Rewriting Math Word Problems with Large Language Models

Kole Norberg[1], Husni Almoubayyed[1], Stephen E. Fancsali[1], Logan De Ley[1], Kyle Weldon[1], April Murphy[1] and Steve Ritter[1]

[1]*Carnegie Learning, Inc., 501 Grant St, Pittsburgh, PA 15219*

Abstract

Large Language Models have recently achieved high performance on many writing tasks. In a recent study, math word problems in Carnegie Learning's MATHia adaptive learning software were rewritten by human authors to improve their clarity and specificity. The randomized experiment found that emerging readers who received the rewritten word problems spent less time completing the problems and also achieved higher mastery compared to emerging readers who received the original content. We used GPT-4 to rewrite the same set of math word problems, prompting it to follow the same guidelines that the human authors followed. We lay out our prompt engineering process, comparing several prompting strategies: zero-shot, few-shot, and chain-of-thought prompting. Additionally, we overview how we leveraged GPT's ability to write python code in order to encode mathematical components of word problems. We report text analysis of the original, human-rewritten, and GPT-rewritten problems. GPT rewrites had the most optimal readability, lexical diversity, and cohesion scores but used more low frequency words. We present our plan to test the GPT outputs in upcoming randomized field trials in MATHia.

Keywords

Large language models, artificial intelligence, intelligent tutoring system, personalized learning

1. Introduction

Recent studies have found that building more comprehensive learner models results in better student learning outcomes and experience. In particular, research has pointed towards strong connections between mathematics and reading comprehension (e.g., [1]). Almoubayyed et al. [2] built a highly-accurate machine learning model to predict students' reading ability based on their interactions with an introductory activity in a mathematics adaptive learning software, MATHia.[1] This model enables researchers to target emerging readers as a sub-population when building out reading supports in MATHia (600,000+ user base) without the need for intrusive reading tests or collecting state exam scores.

Supporting the connection between reading skill and math outcomes, a recent randomized field trial with 12,000+ students demonstrated that improving the readability of math word problems in Carnegie Learning's MATHia improved student outcomes. This was particularly true among emerging readers who were identified using the the predictive model developed in

AIEd23: Artificial Intelligence in Education, Empowering Education with LLMs Workshop, July 07, 2023, Tokyo, Japan
knorberg1@carnegielearning.com (K. Norberg); halmoubayyed@carnegielearning.com (H. Almoubayyed)

[1]The predictive model generalized well to different states and demographics in [3].

[2]. Specifically, predicted emerging readers who received the rewritten word problems mastered more skills, made fewer errors, and required fewer problem opportunities to reach mastery, saving them a significant amount of time (over a third) compared to emerging readers who received the original problems. Despite this success, it is unclear whether manually rewriting all word problems in MATHia is a scalable solution or indeed if the results will replicate across hundreds of lessons in Carnegie Learning's content portfolio.

Recent developments in Large Language Models (LLMs) have allowed many language tasks to scale up in less time and with lower cost. Given the major improvements shown in [4], we looked towards LLMs as a potential method to allow us to efficiently scale up the rewriting effort. This work introduces our process of prompt engineering ChatGPT [5] to rewrite word problems according to the same style guidelines that were used by human authors in [4]. We also include a comprehensive text analysis, using the Automatic Readability Tool for English (ARTE, [6]), of the outputs of GPT and compare it to that of the original problems and the human-rewrites from [4]. Finally, we lay out our plan to carry out a randomized field trial to test student outcomes from the GPT-4 rewrites according to a set of defined metrics.

2. MATHia & UpGrade

MATHia (formerly Cognitive Tutor, [7]) is an intelligent tutoring system (ITS) developed by Carnegie Learning and currently used by 600,000+ students across the United States. MATHia is typically used as part of a blended curriculum in the classroom. Content in MATHia is delivered in lessons known as 'workspaces,' with workspaces being classified as either 'Concept Builders' or 'Mastery' workspaces. Students work through a set of pre-defined steps in Concept Builders to learn new concepts in mathematics by interacting with adaptive multimedia tools. In Mastery workspaces, problems are adaptively selected, according to a Bayesian Knowledge Tracing (BKT; [8]) implementation, providing practice opportunities for students to master a set of skills. Students are either 'graduated' or 'promoted' from Mastery workspaces. A student is graduated when they successfully master all skills in a workspace. A student is promoted when they fail to master at least one skill in a workspace after completing a pre-defined maximum number of problems. In this work, we focus on two Mastery workspaces that were also the focus of [4]. Both workspaces involve analyzing models of two-step linear relationships, but one deals with integer numbers, referred to as 'Integers' and one deals with rational numbers, referred to as 'Rationals.' These two workspaces were originally chosen due to the correlation between students' performance in them and students' end-of-year English Language Arts (ELA) state test scores being high, particularly compared to the correlation between students performance in them and their end-of-year math state test scores, for a sample of students, as described in [2].

