QReT: Quality Aware Token Count Reduction

Anonymous ACL submission

Abstract

LLMs are widely used nowadays by several enterprises for various use cases. This is due to their general applicability and demonstrated success across multiple domains and tasks. However, there is a monetary cost associated with the use of commercially available inference APIs to LLMs. This cost generally depends on the number of input and output tokens and the cost parameters of the provider. In this work, we propose a framework QReT for reducing the input token count in prompts in a controllable quality aware manner. QReT first paraphrases the prompt to reduce token counts while maintaining quality measures. Secondly, it applies certain heuristics, again a controlled manner to reduce the final token count, without affecting the understanding by LLMs (hence, the output quality). We empirically validate QReT across several datasets and tasks and show its effectiveness.

1 Introduction

The last few years have witnessed remarkable and rapid growth in using Large Language Models (LLMs) across various domains. ChatGPT is estimated to cost over $700,000 per day to operate [17], and using GPT-4 to support customer service can cost a small business over $21,000 a month [20]. Since these costs primarily depend on the number of input and output tokens and the corresponding API calls, reducing the number of tokens in a smart way without compromising quality can help in significantly reducing the costs. Moreover, LLMs offer a restricted token context window that often makes fitting the entire query into a single prompt infeasible. Therefore, reducing the tokens in a smart way can be further helpful in fitting the contexts.

Reducing tokens however can cause a loss in information and meaning, resulting in depreciated response quality. Moreover, token count relies on the LLM tokeniser, and thus, any token reduction scheme must incorporate the respective tokenisers to be consistently usable across LLMs. Finally, token count reduction is generally less explored area in literature; there is limited work and no dataset dedicated to it. Hence, the quality aware reduction in token count is a challenging problem.

Typical use case scenarios include summarization and question answering over documents, where LLMs like GPT must be queried with large text prompts. Question answering involves a retrieval stage; the text output is fed to an LLM along with the query. Similarly, for generating a document's section-wise summaries, we will need to query the LLM with the entire content in each section. In both cases, we can reduce tokens to save costs before feeding the text as input to an LLM.

We propose a quality aware token count reduction system QReT that consists of two main components: (i)Token Optimized Text Simplification, (ii) Token Optimization Heuristics. Our main contributions are as follows:

- Token Optimized Text Simplification: Paraphrasing sentences in a controllable manner to reduce token count and meet quality requirements.
- Token optimization heuristics: Tokeniser aware heuristics, that are chosen judiciously for the given context and applied in the optimal order.
- We release the annotated datasets for further research by the community[1].

2 Related Work

There exists some prior work in sentence or passage level paraphrasing [15, 16] while attempting to preserve the information and meaning, however, these

[1] https://anonymous.4open.science/r/llm-cogs-5FCF/sentence_simplification/README.md

are not directly applicable for token count reduction. There is also some work in compressing sentences using Reinforcement Learning (SCRL) [6] that extracts a sequence of tokens from a given sentence. However, this method simply drops words, and can result in incoherent sentences, leading to loss in meaning. Moreover this method lacks flexibility, and neither does it leverage reordering, semantic or lexical changes nor can it change the target length at run time (as it requires retraining the model). A related work, GPTrim is an open-source python library which uses heuristics like removal of punctation, stop words and stemming to reduce token count, however, the quality often suffers. Hence, quality aware reduction in token count becomes challenging. Most of these approaches work in a tokenizer agnostic manner, hence leave room for inefficiencies in token count reduction.

Recently, LLMLingua [9] and its follow up works from Microsoft (`https://github.com/microsoft/LLMLingua`) have proposed token reduction using smaller LLMs, where compression is done based on perplexity of generated tokens. While this can given good token reduction, the resultant output becomes unreadable by humans and the quality drop seems unpredictable (as observed by our experiments on enterprise Question Answering tasks). It is therefore risky to use such a method for customer facing products.

3 Token Optimized Text Simplification

We propose simplifying the sentences in input prompts in a token aware manner, while preserving semantics to maintain the quality of outputs. We took inspiration from the work of Martin et al. [15] who build a sequence-to-sequence model for generating audience centric simplifications for easier readability. They adapt a discrete parameterization mechanism that provides explicit control on simplification via various parameters like number of characters, Levenshtein similarity [11], word frequency ratio and dependency tree depth [16]. To control various parameters while simplification at inference time, the parallel training data is labelled with tags corresponding to the desired controllable parameters. We build upon this work and leverage the above technique to control the token count and information loss in the paraphrased sentences.

