pairs of (x, y).

For each (x, y), we first determined if each word in x was rewritten compared to y. We assumed that a token $x_i \in x$ was rewritten if a token with the same lemma as x_i was not in $\{y_j | \max(0, i - 3) \leq j \leq \min(M, i + 3)\}$. Here we obtained a sequence $c \in \{0, 1\}^N$, where each element c_i corresponds to whether the token x_i was rewritten or not. If x_i was written in y, c_i is 1; otherwise c_i is 0. Then, we defined a score $r(c)$ for each (x, y) as follows:

$$r(c) = \frac{\sum_{i=1}^N c_i}{|c|}$$

where $|\cdot|$ returns the length of the vector. If $r(c) > 0.4$, we did not attach the edit marks.

When $r(c) \leq 0.4$, we obtained a span $s = (a, b)$ for x and c as follows:

$$\underset{(a,b)\in\mathcal{S}}{\mathrm{argmax}} \quad \sum_{i=a}^{b} c_i' - \sum_{i=0}^{a-1} c_i' - \sum_{i=b+1}^{N+1} c_i'$$

$$\text{where} \quad c_i' = \begin{cases} 10 & (c_i = 1) \\ 0 & (i = 0, N + 1) \\ -1 & (\text{otherwise}) \end{cases}$$

$$\mathcal{S} = \{(a, b) \mid a, b \in 1, \cdots, N, a \leq b\}$$

Based on the obtained $s = (a, b)$, we inserted `<?` before the token x_a, and `?>` after the token x_b. We included the data with special symbols added by such a procedure in the training data.

When the users select a subsequence of a sentence in Langsmith, the edit marks are attached to the input sentence. For example, if the user selects a span "promote" in the sentence "This formulation of the input and output promotes human-computer interaction.", the input to the revision feature is formatted as follows: `This formulation of the input and output <? promotes ?> human-computer interaction.`

Model. Table 4 shows the hyperparameters of the revision model. In the decoding phase, we used the diverse beam search (Vijayakumar et al., 2018). Beam size is set to 15. The diverse beam group and the diverse beam strength are 15 and 1.0, respectively.

Specifically, we first obtained top-15 hypotheses, and then these hypotheses were re-ranked by the language model. Here, the language model considers 20 tokens in the left context and 20 tokens in the right context beyond the sentence. We excluded the hypotheses with a perplexity greater than 1.3 times the perplexity of the input. We finally showed the top-8 revisions re-ranked to the users. The language model used for re-ranking is the same as the model used for the completion feature (Appendix B).

B Details on completion model

Data. We collected 234,830 PDFs of the papers published in ACL Anthology by 2019. We used GROBID[22] for extracting the text information from the PDF files. The training data is formatted as shown in Table 5. The title name is omitted with 20% probability. The order of the sections in the same paper was shuffled.

Model. We used a pre-trained GPT-2 small (117M). Table 6 shows the hyperparameters for fine-tuning the pre-trained GPT-2. We used an implementation in Transformers (Wolf et al., 2019). We used nucleus sampling (Holtzman et al., 2020) with $p = 0.97$ to generate the texts.

C Statistics of the drafts

Table 7 shows the statistics of the drafts collected in Section 4. The column "word type" shows the number of types of the tokens used in the drafts.

---

[22] https://github.com/kermitt2/grobid

Fairseq model	architecture	lightconv_iwslt_de_en
Optimizer	algorithm	Adam
	learning rate	5e-4
	adam epsilon	1e-08
	adam betas	(0.9, 0.98)
	weight decay	0.0001
	clip norm	0.0
Learning rate scheduler	type	inverse_sqrt
	warmup updates	4000
	warmup init lrarning rate	1e-7
	min learning rate	1e-9
Training	batch size	24,000 tokens
	updates	1,050,530 steps

Table 4: Hyperparameters of the revision feature.

```
@ Title @

* Section name
Texts in the section
· · ·

* Section name
Texts in the section
⟨|endoftext|⟩

@ Title (of another paper) @
· · ·
```

Table 5: The format of the training data for the completion model.

D   Details on the evaluation in Section 4.1

We used BLEURT-Base with 128 max tokens.[23] BLEURT is designed to evaluate the similarity of a given sentence pair. Thus, we first split each draft into sentences, and each sentence in first drafts is aligned with the most similar sentence in the corresponding final draft. Sentence splitting and sentence alignment is achieved by spaCy.[24] Note that the references has been created so that the sentence separation does not change from the original first draft. Finally, we calculate each sentence pair with BLEURT, and averaged the results.

---

[23]https://storage.googleapis.com/bleurt-oss/bleurt-base-128.zip
[24]Sentence similarity is computed using cosine similarity of average word vectors. We used spaCy's en_core_web_lg model.

Model	architecture	gpt2
Optimizer	algorithm	Adam
	learning rate	5e-5
	adam epsilon	1e-8
	adam betas	(0.9, 0.999)
	weight decay	0.0
	clip norm	1.0
Learning rate scheduler	type	linear
	warmup updates	0
	max learning rate	5e-5
	total epochs (just used for scheduling)	100
Training	batch size	262,144 tokens
	updates	138,300 steps

Table 6: Hyperparameters for fine-tuning LMs.

drafts	length	word types
Final drafts (reference)	199 ± 52	108 ± 17
HUMAN&MACHINE	192 ± 40	101 ± 17
HUMAN-ONLY	192 ± 43	100 ± 16
MACHINE-ONLY	199 ± 58	105 ± 22
First drafts	202 ± 56	104 ± 22

Table 7: Statistics of the drafts. The scores are averaged over the drafts. The values following "$\pm$" denote the standard deviation of the scores.