UpGrade [9] has been recently developed as a free and open-source platform for running large-scale, randomized field tests. Upgrade integrates with ITSs, such as MATHia, to enable random assignment to experimental conditions. In [4], UpGrade was used to randomly assign 12,000+ students to the control or human-rewrite conditions over a period of around 6 weeks.

3. Prompt Engineering

Our prompts were written to enable GPT-4[2] to take any math word problem and revise it according to a style guide developed by literacy experts at Carnegie Learning. In this section, we provide an overview of our steps to engineering a successful prompt. We also highlight areas of the revision process that were particularly challenging for GPT to generate.

We used a mix of zero-shot [10, 11], few-shot [12], and chain-of-thought learning [13, 14]. Table 1 illustrates revisions produced by GPT-4, and for comparison GPT-3, for each of these prompting styles. We also provide original and final versions as they appear in MATHia in at https://osf.io/xwz3h/.

3.1. Zero-shot learning

Zero-shot learning prompts provide an LLM with a set of instructions to follow but critically do not include exemplars. Such prompts produced revisions that were better than the originals but which did not fully adhere to the style guide. With our texts, zero-shot learning worked well for broad structural changes such as adding a topic sentence and contextualizing negative numbers. However, it was unable to reliably address issues related to simplifying vocabulary and sentence structure (e.g., removing prepositional clauses and passive voice). Although words like *subterranean* were successfully replaced by GPT-4, words like *eclair* were not. As will remain true in our overview of few-shot prompting, it was difficult for GPT-4 to make revisions that changed the meaning of the word. Therefore, in zero-shot learning, if a simpler, exact synonym could not be found, the word would not be replaced.

3.2. Few-shot learning

In order to improve GPT's precision and reliability, we added examples of changes that might be made for each step. We used a mix of 1-shot (one example) and 2-shot (two examples) learning based on the complexity of the specific style guideline and observed performance. Further, we asked GPT to first explain rules before carrying them out. For example, we asked it to define passive voice and then label sentences accordingly. Critically, we asked GPT to perform its revisions iteratively, so it would revise for passive voice in one step and then word choice in another. This successfully improved GPT's reliability in regards to using active voice. However, GPT continued to struggle with vocabulary changes. It was now able to identify difficult vocabulary, but if a simpler synonym was not available, it resorted to a broader more general term. This sometimes made the vocabulary of the problem confusing (e.g., replacing *eclair* with *pastry* when *donuts*, another item in the problem, are also a type of *pastry*). If pushed to be specific, it would describe the word, replacing *eclair* with *long pastry* to differentiate it from *donut*. Further, under few-shot learning, it began introducing prepositional clauses into sentences rather than removing them (e.g., *For the high school trip, ...* and *At a bake sale, ...*

[2]All revisions were generated by GPT-4 using chat.openai.com as we did not have access to the GPT-4 API at the time of production. We discuss how access to the API has modified our approach in the Conclusions and Future Work section.

3.3. Chain-of-thought learning

In our final iteration, we used chain-of-thought learning. Prompts of this type provide reasoning with their examples. The beginning of our chain-of-thought prompts contained a mock word problem, its revision, and rationale for how the revision obeyed the guidelines.

This final iteration of our prompt [3] also provided more scaffolding for identifying and replacing words. Because GPT-4 struggled with changing the meaning of the original word within a text, we asked it to just remove the word, replace it with a blank, and then replace the blank with the most likely word. Following this approach, it was able to break away from using synonyms and select words which suited the context and the age of the user (e.g., it replaced *eclair* with *cupcake*). This version of the prompt was successful on the first try at revising 26 out of 30 problems. It was successful for 3 out of 4 of the remaining problems on its second try. Its struggle with the final problem was rooted in ambiguous pronoun use in the original text. We revised the original by replacing the ambiguous pronoun with its referent, and the next GPT-4 revision was successful.