We train our model on the WikiLarge [23] dataset. The dataset contains 296,402/2,000/359 samples (train/val/test) of automatically aligned complex-simple sentence pairs from English Wikipedia and Simple English Wikipedia. We label the complex-simple sentence pairs with two parameters, NUM_TOKENS_RATIO and BERT_SCORE. The former corresponds to the ratio of the number of tokens (using OpenAI's cl100k-base [4] tokenizer) in the simple and the complex sentence, and the latter is the BERTScore [22] between the two sentences.

The model is provided with oracle information on the target sequence in the form of control tokens appended to the source sequence. For example, if the desired token count in the target sequence is 70% of the token count in the source sequence while the desired BERTScore should be 0.95 with the original sentence,, we append [BERTSCORE_0.95 NUM_TOKENS_RATIO_0.70] tag to the source sentence.

Training Details: Our backbone architecture is BART-large [12], a transformer encoder-decoder (seq2seq). We use the fairseq [18] implementation for BART-large from [15], keeping the optimization procedure and hyper-parameters the same as the original implementation. The model was trained on 4 Nvidia a10g GPUs for approximately 10 hours. Figure 1 shows an example output of the model. Further examples are listed in Table 1 for qualitative evaluation by the reader.

<BERTSCORE_0.95> <NUMTOKENSRATIO_0.8> Archaeological evidence suggests a history of settlement in the area since roughly 2000 BC.

Archaeological evidence shows settlement in the area since 2000 BC.

Figure 1: Token optimized text simplification example.

4 Token Optimization Heuristics

Here we describe some general heuristic rules that we observed can be applied for reducing token count while maintaining quality. We discuss the rules, as well as their effects and the associated optimization problem of applying them. We chose the OpenAI tokenizer tiktoken [4] for implementation, experimentation, and testing. We manually inspected the tokenized version of samples of texts taken from question-answering datasets like ReCLor [21], LogiQA [14], and MS-Marco [5] and analyzed the tokenizer inefficiencies. Based on these observations, we devise generalizable rules to edit the words or phrases to reduce token count

Original Sentence	Simplified Sentence @ 0.8	Simplified Sentence @ 0.6
Effective altruism advocates using evidence to determine the most effective ways to benefit others.	Effective altruism uses evidence to find the best way to help others.	Effective altruism is about using evidence to help others.
The joyful choir's harmonious melody resonated through the cathedral, captivating the congregation.	The joyful melody could be heard all through the cathedral.	The joyful melody could be heard all through the cathedral.
Jeddah is the principal gateway to Mecca, Islam's holiest city, which able-bodied Muslims are required to visit at least once in their lifetime.	Jeddah is the main gateway to Mecca, Islam's holiest city. Muslims must visit Mecca at least once in their lives.	Jeddah is the main city on the road to Mecca, Islam's holiest city.

Table 1: Qualitative Examples

while retaining the maximum information of the original text. In total, we devise eight heuristics, the details of which can be found in Table 2.

4.1 Optimized application of Heuristics

Let us say we have a passage P where the sentences of the passage are $\{s_1, s_2, \ldots, s_n\}$. Further, we have $\{H_1, H2, \ldots, H_m\}$ as our token trimming heuristics. Define $x_{i,j}$ as the indicator variable if heuristic H_j is selected to be applied on sentence s_i. Define $c_{i,j}$ as the cost i.e., the estimated performance degradation and let $p_{i,j}$ be the profit i.e., number of tokens saved upon applying H_j to s_i. Let us say we can tolerate a maximum performance loss of C , then the choice of heuristics for a given s_i reduces to the knapsack problem, where the capacity is C, cost is $c_{i,j}$ and profit is $p_{i,j}$ for heuristic (item) H_j. Once we solve the knapsack problem approximately, we will have for each sentence which heuristics to apply. Since the number of heuristics is ≤ 8, we brute force through the search space to determine the optimal order of application of these heuristics on each sentence.

5 Experiments on Token Optimization

We first describe the experiments on open source and generic datasets and wide variety of use cases such as Question Answering, Summarization and NLI tasks.