Table 1
Example revisions

Learning Type	Original	GPT-3.5	GPT-4
Zero Shot	You are in a harness preparing to explore a subterranean cave. As you begin, you are 1 meter above ground level. As you descend, your elevation changes at a rate of -16 meters every minute. The situation can be modeled by the equation y = 1 - 16x	Equipped with a harness, you stand prepared to venture into a subterranean cave. Initially, your position is 1 meter above ground level. As you make your way down, your elevation changes at a rate of -16 meters every minute. This can be represented mathematically by the equation y = 1 - 16x, where y indicates your current elevation and x represents the number of minutes elapsed.	You're ready for an underground cave adventure, standing 1 meter above the ground in your harness. Each minute, you descend by 16 meters. This scenario can be expressed as y = 1 - 16x.
Few Shot	same as above	You are exploring a subterranean cave, starting 1 meter above ground level. Your elevation, represented by y, changes with time, x, at a rate of -16 meters every minute. As you descend, your elevation decreases at a rate of 16 meters every minute according to the equation y = 1 - 16x.	You are ready to explore an underground cave, strapped in a harness and standing 1 meter above the ground. Each minute you go down, your height decreases by 16 meters. We can use the equation y = 1 - 16x to show this change in height over time.
Chain of Thought	same as above	You are in a harness preparing to explore a subterranean cave. As you begin, you are 1 meter above ground level. As you descend, your elevation changes at a rate of -16 meters every minute. The situation can be modeled by the equation y = 1 - 16x	You are getting ready to explore an underground cave wearing a harness. As you begin, you are 1 meter above ground level. Your elevation changes at a rate of 16 meters downward every minute. The equation y = 1 - 16x models the descent.

Table 2 overviews a few other strategies we tried and how well they worked (or didn't).

4. Encoding Word Problem Components

In order for a problem to work within MATHia, the mathematical components need to be encoded. After rewriting the problems, we also prompted GPT to complete the encodings. The word problems that were revised are from two workspaces which have a similar structure: they provide a student with a scenario that involves an equation of the form $y = mx + c$, and then

[3]Prompts, including the final iteration referred to here, are available at https://osf.io/xwz3h/.

Table 2
Other strategies

Strategies	Description
Producing multiple versions	At times, we asked GPT to provide multiple rewrites and select the best version. Sometimes the additional revisions were better. More often, the additional revisions were worse.
Referencing readability tools	One way to determine the readability of a text is to evaluate the frequency of its vocabulary against a corpus. Despite GPT's claim to the contrary, we saw no evidence that GPT was able to reference or use a corpus, nor was it able to evaluate the grade level of the text.
GPT engineers the prompt	When a prompt failed, we prompted GPT to explain how it understood the prompt so we could make modifications. Most of the time, this did not yield any insights. Nevertheless, there were times when GPT was able to rewrite the rule in a way that it would reliably follow it during future revisions.
Negative prompting	It was often necessary to tell GPT *not* to do something, most notably, not to use pronouns. GPT-4 does better in general when given directions on what to do rather than what not to do (e.g., "use specific language" rather than "do not use general language"). However, we found it difficult to avoid words like "not, without, etc." Ultimately we found that when a negative prompt was required, specifics worked better than generalities. Telling GPT-4 to avoid pronouns did not work. Telling it to avoid "he, she, they ..." worked better.
Evaluating and repeating steps	At times, we included instructions in the prompt to trigger GPT to evaluate output and repeat steps if errors were present. GPT never repeated any of the steps, even when we would have liked it to do so.
AI glitches	One GPT glitch is that it would often stop writing the middle of producing the output. This was solved every time by prompting the AI to "continue" but increased the number of prompts per problem. Other, rarer, glitches included GPT starting on a step other than Step 1a or declaring all steps unnecessary and skipping to the final output. This was more common with GPT-3 than GPT-4 but did occasionally require opening a new chat GPT-4 session to fix.

ask the student to match each of the elements m, c, x, mx, y with what they represent in that particular scenario. LLMs struggle to translate language into mathematics. However, they excel with programming tasks. We overcame GPT's limitations with mathematics by asking it to write a python function that takes in the components of m, c, x and output y in one prompt, and then use the function to identify the five components in a second prompt. Figure 1 shows an example of this prompting and the GPT outcomes.

GPT was successful in all but one case. Upon further examination of that case, we found that the original word problem text was written in a way that was overly vague. This highlighted a use-case for GPT that we had not originally considered. GPT may be used as a quality assurance tool: if GPT fails at identifying the components, it may indicate that the problem is ambiguous and should be rewritten.

For this particular task, zero-shot learning was all that was required. In this study, we carried out the encodings in a single chain, with zero-shot learning, and in batches of 3-5 problems at a time. Nevertheless, experimental use of few-shot learning and additional prompting helped constrain its responses. With the API, it might be more efficient to use multiple chains with few-shot learning instead.

GPT often tried to be more helpful by providing code and numerical examples, and often went on for too long trying to explain the python function. To counter this behavior, we always added text to our prompt such as, "Do not provide numbers or code examples."

5. LLM Output Analysis

Readability was assessed across multiple metrics using the Automatic Readability Tool for English (ARTE) [6]. We selected a subset of the measures provided by ARTE that we believed were representative. Results from ARTE measures are reported in Table 3. On average, the

Prompt
Write a python function that takes a slope ”m”, an intercept ”c”, an independent value ”x” and returns a dependent value ”y”. Do not provide numbers or code examples.

Completion
Here’s a Python function that takes the slope (m), intercept (c), and independent value (x) as arguments, and returns the dependent value (y) based on the linear equation y = mx + c:

```
def linear_equation(m, c, x):
    y = m * x + c
    return y
```

This function first calculates the dependent value (y) using the provided slope (m), intercept (c), and independent value (x). Then, it returns the computed value (y) as the output.

Prompt
Apply the python function to the following problem and concisely explain the quantities x, m, c, m*x, and y. Do not provide any numbers. Do not provide code examples. Provide the answer as a list. Use conversational language and active voice when possible, and repeat words from the word problem as much as possible.
Alex got a gift card for Joe’s Juice Bar and uses it to buy mango juice. Alex received a $25 gift card. On each visit, Alex buys a glass of mango juice, and the value of the card decreases by $3. The equation $y = 25 - 3x$ models how the value of the gift card changes.

Completion
Alex’s gift card for Joe’s Juice Bar:

- x: the number of glasses of mango juice Alex buys
- m: the cost of each glass of mango juice, in dollars
- c: the initial value of the gift card, in dollars
- m*x: the total amount spent on mango juice, in dollars
- y: the remaining value of the gift card after buying mango juice, in dollars

Figure 1: The prompting process and example completions for the task of encoding math word problem components in GPT-4. Asking GPT-4 to apply a python function to a word problems yielded much more consistent responses than prompting it to do the same without a python function.

math problems had 46.94 words each. This is considerably less than the recommended number of words (170-200 words) for achieving reliable results [15][16].[4] Thus, we also aggregated all of the word problems to produce one large text per condition with m = 1,425 words. Results for the longer text matched those for the shorter texts and are reported in Table 3 for completion.

[4]Zhou et al. [15] analyzed reading scores across tools and metrics. They found that the same readability metric may vary across tools due to differences in its implementation (e.g., how it counts syllables). Further, readability metrics measuring similar outcomes (e.g., coherence, grade level) also varied in their determination. These differences diminished once a text reached 200 words, and meaningful improvement beyond 900 words was not detected.

- **CAREC-M** (modified Crowdsourced Algorithm of Reading Comprehension (CAREC-M) [17]) considers features related to syntax, lexical diversity, and cohesion, modified to account for text length. CAREC-M scores were lowest for the GPT-4 rewrites, indicating that GPT-4 successfully improved cohesion and lowered lexical diversity as compared to the original and human-rewritten text.
- **SBERT** (Sentence-BERT [18, 19]) is a transformer based deep learning model which assesses the semantic similarity within a text. The higher values for the GPT-4 rewrites indicate improved readability as compared to the original texts. For the aggregated set of texts, the GPT-4 rewrites also improved over human rewrites.
- **FKGL** Flesch-Kincaid Grade Level [16] compares word counts to sentence and syllable counts to determine the appropriate grade level for a text. FKGL decreases for both sets of rewrites indicating that humans and GPT used words with fewer syllables and wrote shorter sentences compared to the original texts.
- **NDC** (New Dale-Chall [20]) is calculated based on sentence length and the percentage of words in a text that may be considered unfamiliar (i.e., are not part of a set of 5,000 pre-selected common words). This was the only metric for which the GPT-4 rewrites declined in performance relative to the original texts. During the prompt revision process, GPT-4 persistently struggled with identifying and replacing low frequency words. The results here suggest GPT-4 was unable to reach human proficiency at selecting grade-level appropriate words. In Conclusions and Future Work, we discuss how we intend to improve on this going forward.