5.1 Datasets

We use Question Answering and NLI datasets to evaluate and benchmark our token optimization methods. We use multiple-choice question-answering over long-form question answering datasets for a variety of reasons. Firstly, since we use a powerful LLM like GPT 3.5 Turbo to evaluate, we need challenging datasets that involve logical reasoning to arrive at the correct response. To the best of our knowledge, there are no appropriate logical long-form QA datasets; however, several challenging MCQ and NLI datasets suit our purpose. Secondly, metrics for evaluating long-form question-answering tasks are not reliable, given the subjective nature of the task. We have used BERTScore to evaluate summarization on an enterprise summarization dataset Dataset I, provided by Adobe Inc. However, BERTScore has its limitations as an evaluation metric for question-answering. Finally, since LLMs are proficient at generating coherent and contextually appropriate responses, the model compensates for the compressed text or dropped words, and the variance in results of long-form QA is minimal across various compression methods. Thus, we cannot capture the actual loss in information and meaning owing to LLM capabilities when evaluating long-form QA datasets. We give details of the datasets used for our Token optimization experiments in Table 3.

5.2 Results on Token Optimization

We experimented on 3 datasets: namely CosmosQA(QnA), Control(NLI), Dataset I (Summary). We compared QReT against GPTrim and SCRL as baselines (refer Section 2). Recall that in QReT, the original context is converted into a simplified version by the simplification module. On top of this simplified context, various heuristics are ap-

Heuristic (Abbv.)	Description
Adjust Spaces and Capitalizations (CS)	Prepending space and changing the case of the first letter of some words reduce the token count.
Replace Synonyms (RS)	We use the thesaurus [3] synonym dictionary to replace high token count words with their less token count counterparts.
Lemmatization and Stemming (LS)	We implement the lemmatization of words by first stemming using the NLTK stemmer [1] and then using spell correction with [2]. This is done only in cases where there is a reduction in the tokens.
Bracket Removal (RB)	Removing round parenthesis is found to save tokens.
Handle Compound Words (HC)	We create a dictionary of prefixes and split compound words by adding a space after the prefix in the cases where there is a token count reduction.
Stop Word Removal (RSW)	Removal of selective stop words is found to save tokens.
Punctuation Removal (RP)	Removal of selective punctuation marks saves tokens. This needs to be done carefully so as not to affect Math expressions or Time expressions
Handle Acronyms (RA)	We remove the dots between the letters of an acronym to reduce the token count where applicable.

Table 2: Token Reduction Heuristics

Dataset	Task	Description
CosmosQA [8]	Question Answering	CosmosQA is a large-scale dataset of 35.6K problems that require commonsense-based reading comprehension, formulated as multiple-choice questions. It focuses on reading between the lines over a diverse collection of people's everyday narratives, asking questions concerning the likely causes or effects of events that require reasoning beyond the exact text spans in the context.
LogiQA [14]	Question Answering	LogiQA is sourced from expert-written questions for testing human Logical reasoning. It consists of 8,678 QA instances, covering multiple types of deductive reasoning
ReCLoR [21]	Question Answering	ReClor is a dataset extracted from logical reasoning questions of standardized graduate admission examinations. Empirical results show that the state-of-the-art models struggle on ReClor with poor performance.
ConTRoL [13]	Question Answering	ConTRoL is a dataset for ConTextual Reasoning over Long texts. Consisting of 8,325 expert-designed "context-hypothesis" pairs with gold labels, ConTRoL is a passage-level NLI dataset focusing on complex contextual reasoning types such as logical reasoning.
Dataset I	Summarization	Dataset I is a summarization dataset constructed from sections from 80+ PDFs from Adobe Inc. PDF corpus. The gold summaries are obtained by using GPT-4. This data set is the most reflective of our use case, i.e., real-world documents.
Dataset II	Summarization	Dataset II is a summarization dataset constructed from taking samples from public datasets namely, bigpatent[19], samsum[7], wiki bio[10]. The gold summaries are generated using GPT-3.5-Turbo, it contains candidate summaries from vicuna-13b, Text-Davinci-003 and Text-Curie-001.

Table 3: Overview of datasets used to evaluate our Token Optimization Module (QReT)

plied in a controlled manner to further reduce the token count and complexity. The modified context is then used as the input context for the concerned task.

5.2.1 Token Reduction and Quality:

We found that QReT and SCRL lead to comparable loss in performance with more compression being achieved by QReT. GPTrim, on the other hand, though providing highest compression percentage also leads to much higher loss in performance as can be seen in Figure 3 and 2.

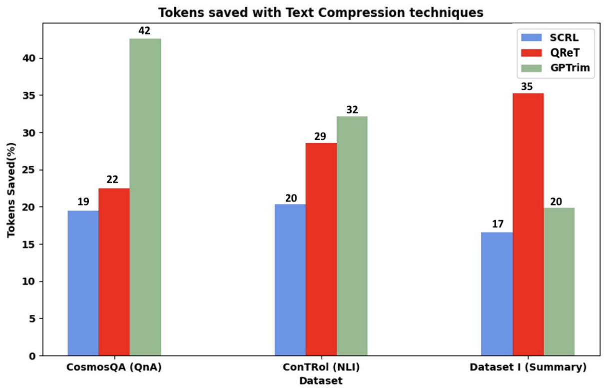

Figure 2: Compression Achieved.

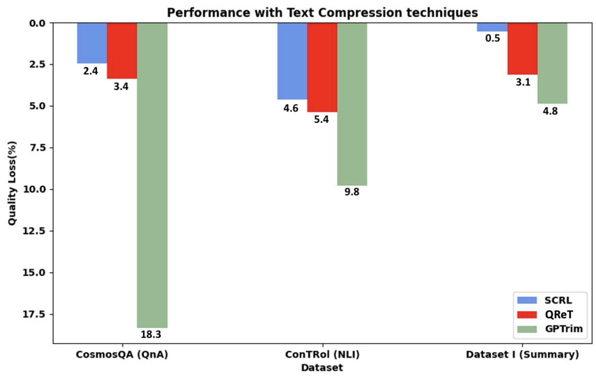

Figure 3: Performance Loss Obtained.

Here we list the results for each dataset on the token optimization experiments. Table 4 lists the results on CosmosQA, Table 5 on ConTRol and Table 6 on Dataset I.

5.2.2 Optimized Token Reduction:

We further experimented with **optimized token reduction heuristics** by controlling the quality loss parameter comparing to a brute force application of all heuristics in a fixed order. We can see that by setting the loss threshold, we are able to reduce the quality loss in a controlled manner, while achieving similar token reduction.

Compression Method	Accuracy	Compression %
None	0.736	0.0
GPTrim	0.601	42.6
SCRL	0.718	19.5
QReT	0.711	22.5

Table 4: Token Compression on CosmosQA

Compression Method	Accuracy	Compression %
None	0.521	0.0
GPTrim	0.470	32.13
SCRL	0.497	20.3
QReT	0.493	28.6

Table 5: Token Compression on ConTRoL

Table 7 shows the tradeoff of quality loss with tokens saved optimally with respect to the brute-force method of applying all heuristics in a fixed order. The $x\%$ Threshold refers to setting the loss tolerance at $x\%$ of the total loss in quality incurred by the brute force method and optimizing the tokens accordingly. In this case, since it was a sentence by sentence comparison, we measured the quality loss in terms of S-BERT similarity. That is, it is measured as $1 - SB(s_1, s_2)$, where $SB(s_1, s_2)$ refers to the S-BERT cosine similarity between the embeddings of sentences s_1 and s_2. We did this on Dataset I, and we are reporting the numbers for 2 such samples as illustrative here.

5.2.3 Token Optimization Module - Ablation Study

We compress some Question-Answering, NLI and Text Summarization datasets using our token optimization module with the above-mentioned heuristics. We evaluate and plot the contributions of each heuristic on the various datasets (Fig. 4).

Table 8 lists the token compression obtained on various datasets.

Compression Method	BertScore	Compression %
None	0.738	0.0
GPTrim	0.702	19.8
SCRL	0.734	16.6
QReT	0.715	35.2

Table 6: Token Compression on Dataset I

Method	Loss-1	Tokens Saved-1	Loss-2	Tokens Saved-2
Brute Force	0.04	7	0.025	14
90% Threshold	0.0285	5	0.022	13
80% Threshold	0.0285	5	0.0148	9
70% Threshold	0.008	2	0.0148	9

Table 7: Token Optimization Trade-off

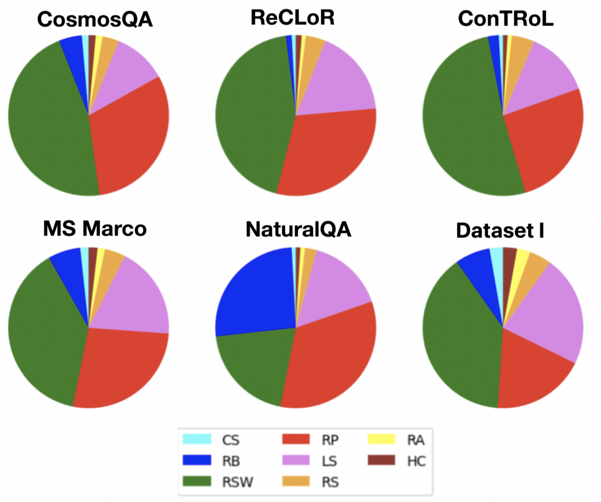

Figure 4: Ablation study of various heuristics

Dataset	Compression %
CosmosQA	18.27
ReCLoR	18.70
ConTRoL	21.44
Natural QA	20.91
MS Marco	21.44
Dataset I	22.07

Table 8: Token Compression obtained on various datasets.