Table 3
Text difficulty metrics

Scenario	CAREC-M		SBERT*		NDC		FKGL	
	m	*se*	*m*	*se*	*m*	*se*	*m*	*se*
GPT	0.19	0.01	-0.13	0.08	8.70	0.24	5.47	0.33
Human	0.24	0.01	-0.13	0.07	8.41	0.21	5.51	0.28
Original	0.25	0.01	-0.21	0.06	8.61	0.17	5.85	0.27
GPT Aggregated	0.11		-0.56		8.66		5.32	
Human Aggregated	0.19		-0.94		8.37		5.39	
Original Aggregated	0.23		-0.93		8.59		5.63	

Note: Means and std. errors for readability scores. *Lower values indicate greater readability except for SBERT where higher (here less negative) scores are better.

6. Evaluation Plan and Metrics

While we have carried out a text analysis of the problems in their three varieties: original, human-rewritten, and GPT-rewritten, a more comprehensive method of comparing them is to measure student outcomes. We plan to run a randomized field trial of the human-rewritten and GPT-rewritten problems. We plan to evaluate the variants following the metrics laid out in [4], namely, median time to completing the workspace, average number of problems completed, average number of hints used, and promotion rate (percentage of students that failed to reach mastery on all skills in a workspace). Furthermore, we will use the predictive model from [2] to evaluate the variants separately on students predicted to be emerging readers as well as the overall student population. We hope to achieve similar success with GPT rewrites as was achieved by the human rewrites, allowing us to use LLMs to scale up this work significantly to the dozens of workspaces that use word problems in MATHia, likely at a substantially lower cost.

7. Conclusions and Future Work

The use of LLMs have allowed us to scale-up and replicate writing tasks that, when previously done by humans, were found to improve student outcomes. Using chain-of-thought learning, we were able to use GPT-4 to rewrite math word problems in MATHia following a set of guidelines intended to support emerging readers. Further, we circumvented GPT-4's struggle with formulating and solving mathematical problems by asking it to produce python code which would encode those problems.

Text analysis on GPT-4's output showed that GPT-4's rewrites improved across several metrics: readability, lexical diversity, and cohesion; but scored lower on a metric evaluating use of familiar words. The rewrites produced by GPT-4 are now being presented to MATHia users as part of a randomized controlled experiment to test their effectiveness. If the GPT-4 rewrites produce similar advances in math outcomes as the human-rewrites did, it will represent a path forward to quickly improving learning content throughout MATHia.

Although our first attempt using chat.openai.com was successful, the final prompt was nearly three pages and used 2,143 tokens. Further, GPT-4 would occasionally stop in the middle of production and need to be prompted to continue. On average, we needed to use 3 of our allocated 25 prompts per three hours for each problem. Continuing forward with chat.openai.com presents a serious barrier to scalability. To scale this work, we are instead using the GPT-4 API. Using the API, we can reduce the number of tokens per prompt by using natural language processing in python to evaluate the text (e.g., to label passive sentences, pronouns, and rare words). By offloading tagging and evaluation steps (steps 1a-b & d-f, 3a-b, step 8, 9, and 10a-b & d-e of the final prompt), we have had some success in producing similar revisions to the ones reported here with as few as 427 tokens.

LLMs are proving to be useful tools for writing large amounts of content to specification. GPT-4, in particular, can quickly detect and mimic style and structure and implement broad structural changes. The focus of this work was on improving MATHia word problems to enable emerging readers to engage more with understanding math and less with parsing text. However, the steps of producing a prompt to evaluate and revise text as well as identify important components of math problems are transferable. We found that GPT is a scalable resource for implementing fast changes which would previously have required hundreds of hours of labor.

We recommend strategies which consider the nature of the LLM. Although we discuss GPT-4 as ”evaluating text,” in reality, it is a prediction engine that uses prior text to inform its output. Asking it to define and label text helps to guide the prediction process in a way that ”evaluation” does not. Similarly, to see GPT-4 generate a greater range of word substitutions, it can help to provide less constraint in guiding the substitution. In our revisions, GPT-4 appeared to use the meaning of the words in the original text as a constraint. Asking it to remove the words freed it from this constraint. Finally, GPT-4 has been extensively trained on python code. We found it could use that code as a structure for its response when engaging in tasks that require mathematical reasoning. Keeping these techniques and caveats in mind, we were able to greatly reduce the time it takes to improve the readability of select MATHia word problems.

If we find that these GPT-4 rewrites result in better student outcomes, such as reduced time and increased accuracy for emerging readers, this would allow us to scale-up rewriting content in many other workspaces. Furthermore, it would be interesting to study how successful GPT-4 or other LLMs are in writing original content that replaces older content while using the same knowledge component models in MATHia or similar ITSs. A longer-term goal would be more real-time content generation, where problems can be personalized to students' interests, without having a pre-defined bank of problems for each topic. We believe that this work is a first step towards such personalization, but there are several issues that need to be addressed with LLMs to achieve such real-time problem personalization. Namely, we seek to insure that LLM outputs are always consistent, automatically integratable with the ITS, mathematically correct, and helpful for student learning.

Acknowledgments

The research reported here was supported in part by the Institute of Education Sciences, U.S. Department of Education, through grant R324A210289 to Center for Applied Special Technology (CAST). The opinions expressed are those of the authors and do not represent the views of the Institute or the U.S. Department of Education.

References

[1] K. R. Koedinger, M. J. Nathan, The real story behind story problems: Effects of representations on quantitative reasoning, Journal of the Learning Sciences 13 (2004) 129–164. doi:10.1207/s15327809jls1302_1.

[2] H. Almoubayyed, S. E. Fancsali, S. Ritter, Instruction-embedded assessment for reading ability in adaptive mathematics software, in: Proceedings of the 13th International Conference on Learning Analytics and Knowledge, LAK '23, Association for Computing Machinery, New York, NY, USA, 2023.

[3] H. Almoubayyed, S. E. Fancsali, S. Ritter, Generalizing predictive models of reading ability in adaptive mathematics software, in: International Conference on Educational Data Mining 2023, EDM2023, 2023.

[4] H. Almoubayyed, R. Bastoni, S. Berman, S. Galasso, M. Jensen, L. Lester, A. Murphy, M. Swartz, K. Weldon, S. E. Fancsali, J. Gropen, S. Ritter, Rewriting math word problems to improve learning outcomes for emerging readers: A randomized field trial in carnegie learning's mathia, in: The 24th International Conference on Artificial Intelligence in Education (AIED 2023), AIEd '23, Springer Nature, 2023.

[5] OpenAI, ChatGPT (Mar 14 version), https://chat.openai.com/chat, 2023. [Large language model].
[6] J. Choi, S. A. Crossley, Advances in readability research: A new readability web app for english, 2022 International Conference on Advanced Learning Technologies (ICALT) (2022) 1–5.
[7] S. Ritter, J. R. Anderson, K. Koedinger, A. T. Corbett, Cognitive tutor: Applied research in mathematics education, Psychonomic Bulletin & Review 14 (2007) 249–255.
[8] J. R. Anderson, A. T. Corbett, Knowledge tracing: Modeling the acquisition of procedural knowledge, User Modeling and User-Adapted Interaction 4 (1994) 253–278.
[9] S. Ritter, A. Murphy, S. E. Fancsali, V. Fitkariwala, J. D. Lomas, Upgrade: An open source tool to support a/b testing in educational software, in: Proceedings of the First Workshop on Educational A/B Testing at Scale, EdTech Books, 2020.
[10] M. Palatucci, D. Pomerleau, G. E. Hinton, T. M. Mitchell, Zero-shot learning with semantic output codes, Advances in neural information processing systems 22 (2009).
[11] Y. Xian, B. Schiele, Z. Akata, Zero-shot learning-the good, the bad and the ugly, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 4582–4591.
[12] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.
[13] E. Saravia, Prompt Engineering Guide, https://github.com/dair-ai/Prompt-Engineering-Guide (2022).
[14] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. Chi, Q. Le, D. Zhou, Chain of thought prompting elicits reasoning in large language models, arXiv preprint arXiv:2201.11903 (2022).
[15] S. Zhou, H. Jeong, P. A. Green, How consistent are the best-known readability equations in estimating the readability of design standards?, IEEE Transactions on Professional Communication 60 (2017) 97–111.
[16] J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, B. S. Chissom, Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel, Technical Report, Naval Technical Training Command Millington TN Research Branch, 1975.
[17] S. A. Crossley, K. Kyle, M. Dascalu, The tool for the automatic analysis of cohesion 2.0: Integrating semantic similarity and text overlap, Behavior research methods 51 (2019) 14–27.
[18] S. Crossley, J. S. Choi, Y. Scherber, M. Lucka, Using large language models to develop readability formulas for educational settings, in: Proceedings of the AIED, 2023.
[19] N. Reimers, I. Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, arXiv preprint arXiv:1908.10084 (2019).
[20] J. S. Chall, E. Dale, Readability revisited: The new Dale-Chall readability formula, Brookline Books, 1995.