5.3 Experiments on Enterprise Document processing use case

We have integrated our token optimization pipeline in an enterprise specific document processing pipeline and have evaluated on questing answering tasks. The token savings is of the order of $10 - 17\%$ across the documents (input). The latency is in milliseconds. Human evaluation studies have also been done comparing the token optimized outputs with the results generated with out token optimization, henceforth referred to as Neptune. In most metrics, token optimization seems to be doing better including overall satisfaction which is higher with token optimization (see Figure 5).

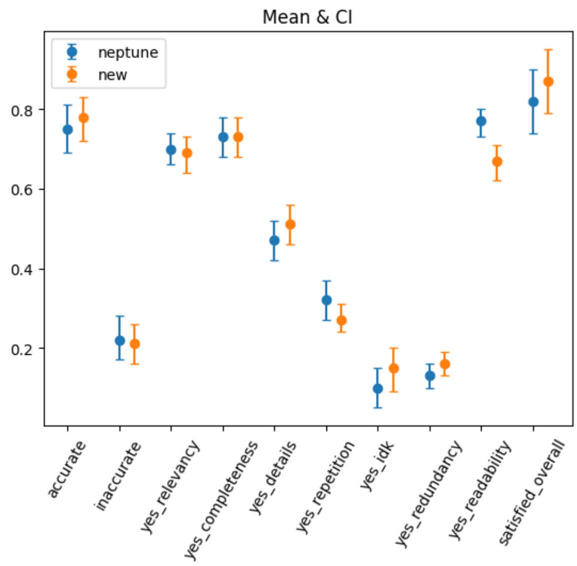

Figure 5: Box plot of various metrics comparing Neptune (referring to the study done without token optimization) with new (referring to same study done with token optimization).

The readability scores seemed slightly lower, hence a follow up evaluation focused on comparing readability was conducted. Focusing only on the subset of 34 questions where scores were different, we wanted to understand whether there were any differences between the neptune and optimization answers. 30 users were asked to review both system answers simultaneously and asked to select which one they preferred (in terms of readability). Additionally users reported any readability issues for each answer. Presentation order was counterbalanced and system name was hidden from users.

There was no clear indication of preference for one method over the other (Table 9). Users rated the answers as equivalent or picked one over the other uniformly. Table 10 shows the percentage of readability issues identified. For Neptune, 43.5% of answers were marked with no issues. For optimization 40.63% of answers were marked with no issues. We conclude that readability issues are likely to occur in both systems with no significant impact from the optimization approach.

Table 9

system	mean	std	lower bound	upper bound
same	0.37	0.22	0.3	0.45
neptune	0.34	0.24	0.26	0.42
optimization	0.29	0.23	0.21	0.37

Table 10

34 questions	Neptune	Optimization
NO ISSUES	43.50	40.63
LONG OR COMPLEX SENTENCES: Difficult to follow the answer, unnecessarily long and complicated sentences.	19.50	22.14
HIGHLY TECHNICAL LANGUAGE: Use of highly technical language while it can be presented in simple language.	18.75	14.84
AMBIGUOUS LANGUAGE: Vague and confusing responses with inconclusive answers.	9.75	8.27
IMPROPER FORMATTING: Difficult to read due to improper headers, spaces, paragraph break or lists.	6.50	10.95
OTHER ISSUES: please explain in the text box below.	2.00	3.16

5.4 Experiments on Enterprise Email Generation Applications

We have further integrated our pipeline with an enterprise email generation pipeline for generating marketing content emails (where the token optimzation is applied on the input prompt containing instructions for generating the email). Figure 6 shows the token reduction obtained by applying the heuristics alone on the datasets. Figure 7 shows the box plot distrbution of latency incurred by the heuristics. In terms of quality, we observed there is minimal impact (less than 3% drop) as measured by enterprise specific adherence metrics.

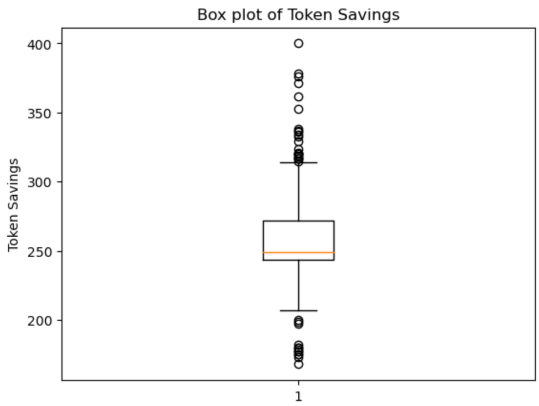

Figure 6: Token reduction on Email Generation datasets.

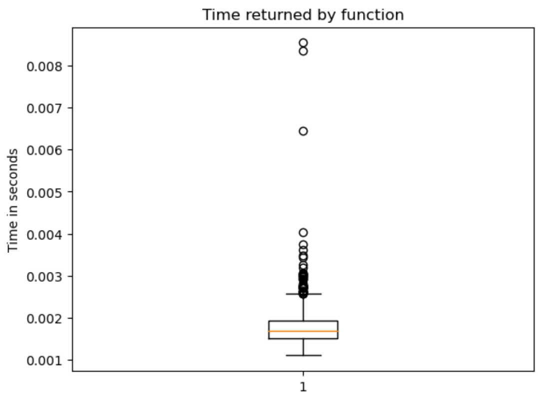

Figure 7: Latency incurred on Email Generation datasets.

5.4.1 Comparison with LLMLingua on enterprise use cases

Apart from the issue of non-interpretability, we found that LLMLingua incurs high latency of the order of 15-20 seconds for enterprise documents where as QRet token optimization heuristics takes milliseconds.

6 Conclusion

We conclude that token optimization is a very effective strategy for reducing token counts in a controllable manner for reducing costs. We release the sentence simplification annotated datasets to the community for further research.

7 Limitations

This is a heuristic method and more comprehensive study needs to be done.

References

[1] NLTK. https://www.nltk.org/.

[2] pyspellchecker. https://pypi.org/project/pyspellchecker/.

[3] thesaurus. https://github.com/zaibacu/thesaurus.

[4] Tiktoken. https://github.com/openai/tiktoken.

[5] Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2016. Ms marco: A human generated machine reading comprehension dataset.

[6] Demian Gholipour Ghalandari, Chris Hokamp, and Georgiana Ifrim. 2022. Efficient unsupervised sentence compression by fine-tuning transformers with reinforcement learning.

[7] Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization.

[8] Lifu Huang, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. Cosmos qa: Machine reading comprehension with contextual commonsense reasoning. *Preprint*, arXiv:1909.00277.

[9] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376.

[10] Remi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain.

[11] Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

[12] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

[13] Hanmeng Liu, Leyang Cui, Jian Liu, and Yue Zhang. 2020. Natural language inference in context – investigating contextual reasoning over long texts. *Preprint*, arXiv:2011.04864.

[14] Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *Preprint*, arXiv:2007.08124.

[15] Louis Martin, Angela Fan, Éric de la Clergerie, Antoine Bordes, and Benoît Sagot. 2020. Muss: Multilingual unsupervised sentence simplification by mining paraphrases.

[16] Louis Martin, Benoît Sagot, Éric de la Clergerie, and Antoine Bordes. 2020. Controllable sentence simplification. *Preprint*, arXiv:1910.02677.

[17] Aaron Mok. 2023. Chatgpt could cost over $700,000 per day to operate. microsoft is reportedly trying to make it cheaper. https://www.businessinsider.in/tech/news/chatgpt-could-cost-over-700000-per-day-to-operate-microsoft-is-reportedly-trying-to-make-it-cheaper-/articleshow/99637548.cms. [Online; accessed Jan-16-2024].

[18] 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.

[19] Eva Sharma, Chen Li, and Lu Wang. 2019. Bigpatent: A large-scale dataset for abstractive and coherent summarization.

[20] Claudia Slowick. 2023. How much does it cost to use gpt models? gpt-3 pricing explained. https://neoteric.eu/blog/how-much-does-it-cost-to-use-gpt-models-gpt-3-pricing-explained/. [Online; accessed Jan-16-2024].

[21] Weihao Yu, Zihang Jiang, Yanfei Dong, and Jiashi Feng. 2020. Reclor: A reading comprehension dataset requiring logical reasoning. *Preprint*, arXiv:2002.04326.

[22] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

[23] Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning.