Fundamental Limits of Prompt Compression: A Rate-Distortion Framework for Black-Box Language Models

Adway Girish [* 1] Alliot Nagle [* 2] Marco Bondaschi [1] Michael Gastpar [1] Ashok Vardhan Makkuva [† 1]
Hyeji Kim [† 2]

Abstract

We formalize the problem of token-level *hard* prompt compression for black-box large language models (LLMs). We derive the distortion-rate function for this setup as a linear program, and provide an efficient algorithm to compute this fundamental limit via its dual. We compare the performance of existing compression schemes with this fundamental limit on a synthetic dataset consisting of prompts generated from a Markov chain, natural language queries, and their respective answers. Our empirical analysis demonstrates the criticality of the compressor being aware of the downstream task/query for the black-box. We observe a large gap between the performance of current prompt compression methods and the optimal strategy, and propose a query-aware, variable-rate adaptation of a prior work to close the gap.

1. Introduction

In spite of the recent success of transformer-based (Vaswani et al., 2017) large language models (LLMs) in language modeling tasks, inference calls to a transformer can be costly in both time and memory usage. Although implementation-level optimizations (Kwon et al., 2023; Dao et al., 2022; Dao, 2023) and architecture-level optimizations and alternatives (Wang et al., 2020; Peng et al., 2023; Gu & Dao, 2023) have been proposed, a third type of optimization that compresses the input (an *input-level* optimization) has the benefit that it directly reduces the resource usage of an LLM inference call, *and* it can be used in conjunction with other types of optimizations for further efficiency gains. It is also the only technique available when seeking to lower costs for black-box API calls to closed-source models, where the associated cost to the caller is determined by the runtime and the number of input tokens. *In this work, we offer a framework and analysis for a recent body of literature in this direction, known as prompt compression* (Askell et al., 2021; Snell et al., 2022; Wingate et al., 2022) (more in App. A).

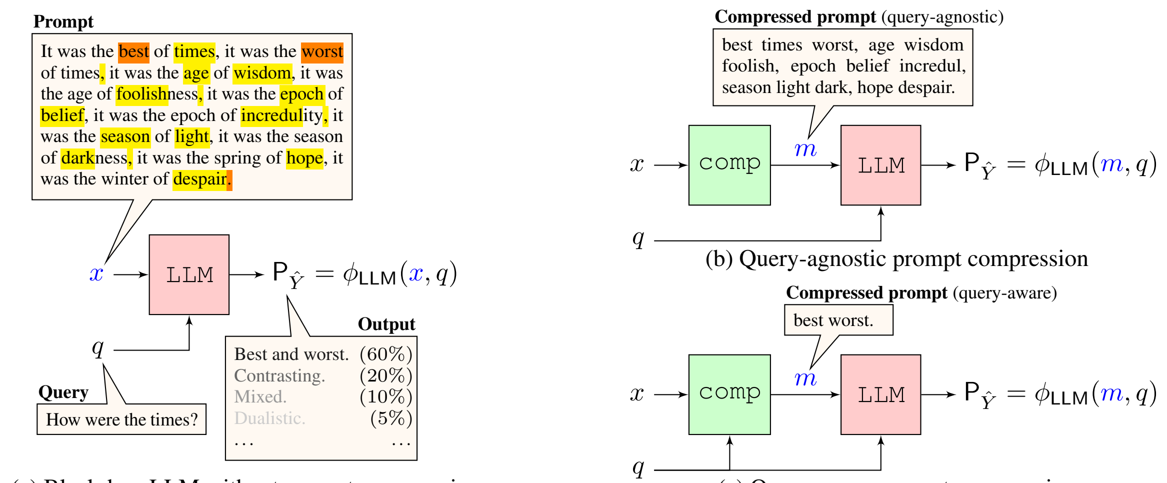
The goal of a prompt compression method is to transform a sequence of input tokens into a shorter sequence of tokens that generates the same semantic response when passed as input to a target LLM, thereby decreasing memory and runtime requirements. In our framework and analysis, we focus on the *prompt compression for black-box models* setting, where the output of a prompt compression method is a set of tokens ("hard prompts") (Li et al., 2023; Jiang et al., 2023c;b; Pan et al., 2024), and exclude methods which output embedding vectors ("soft prompts") (Mu et al., 2023; Ge et al., 2024; Chevalier et al., 2023) as those are not transferable to black-box models.

Despite the progress made in the prompt compression literature, there is a lack of proper formalization of this problem. Though most works propose methods that work well, they offer no insight into key questions, such as *"How far are we from the theoretical limit of the rate-distortion trade-off?"*, *"How essential is the conditioning on the query when compressing the prompt?"*, among others. We offer a unifying framework for the problem of prompt compression and seek to answer these questions with theory and experimental results. Our main contributions can be summarized as:

1. ***Theoretical analysis:*** We formulate prompt compression as a rate-distortion problem (Sec. 2.1), characterize the optimal trade-off between the rate of compression and the distortion incurred, i.e., the *distortion-rate function*, and provide a geometric algorithm to compute it via a dual linear program (Sec. 2.2, Sec. 2.3).
2. ***Evaluation:*** We introduce a synthetic dataset with binary prompts and natural language queries, for which we can compute the distortion-rate function, and compare and obtain insights on existing prompt compression algorithms as in Fig. 2 (Sec. 3).
3. ***Algorithm design:*** We propose "LLMLingua-2 Dynamic," a query-aware, variable-rate adaptation of LLMLingua-2 (Pan et al., 2024) that outperforms all other prompt compression methods on our dataset and has a distortion-rate curve that significantly reduces the gap with the theoretical limit (Sec. 3).

[*]† Equal contribution [1]EPFL [2]UT Austin. Correspondence to: Adway Girish <adway.girish@epfl.ch>, Alliot Nagle <acnagle@utexas.edu>.

Work presented at TF2M workshop at ICML 2024, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).


Prompt
It was the best of times, it was the worst of times, it was the age of wisdom, it was the age of foolishness, it was the epoch of belief, it was the epoch of incredulity, it was the season of light, it was the season of darkness, it was the spring of hope, it was the winter of despair.
x
LLM
$\mathsf{P}_{\hat{Y}} = \phi_{\mathsf{LLM}}(x, q)$
q
Query
How were the times?
Output
Best and worst. (60%)
Contrasting. (20%)
Mixed. (10%)
Dualistic. (5%)
… …
Compressed prompt (query-agnostic)
best times worst, age wisdom foolish, epoch belief incredul, season light dark, hope despair.
x
comp
m
LLM
$\mathsf{P}_{\hat{Y}} = \phi_{\mathsf{LLM}}(m, q)$
q
Compressed prompt (query-aware)
best worst.
x
comp
m
LLM
$\mathsf{P}_{\hat{Y}} = \phi_{\mathsf{LLM}}(m, q)$
q

(a) Black-box LLM without prompt compression.

(b) Query-agnostic prompt compression

(c) Query-aware prompt compression

Figure 1: Model for prompt compression in LLMs. (a): Without prompt compression, the LLM takes a long **Prompt** and **Query** as input, and produces an **Output** distribution. (b), (c): A compressor `comp` shortens the prompt to obtain a **Compressed prompt**, which together with the query, is input to the LLM. (b) `comp` does not have access to the query, and preserves all highlighted tokens. (c) `comp` has access to the query, and preserves only the tokens highlighted in orange.

2. Distortion-rate function for prompt compression

Our model for prompt compression is summarized in Fig. 1. We consider two types of prompt compression, query-agnostic (Fig. 1b, where the compressor does not have access to the query) and query-aware (Fig. 1c, where the compressor has access to the query). We focus on the former and define and characterize the *distortion-rate function*, which describes the optimal trade-off between how much and how well the prompt is compressed. An analogous formalization of the query-aware setup can be found in App. C. Refer to App. B for an overview of the notation.

2.1. A formal model for prompt compression

Black-box LLM. We assume that our target LLM is pretrained LLM with vocabulary $\mathcal{V}$. It takes a pair of the *prompt* $x \in \mathcal{V}^{n_x}$ and the *query* $q \in \mathcal{V}^{n_q}$, $(x, q) \in \mathcal{V}^{n_x+n_q}$ as input The *output* of the LLM is given by $\mathsf{P}_{\hat{Y}} = \phi_{\mathsf{LLM}}(x, q) \in \mathcal{P}(\mathcal{V}^*)$, where $\phi_{\mathsf{LLM}} : \mathcal{V}^* \to \mathcal{P}(\mathcal{V}^*)$ is a deterministic function which maps sequences of tokens to a probability distribution on sequences of tokens. We denote the set of all prompts x by $\mathcal{X}$ and the set of all queries q by $\mathcal{Q}$. We model prompt-query pairs (X, Q) as random variables drawn according to the joint distribution $\mathsf{P}_{XQ} \in \mathcal{P}(\mathcal{X} \times \mathcal{Q})$.

In cases where we have a correct answer $y \in \mathcal{Y} = \mathcal{V}^*$ corresponding to the pair (x, q), we characterize the "closeness" of the LLM output $\mathsf{P}_{\hat{Y}} = \phi_{\mathsf{LLM}}(x, q)$ to the answer y using a distortion measure $\mathsf{d} : \mathcal{Y} \times \mathcal{P}(\mathcal{Y}) \to [0, \infty]$. When dealing with natural language queries, there is no single answer that is uniquely correct. Thus, we allow for a random dependence on x and q by modelling the *answer* as a random variable Y drawn from the distribution $\mathsf{P}_{Y|XQ}(\cdot|x, q)$. We then characterize the "closeness" between the correct answer and the LLM output by the average distortion, given by $\mathbb{E}_{Y\sim\mathsf{P}_{Y|XQ}(\cdot|x,q)}\left[\mathsf{d}\big(Y, \phi_{\mathsf{LLM}}(x, q)\big)\right]$.

Prompt compression. In *query-agnostic* prompt compression (Fig. 1b), the *compressor* `comp` is modelled a random function from $\mathcal{X}$ to $\mathcal{M}$, i.e., the set of all compressed prompts. The compressor takes in the prompt $X \sim \mathsf{P}_X$ and produces a *compressed prompt* $M = \texttt{comp}(X)$ with $\mathrm{len}(M) \le \mathrm{len}(X)$. The user then provides the LLM with the input (M, Q), instead of (X, Q), and the output distribution produced by the LLM is $\mathsf{P}_{\hat{Y}} = \phi_{\mathsf{LLM}}(M, Q)$. To quantify the performance of this compressor `comp`, two quantities are of interest, to measure *how much* and *how well* the prompt is compressed respectively:

(1) the *rate* $\mathbb{E}_{\mathsf{P}_{MXQY}}\left[\frac{\mathrm{len}(M)}{\mathrm{len}(X)}\right]$, and

(2) the *distortion* $\mathbb{E}_{\mathsf{P}_{MXQY}}\left[\mathsf{d}\big(Y, \phi_{\mathsf{LLM}}(M, Q)\big)\right]$,

Clearly, there is a trade-off between these quantities, which we study via the distortion-rate function in Sec. 2.2.

2.2. Rate-distortion formulation for prompt compression

Distortion-rate function $D^*(R)$. The *distortion-rate function* for any compression problem characterizes the fundamental trade-off between the distortion and the rate (Shannon, 1959; Berger, 1971; Cover & Thomas, 2006; El Gamal & Kim, 2011). For a given rate R, the distortion-rate function $D^*(R)$ is the smallest distortion that can be achieved by a compressor with rate at most R.

$D^*(R)$ **for query-agnostic prompt compression.** We model the compressor as a random function from $\mathcal{X}$ to $\mathcal{M}$, which, by the functional representation lemma (El Gamal & Kim, 2011; Hajek & Pursley, 1979), is equivalent to a conditional distribution $\mathsf{P}_{M|X}$. Thus, we can explicitly write

$$\begin{aligned} D^*(R) = \inf_{\mathsf{P}_{M|X}} \quad & \mathbb{E}_{\mathsf{P}_{MXQY}}\left[\mathsf{d}\big(Y, \phi_{\mathsf{LLM}}(M,Q)\big)\right] \\ \text{s.t.} \quad & \mathsf{P}_{M|X} \text{ is a compressor, and} \\ & \mathbb{E}_{\mathsf{P}_{MXQY}}\left[\frac{\operatorname{len}(M)}{\operatorname{len}(X)}\right] \le R, \end{aligned} \tag{1}$$

The constraint "$\mathsf{P}_{M|X}$ is a compressor" is short for: (1) for each $x \in \mathcal{X}$, $\sum_{m\in\mathcal{M}} \mathsf{P}_{M|X}(m|x) = 1$, and (2) if $\operatorname{len}(m) > \operatorname{len}(x)$, then $\mathsf{P}_{M|X}(m|x) = 0$. The extension to the query-aware setting is straightforward; we then have query-dependent (or conditional) distortion-rate functions $D_q^*(R)$ for each $q \in \mathcal{Q}$, and an average distortion-rate function, denoted by $\bar{D}^*(R)$ (refer to App. C).

We provide an overview of rate-distortion theory from the information theory literature in App. E and describe how this setup compares; in short, this is a rate-distortion problem with side information for function computing *with a "fixed decoder"*, which has not been studied before.

2.3. Linear program formulation of the distortion-rate function

We now focus on solving the optimization problem in (1), where the objective and constraints are linear in $\mathsf{P}_{M|X}$. Though it is a linear program (LP), which is simple from an optimization perspective (Boyd & Vandenberghe, 2004; Dantzig, 2002), the dimensions of problem are still large and solving the LP directly using off-the-shelf solvers is infeasible. We first rewrite (1) as an explicit LP using optimization-theoretic notation, and hide the probabilistic notation involving expectations and conditional probabilities in the parameters of the LP.

Proposition 1 (Primal LP). *The distortion-rate function for query-agnostic prompt compression* (1) *is given by the solution to the linear program*

$$\begin{aligned} D^*(R) = \inf_{\left(\boldsymbol{z}_x \in \mathbb{R}_+^{\mathcal{M}_x}\right)_{x\in\mathcal{X}}} \quad & \sum_{x\in\mathcal{X}} \boldsymbol{D}_x^\top \boldsymbol{z}_x \\ \textit{s.t.} \quad & \sum_{x\in\mathcal{X}} \boldsymbol{R}_x^\top \boldsymbol{z}_x \le R, \\ & \mathbf{1}^\top \boldsymbol{z}_x = 1, \quad \forall x \in \mathcal{X}, \end{aligned} \tag{LP}$$

where for each $x \in \mathcal{X}$, $\mathcal{M}_x$ denotes the set of compressed prompts associated to x, i.e., the set of all possible token sequences of length smaller than $\operatorname{len}(x)$*, the vectors $\boldsymbol{z}_x \in \mathbb{R}_+^{\mathcal{M}_x}$ are the optimization variables and the constants $\boldsymbol{D}_x, \boldsymbol{R}_x \in \mathbb{R}_+^{\mathcal{M}_x}$ with components indexed by $m \in \mathcal{M}_x$ are given by*

$$\begin{aligned} \boldsymbol{D}_{x,m} &\triangleq \mathsf{P}_X(x)\, \mathbb{E}_{\mathsf{P}_{QY|MX}(\cdot,\cdot|m,x)}\left[\mathsf{d}(Y, \phi_{\mathsf{LLM}}(m,Q))\right], \\ \boldsymbol{R}_{x,m} &\triangleq \mathsf{P}_X(x)\, \frac{\operatorname{len}(m)}{\operatorname{len}(x)}. \end{aligned} \tag{2}$$

Though solving (LP) directly may be infeasible, the dual of (LP) can be solved efficiently using a simple geometric algorithm. We characterize the dual in Thm. 1; refer to App. D.1 for a full proof. The algorithm, given in Alg. 1, takes as input R, $(\boldsymbol{D}_x)_{x\in\mathcal{X}}$, $(\boldsymbol{R}_x)_{x\in\mathcal{X}}$, and returns as output the distortion-rate function at R, i.e., $D^*(R)$; refer to App. D.2 for a step-by-step example showing its working.

Theorem 1 (Dual LP). *The distortion-rate function for query-agnostic prompt compression* (1) *is given by the solution to the dual of the linear program* (LP)*, i.e.,*

$$D^*(R) = \sup_{\lambda \ge 0} \left\{ -\lambda R + \sum_{x\in\mathcal{X}} \min_{m\in\mathcal{M}_x} \left[\boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m}\right] \right\}. \tag{dual-LP}$$

Algorithm 1: To compute $D^*(R)$ via (dual-LP)

```
Input: $R$, $(\boldsymbol{D}_x)_{x\in\mathcal{X}}$, $(\boldsymbol{R}_x)_{x\in\mathcal{X}}$
Output: $D^*(R)$, the distortion-rate function at rate $R$
1  for $x \in \mathcal{X}$ do
2      Find $\mathcal{M}_{\text{env}}^{(x)} \subseteq \mathcal{M}_x$ such that $\{(\boldsymbol{R}_{x,m}, \boldsymbol{D}_{x,m})\}_{m\in\mathcal{M}_{\text{env}}^{(x)}}$
         are on the lower-left concave boundary of
         $\{(\boldsymbol{R}_{x,m}, \boldsymbol{D}_{x,m})\}_{m\in\mathcal{M}_x}$ ;    ▷ see Fig. 3 for an example
3      $\left\{m_1^{(x)}, m_2^{(x)}, \dots, m_{k_x}^{(x)}\right\} \leftarrow \mathcal{M}_{\text{env}}^{(x)}$ ordered such that
         $\boldsymbol{R}_{x,m_{k_x}^{(x)}} > \dots > \boldsymbol{R}_{x,m_1^{(x)}}$;
4      for $i = 1, \dots, k_x - 1$ do $\lambda_i^{(x)} \leftarrow \frac{\boldsymbol{D}_{x,m_i^{(x)}} - \boldsymbol{D}_{x,m_{i+1}^{(x)}}}{\boldsymbol{R}_{x,m_{i+1}^{(x)}} - \boldsymbol{R}_{x,m_i^{(x)}}}$;
5      $\lambda_0^{(x)} \leftarrow +\infty$; $\lambda_{k_x}^{(x)} \leftarrow 0$;
6      $\Lambda^{(x)} \leftarrow \left\{\lambda_0^{(x)}, \lambda_1^{(x)}, \dots, \lambda_{k_x-1}^{(x)}, \lambda_{k_x}^{(x)}\right\}$;
7  $\left\{\widetilde{\lambda}_0, \dots, \widetilde{\lambda}_k\right\} \leftarrow \bigcup_{x\in\mathcal{X}} \Lambda^{(x)}$ with
     $+\infty = \widetilde{\lambda}_0 > \widetilde{\lambda}_1 > \dots > \widetilde{\lambda}_{k-1} > \widetilde{\lambda}_k = 0$;
8  for $x \in \mathcal{X}$ do
9      for $j = 1, \dots, k$ do
10         Set $\widetilde{m}_j^{(x)} \leftarrow m_i^{(x)}$ where
             $i \in \{1, \dots, k_x\} : \left(\lambda_i^{(x)}, \lambda_{i-1}^{(x)}\right) \supseteq \left(\widetilde{\lambda}_j, \widetilde{\lambda}_{j-1}\right)$
11 for $j = 1, \dots, k$ do
12     if $\sum_{x\in\mathcal{X}} \boldsymbol{R}_{x,\widetilde{m}_j^{(x)}} > R$ then $\lambda_j \leftarrow \widetilde{\lambda}_{j-1}$
         else $\lambda_j \leftarrow \widetilde{\lambda}_j$ ;
13     $D_j \leftarrow -\lambda_j R + \sum_{x\in\mathcal{X}} \left[\boldsymbol{D}_{x,\widetilde{m}_j^{(x)}} + \lambda_j \boldsymbol{R}_{x,\widetilde{m}_j^{(x)}}\right]$;
14 Return $\max_{j=1,\dots,k} D_j$ ;    ▷ $= D^*(R)$
```

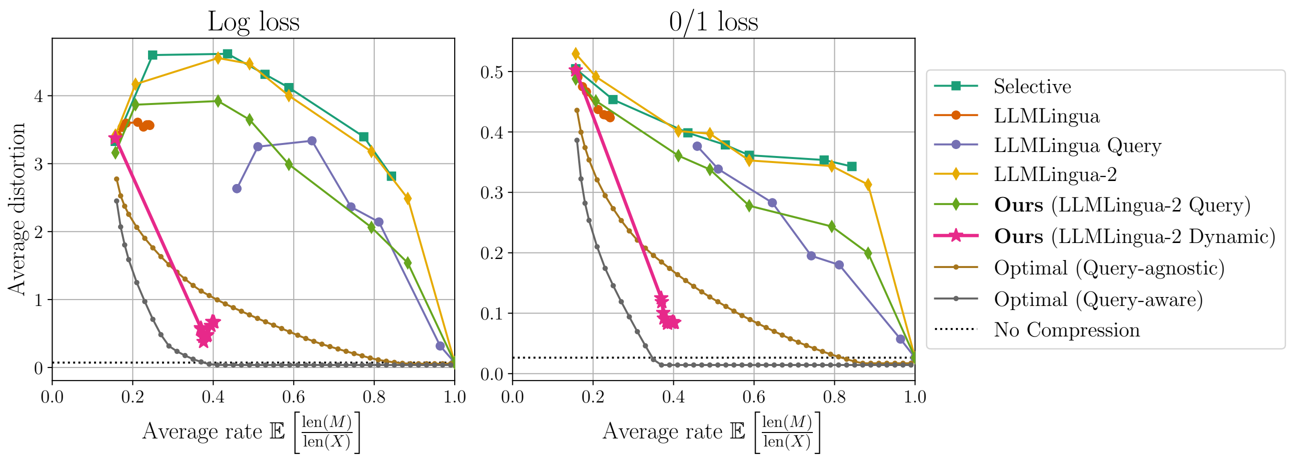


Figure 2: The rate-distortion trade-offs of prompt compression methods compared against the query-agnostic and query-aware theoretical limits. The distortion measures used are log loss **(left)** and 0/1 loss **(right)** as defined in (8).

3. Experiments

Our main experimental contribution is summarized in Fig. 2. We compare the performance of existing prompt compression methods with the theoretical limit, the distortion-rate function, for a synthetic dataset. We briefly summarize our experimental setup here; refer to App. F for a more elaborate explanation.

Dataset. To run experiments that are computationally tractable but still meaningful to the prompt compression problem, we construct a synthetic dataset $\{(x_i, q_i, y_i)\}_{i=1}^N$ with (1) binary prompts x_i with $4 \leq \text{len}(x_i) \leq 10$, drawn from a Markov chain (e.g., "00101") , (2) natural language queries q_i (e.g., "Compute the parity"), and (3) their associated answers y_i (e.g., "0"). The optimal distortion-rate function is computed using Alg. 1, taking P_{XQY} to be the empirical distribution $\mathsf{P}_{XQY} = \frac{1}{N}\sum_{i=1}^N \delta_{(x_i,q_i,y_i)}$. We take $N = 1400$; refer to App. H.1 for a complete specification and to App. G for more details about this choice.

Baseline methods and models. We compare the optimal rate-distortion trade-off (both query-aware and query-agnostic) with prompt compression methods that can be used to compress prompts for a black-box target LLM: Selective Context (Li et al., 2023), LLMLingua (Jiang et al., 2023c), LLMLingua Query (Jiang et al., 2023b), LLMLingua-2 (Pan et al., 2024). We use Mistral 7B Instruct v0.2 (Jiang et al., 2023a) as our black-box target LLM.

Our proposed methods. We add two novel contributions over LLMLingua-2: (1) we adapt LLMLingua-2 to the query-aware setting, whereas the original work only proposed the query-agnostic approach, which we call "LLMLingua-2 Query," and (2) we further adapt this query-aware approach into a variable-rate approach we refer to as "LLMLingua-2 Dynamic." Refer to App. F for more details.

Main results. From Fig. 2, we observe that (1) most existing methods are far from the theoretical limit, suggesting that there is still room for improvement in this field, (2) conditioning on the query allows for a significant improvement, as seen by the performance of our query-aware adaptation LLMLingua-2 Query against the query-agnostic LLMLingua-2 (Pan et al., 2024), and (3) our proposed method LLMLingua-2 Dynamic, a query-aware adaptation of LLMLingua-2, achieves the best performance among all methods considered, and is the only method to outperform the optimal query-agnostic strategy. Moreover, Fig. 7 shows that, for select queries, e.g., "Is the binary string a palindrome?" LLMLingua-2 Dynamic is the only method that matches the optimal *query-aware* strategy in certain ranges of the average rate. In App. I, we provide additional results showing how the choice of query affects the theoretical limit (since some queries are easier to compress for than others) and the effect of tokenization, as described in App. F.

4. Conclusion

We characterize the optimal rate-distortion trade-off for prompt compression for black-box target LLMs as an LP, and derive an algorithm to solve this efficiently via its dual, for both query-agnostic and query-aware settings. We compare these optimal curves with existing prompt compression methods and adapt one of them, LLMLingua-2, to be query-aware and variable-rate; this modified method, which we call "LLMLingua-2 Dynamic," exhibits superior performance. As future work, it is important to exhaustively study our proposed method on natural language datasets. An open problem that remains is to identify a method to compute the optimal rate-distortion trade-off for natural language datasets; we provide preliminary results in this direction in App. G.

Acknowledgements

The authors would like to thank Ananda Theertha Suresh for introducing them to the problem of prompt compression. AG would like to thank Emre Telatar for helpful discussions on the problem formulation.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Askell, A., Bai, Y., Chen, A., Drain, D., Ganguli, D., Henighan, T., Jones, A., Joseph, N., Mann, B., DasSarma, N., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Kernion, J., Ndousse, K., Olsson, C., Amodei, D., Brown, T., Clark, J., McCandlish, S., Olah, C., and Kaplan, J. A general language assistant as a laboratory for alignment, 2021.

Berger, T. *Rate Distortion Theory: A Mathematical Basis For Data Compression*. Prentice-Hall, Inc., Englewood Cliffs, NJ, 1971.

Biderman, S., Schoelkopf, H., Anthony, Q., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., Skowron, A., Sutawika, L., and Van Der Wal, O. Pythia: a suite for analyzing large language models across training and scaling. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Boyd, S. P. and Vandenberghe, L. *Convex optimization*. Cambridge university press, 2004.

Chevalier, A., Wettig, A., Ajith, A., and Chen, D. Adapting language models to compress contexts. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3829–3846, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.232. URL https://aclanthology.org/2023.emnlp-main.232.

Cover, T. M. and Thomas, J. A. *Elements of Information Theory (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience, USA, 2006. ISBN 0471241954.

Dantzig, G. B. Linear programming. *Operations research*, 50(1):42–47, 2002.

Dao, T. FlashAttention-2: Faster attention with better parallelism and work partitioning. 2023.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, 2022.

El Gamal, A. and Kim, Y.-H. *Network Information Theory*. Cambridge University Press, 2011.

Ge, T., Jing, H., Wang, L., Wang, X., Chen, S.-Q., and Wei, F. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=uREj4ZuGJE.

Ghalandari, D., Hokamp, C., and Ifrim, G. Efficient unsupervised sentence compression by fine-tuning transformers with reinforcement learning. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1267–1280, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.90. URL https://aclanthology.org/2022.acl-long.90.

Gray, R. Conditional rate-distortion theory. Technical report, Stanford University, 1972.

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

Hajek, B. and Pursley, M. B. Evaluation of an achievable rate region for the broadcast channel. *IEEE Transactions on Information Theory*, 25(1):36–46, 1979.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023a.

Jiang, H., Wu, Q., , Luo, X., Li, D., Lin, C.-Y., Yang, Y., and Qiu, L. LongLLMLingua: Accelerating and enhancing llms in long context scenarios via prompt compression. *ArXiv preprint*, abs/2310.06839, 2023b. URL https://arxiv.org/abs/2310.06839.

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. LLMLingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376. Association for Computational Linguistics, December 2023c. doi: 10.18653/v1/2023.emnlp-main.825. URL https://aclanthology.org/2023.emnlp-main.825.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, pp. 611–626, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400702297. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Li, Y., Dong, B., Lin, C., and Guerin, F. Compressing context to enhance inference efficiency of large language models, 2023.

Liu, N. F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., and Liang, P. Lost in the middle: How language models use long contexts, 2023. arXiv:2307.03172.

Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. Ro{bert}a: A robustly optimized {bert} pretraining approach, 2020. URL https://openreview.net/forum?id=SyxS0T4tvS.

Mu, J., Li, X. L., and Goodman, N. Learning to compress prompts with gist tokens. 2023.

Niu, T., Xiong, C., and Socher, R. Deleter: Leveraging bert to perform unsupervised successive text compression, 2019.

OpenAI. Gpt-4 technical report, 2024.

Pan, Z., Wu, Q., Jiang, H., Xia, M., Luo, X., Zhang, J., Lin, Q., Ruhle, V., Yang, Y., Lin, C.-Y., Zhao, H. V., Qiu, L., and Zhang, D. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *ArXiv preprint*, abs/2403.12968, 2024. URL https://arxiv.org/abs/2403.12968.

Peng, B., Alcaide, E., Anthony, Q. G., Albalak, A., Arcadinho, S., Biderman, S., Cao, H., Cheng, X., Chung, M. N., Derczynski, L., Du, X., Grella, M., GV, K. K., He, X., Hou, H., Kazienko, P., Kocon, J., Kong, J., Koptyra, B., Lau, H., Lin, J., Mantri, K. S. I., Mom, F., Saito, A., Song, G., Tang, X., Wind, J. S., Woźniak, S., Zhang, Z., Zhou, Q., Zhu, J., and Zhu, R.-J. RWKV: Reinventing RNNs for the transformer era. In *The 2023 Conference on Empirical Methods in Natural Language Processing*, 2023. URL https://openreview.net/forum?id=7SaXczaBpG.

Schumann, R., Mou, L., Lu, Y., Vechtomova, O., and Markert, K. Discrete optimization for unsupervised sentence summarization with word-level extraction. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5032–5042, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.452. URL https://aclanthology.org/2020.acl-main.452.

Shannon, C. E. Coding theorems for a discrete source with a fidelity criterion. *IRE Nat. Conv. Rec*, 4(142-163):1, 1959.

Simons, S. *Minimax Theorems and Their Proofs*, pp. 1–23. Springer US, Boston, MA, 1995. ISBN 978-1-4613-3557-3. doi: 10.1007/978-1-4613-3557-3_1. URL https://doi.org/10.1007/978-1-4613-3557-3_1.

Snell, C., Klein, D., and Zhong, R. Learning by distilling context, 2022.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

von Neumann, J. Zur Theorie der Gesellschaftsspiele. *Math. Ann.*, 100(1):295–320, 1928. ISSN 0025-5831,1432-1807. doi: 10.1007/BF01448847. URL https://doi.org/10.1007/BF01448847.

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity, 2020.

Weissman, T. and El Gamal, A. Source coding with limited-look-ahead side information at the decoder. *IEEE Transactions on Information Theory*, 52(12):5218–5239, 2006. doi: 10.1109/TIT.2006.885500.

Wingate, D., Shoeybi, M., and Sorensen, T. Prompt compression and contrastive conditioning for controllability and toxicity reduction in language models. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2022*, pp. 5621–5634, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-emnlp.412. URL https://aclanthology.org/2022.findings-emnlp.412.

Wyner, A. D. The rate-distortion function for source coding with side information at the decoder-II: General sources. *Information and Control*, 38(1):60–80, 1978. ISSN 0019-9958. doi: https://doi.org/10.1016/S0019-9958(78)90034-7. URL https://www.sciencedirect.com/science/article/pii/S0019995878900347.

Wyner, A. D. and Ziv, J. The rate-distortion function for source coding with side information at the decoder. *IEEE Transactions on Information Theory*, 22(1):1–10, 1976. doi: 10.1109/TIT.1976.1055508.

Yamamoto, H. Wyner-Ziv theory for a general function of the correlated sources (corresp.). *IEEE Transactions on Information Theory*, 28(5):803–807, 1982. doi: 10.1109/TIT.1982.1056560.

The appendix is organized as follows:

1. In App. A, we describe the recent literature in the field of prompt compression.
2. In App. B, we provide a complete summary of the notation used throughout the paper.
3. In App. C, we formally characterize the distortion-rate functions for the query-aware prompt compression setting.
4. In App. D, we prove the main result of the paper (Thm. 1) and give a detailed description of the working of Alg. 1.
5. In App. E, we provide an overview of the information theory literature on rate-distortion theory, and explain how our model compares.
6. In App. F, we provide more details on the experimental as described briefly in Sec. 3.
7. In App. G, we describe the limitations of our paper, and provide suggestions on how these limitations may be overcome.
8. In App. H, we provide details on the technical system details used for the experiments in Sec. 3 and App. I.
9. In App. I, we show additional experimental results.

A. Background and related works

Long prompts slow the inference process due to the increase in the number of tokens that the LLM has to process. It is also known that very long prompts can cause LLMs to ignore or "forget" parts of the input and produce erroneous answers (Liu et al., 2023). Therefore, studying how these prompts can be compressed is essential. As illustrated in Fig. 1, we wish to design a compressor that, upon receiving the prompt, produces a "compressed" version in the sense that it should have fewer tokens than the prompt called the *compressed prompt*, such that a target LLM is able to give answers that are "close enough," in the sense of some appropriately chosen metric, to the ground truth. Though similar in spirit to text summarization, prompt compression has the advantage that the compressed prompt is not required to be human-readable.

All prompt compression methods belong to one of two groups: those that compress the prompt into *soft prompts* and those that compress the prompt into *hard prompts*. In soft-prompt compression, the compressor is trained to transform the input prompt into a set of embedding vectors (sometimes referred to as "soft tokens") that do not map back into the token space. These methods, including Gist Tokens (Mu et al., 2023), AutoCompressor (Chevalier et al., 2023), and In-Context Auto-Encoder (Ge et al., 2024) are trained end-to-end and require specialized fine-tuning of the target LLM to interpret the soft prompt inputs.

In this work, we focus instead on methods that compress the prompt into hard prompts, where the compressor's output is a set of tokens. While it is technically feasible to fine-tune the target LLM in this setting, it is unnecessary and often avoided because the utility of this setting is compressing prompts for black-box models that are not fine-tuned. These methods often use either the target LLM, or a smaller and faster LLM, to compress the prompt. The basic idea behind all these methods is to identify the tokens that are "most relevant," per an appropriate metric, and retain as many of them in the compressed prompt as possible. These methods include Selective Context (Li et al., 2023), LLMLingua (Jiang et al., 2023c), LLMLingua-2 (Pan et al., 2024), and LongLLMLingua (Jiang et al., 2023b). More details on these works can be found in App. F. Precursors to the prompt compression works include text compression methods, which have the added constraint that the compressed text is human-readable (Niu et al., 2019; Schumann et al., 2020; Ghalandari et al., 2022). Prompt compression methods are different from these in that the text only needs to be interpretable by the target LLM, not by a human.

We offer a framework for hard-prompt compression methods where we assume that a query is provided in addition to the compressed prompt during the target LLM inference call. Functionally, this is the most useful interpretation of prompt compression since it clarifies that the goal is to compress the prompt for a given query/task. This setting is also used in the LLMLingua and LongLLMLingua works, and is slightly more general than the setting where no query is used (in that case, the query can be empty).

B. Notation

General. We use $\triangleq$ to signify a definition. For a set $\mathcal{X}$, with $x_i \in \mathcal{X}$ for $i = 1, \ldots, n$, we represent the sequence $(x_1, \ldots, x_n)$ by $x^n \in \mathcal{X}^n$, which is short for $\mathcal{X} \times \cdots \times \mathcal{X}$. We use $\mathcal{X}^*$ to denote $\bigcup_{n \geq 1} \mathcal{X}^n$, the set of all nonempty finite-length sequences on $\mathcal{X}$. In general, for $y \in \mathcal{X}^n$, we denote the length of y by $\mathrm{len}(y) = n$. We denote the cardinality of a set $\mathcal{X}$ by $|\mathcal{X}|$. We use $\mathbb{R}_+$ to denote the set of nonnegative real numbers. For a set $\mathcal{X} = \{x_1, \ldots, x_k\}$, we use the boldface $(\boldsymbol{A}_x)_{x \in \mathcal{X}}$ to denote the vector $(\boldsymbol{A}_{x_1}, \ldots, \boldsymbol{A}_{x_k})$ indexed by elements of $\mathcal{X}$. We also write $\mathbb{R}_+^{\mathcal{X}} = \{(\boldsymbol{v}_x)_{x \in \mathcal{X}} : \boldsymbol{v}_x \in \mathbb{R}_+ \text{ for each } x \in \mathcal{X}\}$. We use $\mathbb{1}$ to denote the indicator function, which takes the value 1 when its argument is true and 0 otherwise. The infimum and supremum of a set of values is denoted using $\inf$ and $\sup$ respectively. We use $\mathbf{0}$ and $\mathbf{1}$ to denote the vectors of appropriate dimension with all elements equal to 0 and 1 respectively.

Probability. We deal with discrete probability distributions on finite sets, for which we use calligraphic letters to denote the set (e.g. $\mathcal{X}$), uppercase letters to denote the random variable (r.v., e.g. X) and lowercase letters to denote samples of the r.v. (e.g. x). The set of all probability distributions on the set $\mathcal{X}$ is denoted by $\mathscr{P}(\mathcal{X})$. The probability distribution of the r.v. X on $\mathcal{X}$ is denoted by $\mathsf{P}_X \in \mathscr{P}(\mathcal{X})$, and we say $X \sim \mathsf{P}_X$. For a (measurable) function f, the expectation of the r.v. $f(X)$ is denoted by $\mathbb{E}_{X \sim \mathsf{P}_X}[f(X)]$, $\mathbb{E}_{\mathsf{P}_X}[f(X)]$, or $\mathbb{E}[f(X)]$, with the subscripts dropped when the distribution and/or r.v.'s are clear from context. The degenerate probability distribution with mass 1 at $x \in \mathcal{X}$ is represented by $\delta_x \in \mathscr{P}(\mathcal{X})$. Conditional probabilities from $\mathcal{X}$ to $\mathcal{Y}$ are denoted as $\mathsf{P}_{Y|X}$, and for each $x \in \mathcal{X}$, we denote the distribution on Y as $\mathsf{P}_{Y|X}(y|x)$.

Problem-setup-specific notation. In our model, we use $\mathcal{V}$ to refer to the vocabulary of the prompt. We use the uppercase letters X to refer to the prompt, M to refer to the compressed prompt, Q to refer to the query and Y to refer to the answer as random variables (and corresponding calligraphic and lowercase letters to denote the set and samples of the r.v. respectively). We use $\mathsf{P}_{\hat{Y}}$ to refer to the output distribution of the LLM, which is modelled as the function ϕ_{LLM}. For a given prompt x, we use $\mathcal{M}_x$ to refer to the set of possible compressed prompts, which is the set of all sequences of length smaller than $\mathrm{len}(x)$. To denote the distortion measure, we use d, which can be either the log loss $\mathsf{d}_{\log}$ or the 0/1 loss $\mathsf{d}_{0/1}$. We denote the query-agnostic distortion-rate function at rate R by $D^*(R)$. The average query-aware distortion-rate function is denoted by $\bar{D}^*(R)$, and the conditional query-aware distortion rate function for query $q \in \mathcal{Q}$ is given by $D_q^*(R)$.

C. Extensions to query-aware prompt compression

As mentioned in Sec. 2.2 and Sec. 2.3, analogous definitions and results can be obtained for the query-aware setting as well. The difference is that the compressor has access to the query in addition to the prompt. Thus, the compressor $\mathtt{comp}$ is a possibly random function from $\mathcal{X} \times \mathcal{Q}$ to $\mathcal{M}$. For the query Q, the compressor maps the prompt X to the compressed prompt $M = \mathtt{comp}(X, Q)$ with $\mathrm{len}(M) \leq \mathrm{len}(X)$. The user provides the input $[M, Q]$ to the LLM, which produces the output distribution $\mathsf{P}_{\hat{Y}} = \phi_{\mathsf{LLM}}(M, Q)$. Just as in the query-agnostic setting, two quantities of interest are

$$\text{(1) the (average) rate } \mathbb{E}\left[\frac{\mathrm{len}(M)}{\mathrm{len}(X)}\right], \quad \text{and} \quad \text{(2) the distortion } \mathbb{E}\left[\mathsf{d}_{\log}(Y, \phi_{\mathsf{LLM}}(M, Q))\right],$$

with both expectations taken with respect to the joint distribution P_{MXQY}. Since different queries may require different amounts of information to be preserved during compression, it is also of interest to define the (conditional) rate and distortion *for the specific query* q as $\mathbb{E}\left[\frac{\mathrm{len}(M)}{\mathrm{len}(X)}\right]$ and $\mathbb{E}\left[\mathsf{d}_{\log}(Y, \phi_{\mathsf{LLM}}(M, q))\right]$ respectively, with both expectations taken with respect to the joint distribution $\mathsf{P}_{MXY|Q}(\cdot|q)$.

The rate-distortion problem for query-aware prompt compression can be also formulated similarly to (1). We model $\mathtt{comp}$ as a random mapping $\mathsf{P}_{M|XQ}$ from $\mathcal{X} \times \mathcal{Q}$ to $\mathcal{M}$. Then, the (average) distortion-rate function at rate R is the smallest distortion that can be achieved by a query-aware compressor with rate at most R, given by

$$\begin{aligned} \bar{D}^*(R) = \quad & \inf_{\mathsf{P}_{M|XQ}} \quad \mathbb{E}\left[\mathsf{d}_{\log}\big(Y, \phi_{\mathsf{LLM}}(M, Q)\big)\right] \\ & \text{s.t.} \quad \mathsf{P}_{M|XQ} \text{ is a compressor, and} \\ & \qquad \mathbb{E}\left[\frac{\mathrm{len}(M)}{\mathrm{len}(X)}\right] \leq R, \end{aligned} \tag{3}$$

with both expectations taken with respect to the joint distribution $\mathsf{P}_{MXQY} = \mathsf{P}_{M|XQ}\mathsf{P}_{XQY}$ induced by the compressor. The condition "$\mathsf{P}_{M|XQ}$ is a compressor" is short for (1) for each $x \in \mathcal{X}$ and $q \in \mathcal{Q}$, $\sum_{m \in \mathcal{M}} \mathsf{P}_{M|XQ}(m|x, q) = 1$, and (2) $\mathsf{P}_{M|XQ}(m|x, q) = 0$ if $\mathrm{len}(m) > \mathrm{len}(x)$. Similarly, the (conditional) distortion-rate function at rate R is the smallest distortion that can be achieved by a query-aware compressor for query q at rate at most R, given by

$$\begin{aligned} D_q^*(R) = \inf_{\mathsf{P}_{M|XQ}(\cdot|\cdot,q)} \quad & \mathbb{E}\left[\mathsf{d}_{\log}\big(Y, \phi_{\mathsf{LLM}}(M,Q)\big)\right] \\ \text{s.t.} \quad & \mathsf{P}_{M|XQ}(\cdot|\cdot,q) \text{ is a compressor, and} \\ & \mathbb{E}\left[\frac{\operatorname{len}(M)}{\operatorname{len}(X)}\right] \le R, \end{aligned} \tag{4}$$

with both expectations taken with respect to $\mathsf{P}_{MXY|Q}(\cdot,\cdot,\cdot|q)$.

Just like the query-agnostic setting, note that both (3) and (4) are linear programs, as the objective and constraints are all linear in $\mathsf{P}_{M|XQ}$ and $\mathsf{P}_{M|XQ}(\cdot|\cdot,q)$ respectively. We obtain explicit linear programs analogous to (LP) by defining constants $\bar{\boldsymbol{D}}_x^q$ and $\bar{\boldsymbol{R}}_x^q \in \mathbb{R}_+^{\mathcal{M}_x}$ for the average distortion-rate function and $\boldsymbol{D}_x^q$ and $\boldsymbol{R}_x^q \in \mathbb{R}_+^{\mathcal{M}_x}$ for the conditional distortion-rate functions, for each $x \in \mathcal{X}$ and $q \in \mathcal{Q}$, similarly to (2).

Proposition 2 (Query-aware primal LPs)**.** *The (average) distortion-rate function for query-aware prompt compression* (3) *is given by the solution to*

$$\begin{aligned} \bar{D}^*(R) = \inf_{\left(\boldsymbol{z}_{x,q} \in \mathbb{R}_+^{\mathcal{M}_x}\right)_{x\in\mathcal{X}, q\in\mathcal{Q}}} \quad & \sum_{x\in\mathcal{X}, q\in\mathcal{Q}} \bar{\boldsymbol{D}}_x^{q\top} \boldsymbol{z}_{x,q} \\ \textit{s.t.} \quad & \sum_{x\in\mathcal{X}, q\in\mathcal{Q}} \bar{\boldsymbol{R}}_x^{q\top} \boldsymbol{z}_{x,q} \le R, \\ & \mathbf{1}^\top \boldsymbol{z}_{x,q} = 1, \quad \forall\, x \in \mathcal{X},\, q \in \mathcal{Q}. \end{aligned} \tag{avg-cond-LP}$$

The (conditional) distortion-rate function for query-aware prompt compression for query q (4) *is given by the solution to*

$$\begin{aligned} D_q^*(R) = \inf_{\left(\boldsymbol{z}_x \in \mathbb{R}_+^{\mathcal{M}_x}\right)_{x\in\mathcal{X}}} \quad & \sum_{x\in\mathcal{X}} \boldsymbol{D}_x^{q\top} \boldsymbol{z}_x \\ \textit{s.t.} \quad & \sum_{x\in\mathcal{X}} \boldsymbol{R}_x^{q\top} \boldsymbol{z}_x \le R, \\ & \mathbf{1}^\top \boldsymbol{z}_x = 1, \quad \forall\, x \in \mathcal{X}. \end{aligned} \tag{cond-LP}$$

For each $x \in \mathcal{X}$, $\mathcal{M}_x$ denotes the set of all possible compressed prompts associated to x, i.e., the set of all possible token sequences of length at most $\operatorname{len}(x)$*, the vectors $\boldsymbol{z}_{x,q} \in \mathbb{R}_+^{\mathcal{M}_x}$ for $q \in \mathcal{Q}$ and $\boldsymbol{z}_x \in \mathbb{R}_+^{\mathcal{M}_x}$ are the optimization variables respectively and the constants $\bar{\boldsymbol{D}}_x^q, \bar{\boldsymbol{R}}_x^q, \boldsymbol{D}_x^q, \boldsymbol{R}_x^q \in \mathbb{R}_+^{\mathcal{M}_x}$ are given by*

$$\bar{\boldsymbol{D}}_{x,m}^q \triangleq \mathsf{P}_{XQ}(x,q)\, \mathbb{E}\left[\mathsf{d}_{\log}(Y, \phi_{\mathsf{LLM}}(m,q))\right] \;\; \textit{and} \;\; \bar{\boldsymbol{R}}_{x,m}^q \triangleq \mathsf{P}_{XQ}(x,q) \frac{\operatorname{len}(m)}{\operatorname{len}(x)}, \quad m \in \mathcal{M}_x, \tag{5}$$

$$\boldsymbol{D}_{x,m}^q \triangleq \mathsf{P}_{X|Q}(x|q)\, \mathbb{E}\left[\mathsf{d}_{\log}(Y, \phi_{\mathsf{LLM}}(m,q))\right] \;\; \textit{and} \;\; \boldsymbol{R}_{x,m}^q \triangleq \mathsf{P}_{X|Q}(x|q) \frac{\operatorname{len}(m)}{\operatorname{len}(x)}, \quad m \in \mathcal{M}_x, \tag{6}$$

with the expectation taken with respect to $\mathsf{P}_{Y|MXQ}(\cdot|m,x,q)$.

Proof. This follows immediately from (3) and (4) by defining the constants $\bar{\boldsymbol{D}}_x^q, \bar{\boldsymbol{R}}_x^q, \boldsymbol{D}_x^q, \boldsymbol{R}_x^q \in \mathbb{R}_+^{\mathcal{M}_x}$ for each $x \in \mathcal{X}$ and $q \in \mathcal{Q}$ as given in (5) and (6) and taking $\boldsymbol{z}_{x,q}$ and $\boldsymbol{z}_x$ to be $\mathsf{P}_{M|XQ}(\cdot|x,q)$ respectively,. We use the fact that $\mathsf{P}_{M|XQ}(m|x,q) = 0$ when $\operatorname{len}(m) > \operatorname{len}(x)$ to reduce the dimension of $\boldsymbol{z}_x$ from $\mathcal{M}$ to $\mathcal{M}_x$. □

Remark 1. An interesting phenomenon here that does not occur in the query-agnostic setting is the comparison between the average and conditional distortion-rate functions, i.e., $\bar{D}^*(R)$ and $D_q^*(R)$ for $q \in \mathcal{Q}$. One possible way to "average" the conditional distortion-rate functions would be to simply compute $\mathbb{E}_{Q\sim\mathsf{P}_Q}\left[D_Q^*(R)\right]$, but we always have $\bar{D}^*(R) \le \mathbb{E}_{Q\sim\mathsf{P}_Q}\left[D_Q^*(R)\right]$. This is because the latter averages the distortion-rate functions over P_Q at a fixed value of the rate, i.e., the prompt for each query is forced to be compressed to the same rate R. For $\bar{D}^*(R)$, on the other hand, only the *average* rate over the queries is required to be R. This allows the compressor to set a higher rate for "difficult queries" that have higher distortion values, and use a lower rate for queries that have lower distortion values in general. This is exactly the phenomenon we exploit in designing the *variable-rate* compression scheme LLMLingua-2 Dynamic in Sec. 3 and App. F, which outperforms other existing schemes in our experiments.

Just as in the query-agnostic setting, it is useful to compute and solve the dual linear programs instead of directly solving the linear programs above.

Theorem 2 (Query-aware dual LPs)**.** *The (average) distortion-rate function for query-aware prompt compression* (3) *is given by the solution to the dual of the linear program* (avg-cond-LP)*, i.e.,*

$$\bar{D}^*(R) = \sup_{\lambda \geq 0} \left\{ -\lambda R + \sum_{x \in \mathcal{X}, q \in \mathcal{Q}} \min_{m \in \mathcal{M}_x} \left[\bar{\boldsymbol{D}}^q_{x,m} + \lambda \bar{\boldsymbol{R}}^q_{x,m} \right] \right\}. \tag{avg-cond-dual-LP}$$

The (conditional) distortion-rate function for query-aware prompt compression (4) *is given by the solution to the dual of the linear program* (cond-LP)*, i.e.,*

$$D^*_q(R) = \sup_{\lambda \geq 0} \left\{ -\lambda R + \sum_{x \in \mathcal{X}} \min_{m \in \mathcal{M}_x} \left[\boldsymbol{D}^q_{x,m} + \lambda \boldsymbol{R}^q_{x,m} \right] \right\}. \tag{cond-dual-LP}$$

Proof. (Conditional) This follows trivially by simply observing that the linear program in (cond-LP) is identical to that in (LP), except that $\boldsymbol{D}_x$ and $\boldsymbol{R}_x$ are replaced by the (conditional) query-aware versions $\boldsymbol{D}^q_x$ and $\boldsymbol{R}^q_x$ respectively. Henc, by Thm. 1 the solution to (cond-LP) is given by (dual-LP) with $\boldsymbol{D}_x$ and $\boldsymbol{R}_x$ replaced by the (conditional) query-aware versions $\boldsymbol{D}^q_x$ and $\boldsymbol{R}^q_x$ respectively, which gives (cond-dual-LP).

(Average) In addition to replacing $\boldsymbol{D}_x$ and $\boldsymbol{R}_x$ from (LP) by the (average) query-aware versions $\bar{\boldsymbol{D}}^q_x$ and $\bar{\boldsymbol{R}}^q_x$ respectively, we also have that the optimization variables are given by $z_{x,q}$ for each pair $(x, q) \in \mathcal{X} \times \mathcal{Q}$ as opposed to simply z_x for each $x \in \mathcal{X}$. Hence, by Thm. 1 the solution to (avg-cond-LP) is given by (dual-LP) with $\boldsymbol{D}_x$ and $\boldsymbol{R}_x$ are replaced by the (average) query-aware versions $\bar{\boldsymbol{D}}^q_x$ and $\bar{\boldsymbol{R}}^q_x$ respectively and $\mathcal{X}$ replaced by $\mathcal{X} \times \mathcal{Q}$. This gives (avg-cond-dual-LP) exactly, and we are done. □

Note that both (avg-cond-dual-LP) and (cond-dual-LP) are of the same form as (dual-LP), with some minor differences. For a given $q \in \mathcal{Q}$, the conditional distortion-rate function $D^*_q(R)$ is identical to the query-unaware distortion-rate function $D^*_q(R)$ with $(\boldsymbol{D}_x, \boldsymbol{R}_x)$ replaced by $(\boldsymbol{D}^q_x, \boldsymbol{R}^q_x)$, and hence can be solved by running Alg. 1 with the input $\left\{R, (\boldsymbol{D}^q_x)_{x \in \mathcal{X}}, (\boldsymbol{R}^q_x)_{x \in \mathcal{X}}\right\}$. For the average distortion-rate function $\bar{D}^*(R)$, in addition to replacing $\boldsymbol{D}_x$ and $\boldsymbol{R}_x$ by $\bar{\boldsymbol{D}}^q_x$ and $\bar{\boldsymbol{R}}^q_x$, we also have that $\mathcal{X}$ is replaced by $\mathcal{X} \times \mathcal{Q}$, hence $\bar{D}^*(R)$ is obtained by running 1 with the input $\left\{R, (\boldsymbol{D}^q_{x'})_{x' \in \mathcal{X}'}, (\boldsymbol{R}^q_{x'})_{x' \in \mathcal{X}'}\right\}$, where $\mathcal{X}' \triangleq \mathcal{X} \times \mathcal{Q}$ and x' runs over all pairs (x, q).

D. The dual linear program: proof and solution

D.1. Derivation of the dual linear program

Proof of Thm. 1. We start from the linear program (LP) and construct its dual. Recall that (LP) is given by

$$\begin{aligned} D^*(R) = \inf_{\left(\boldsymbol{z}_x \in \mathbb{R}^{\mathcal{M}_x}_+\right)_{x \in \mathcal{X}}} \quad & \sum_{x \in \mathcal{X}} \boldsymbol{D}^\top_x \boldsymbol{z}_x \\ \text{s.t.} \quad & \sum_{x \in \mathcal{X}} \boldsymbol{R}^\top_x \boldsymbol{z}_x \leq R, \\ & \mathbf{1}^\top \boldsymbol{z}_x = 1, \quad \forall x \in \mathcal{X}. \end{aligned}$$

Introduce the Lagrange multipliers $\lambda \geq 0$ to handle the inequality constraint and $\mu_x \in \mathbb{R}$ for each $x \in \mathcal{X}$ to handle the equality constraints. Then, the above equation is equivalent to

$$D^*(R) = \inf_{\left(\boldsymbol{z}_x \in \mathbb{R}^{\mathcal{M}_x}_+\right)_{x \in \mathcal{X}}} \left\{ \sum_{x \in \mathcal{X}} \boldsymbol{D}^\top_x \boldsymbol{z}_x + \sup_{\lambda \geq 0} \lambda \left(\sum_{x \in \mathcal{X}} \boldsymbol{R}^\top_x \boldsymbol{z}_x - R \right) + \sum_{x \in \mathcal{X}} \left[\sup_{\mu_x \in \mathbb{R}} \mu_x \left(\mathbf{1}^\top \boldsymbol{z}_x - 1 \right) \right] \right\}.$$

To see why this equivalence holds, observe that the terms $\sup_{\lambda \geq 0} \lambda \left(\sum_{x \in \mathcal{X}} \boldsymbol{R}^\top_x \boldsymbol{z}_x - R \right)$ and $\sup_{\mu_x \in \mathbb{R}} \mu_x \left(\mathbf{1}^\top \boldsymbol{z}_x - 1 \right)$ are both 0 when $(\boldsymbol{z}_x)_{x \in \mathcal{X}}$ is in the feasible set of (LP) and $+\infty$ otherwise. Let $\mu \triangleq (\mu_x)_{x \in \mathcal{X}} \in \mathbb{R}^{\mathcal{X}}$, then we can simplify the above expression by rearranging terms, to obtain

$$D^*(R) = \inf_{\left(\boldsymbol{z}_x \in \mathbb{R}_+^{\mathcal{M}_x}\right)_{x\in\mathcal{X}}} \sup_{\substack{\mu\in\mathbb{R}^{\mathcal{X}},\\ \lambda\geq 0}} \left\{\sum_{x\in\mathcal{X}} \left(\boldsymbol{D}_x + \lambda \boldsymbol{R}_x + \mu_x \mathbf{1}\right)^\top \boldsymbol{z}_x - \lambda R - \sum_{x\in\mathcal{X}} \mu_x\right\}.$$

Note that the objective $\sum_{x\in\mathcal{X}} \left(\boldsymbol{D}_x + \lambda \boldsymbol{R}_x + \mu_x \mathbf{1}\right)^\top \boldsymbol{z}_x - \lambda R - \sum_{x\in\mathcal{X}} \mu_x$ is linear in $(\boldsymbol{z}_x)_{x\in\mathcal{X}}$ and in (μ, λ), and the minimization and maximization are both over convex sets. Hence, by the minmax theorem (von Neumann, 1928; Simons, 1995), we can switch their order without affecting the equality, i.e.,

$$D^*(R) = \sup_{\substack{\mu\in\mathbb{R}^{\mathcal{X}},\\ \lambda\geq 0}} \inf_{\left(\boldsymbol{z}_x \in \mathbb{R}_+^{\mathcal{M}_x}\right)_{x\in\mathcal{X}}} \left\{\sum_{x\in\mathcal{X}} \left(\boldsymbol{D}_x + \lambda \boldsymbol{R}_x + \mu_x \mathbf{1}\right)^\top \boldsymbol{z}_x - \lambda R - \sum_{x\in\mathcal{X}} \mu_x\right\}.$$

If, for some x, there is a component of the vector $\boldsymbol{D}_x + \lambda \boldsymbol{R}_x + \mu_x \mathbf{1} \in \mathbb{R}^{\mathcal{M}_x}$ that is negative, then letting that component of $\boldsymbol{z}_x$ go to infinity, we have that the inner infimum is $-\infty$. On the other hand, if every component of $\boldsymbol{D}_x + \lambda \boldsymbol{R}_x + \mu_x \mathbf{1}$ is nonnegative for every x, then the infimum is simply 0, attained by setting $\boldsymbol{z}_x = \mathbf{0}$. Hence, the above equation reduces to

$$\begin{aligned} D^*(R) = \quad & \sup_{\substack{\mu\in\mathbb{R}^{\mathcal{X}},\\ \lambda\geq 0}} \quad -\lambda R - \sum_{x\in\mathcal{X}} \mu_x \\ & \text{s.t.} \qquad \boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m} + \mu_x \geq 0 \quad \text{ for every } m \in \mathcal{M}_x \text{ and } x \in \mathcal{X}. \end{aligned}$$

For a given x, the constraint $\boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m} + \mu_x \geq 0$ for all $m \in \mathcal{M}_x$ is equivalent to $-\mu_x \leq \min_{m\in\mathcal{M}_x} \left(\boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m}\right)$. Letting $\nu_x \triangleq \min_{m\in\mathcal{M}_x} \left(\boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m}\right) + \mu_x$ and $\nu \triangleq (\nu_x)_{x\in\mathcal{X}}$, the constraint is simply that $\nu_x \geq 0$ for all x, or equivalently, $\nu \in \mathbb{R}_+^{\mathcal{X}}$. Hence, the above equation can be written as

$$D^*(R) = \sup_{\substack{\nu\in\mathbb{R}_+^{\mathcal{X}},\\ \lambda\geq 0}} \; -\lambda R + \sum_{x\in\mathcal{X}} \min_{m\in\mathcal{M}_x} \left(\boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m}\right) - \sum_{x\in\mathcal{X}} \nu_x.$$

Observe that only the first two terms depend on λ, and only the last term depends on ν. This lets us optimize over λ and ν separately, to give

$$\begin{aligned} D^*(R) &= \sup_{\lambda\geq 0} \left\{-\lambda R + \sum_{x\in\mathcal{X}} \min_{m\in\mathcal{M}_x} \left(\boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m}\right)\right\} + \sup_{\nu\in\mathbb{R}_+^{\mathcal{X}}} \left(-\sum_{x\in\mathcal{X}} \nu_x\right) \\ &= \sup_{\lambda\geq 0} \left\{-\lambda R + \sum_{x\in\mathcal{X}} \min_{m\in\mathcal{M}_x} \left(\boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m}\right)\right\}, \end{aligned}$$

since $\sup_{\nu\in\mathbb{R}_+^{\mathcal{X}}} \left(-\sum_{x\in\mathcal{X}} \nu_x\right) = -\inf_{\nu\in\mathbb{R}_+^{\mathcal{X}}} \sum_{x\in\mathcal{X}} \nu_x = -\sum_{x\in\mathcal{X}} \inf_{\nu_x\geq 0} \nu_x = 0$, and we are done. □

D.2. Proof and illustration of Alg. 1

In this section, we explain each step of Alg. 1 in detail. In doing so, we prove that the algorithm does indeed solve (dual-LP), i.e., computes

$$D^*(R) = \sup_{\lambda\geq 0} \left\{-\lambda R + \sum_{x\in\mathcal{X}} \min_{m\in\mathcal{M}_x} \left[\boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m}\right]\right\}.$$

We also use an artificial example as described below to show the working of the algorithm, in particular lines 1–9. For convenience, the algorithm is repeated verbatim below, without comments:

Consider the following artificial example with $\mathcal{X} = \{\alpha, \beta\}$. Let $(\boldsymbol{R}_\alpha, \boldsymbol{D}_\alpha)$ and $(\boldsymbol{R}_\beta, \boldsymbol{D}_\beta)$ be as given by the blue points in the scatter plots over $m \in \mathcal{M}_\alpha$ and $m \in \mathcal{M}_\beta$ respectively in Fig. 3. In our example, we have $|\mathcal{M}_\alpha| = 11$ and $|\mathcal{M}_\beta| = 8$. The following observation is crucial: recall the definitions of $\boldsymbol{R}_x$ and $\boldsymbol{D}_x$,

$$\boldsymbol{D}_{x,m} \triangleq \mathsf{P}_X(x)\, \mathbb{E}\left[\mathsf{d}_{\log}(Y, \phi_{\mathsf{LLM}}(m, Q))\right] \quad \text{and} \quad \boldsymbol{R}_{x,m} \triangleq \mathsf{P}_X(x)\, \frac{\operatorname{len}(m)}{\operatorname{len}(x)}, \qquad m \in \mathcal{M}_x,$$

Algorithm 1: To compute $D^*(R)$ via (dual-LP)

```
Input: $R$, $(\boldsymbol{D}_x)_{x\in\mathcal{X}}$, $(\boldsymbol{R}_x)_{x\in\mathcal{X}}$
Output: $D^*(R)$, the distortion-rate function at rate $R$
1  for $x \in \mathcal{X}$ do
2      Find $\mathcal{M}_{\text{env}}^{(x)} \subseteq \mathcal{M}_x : \{(\boldsymbol{R}_{x,m}, \boldsymbol{D}_{x,m})\}_{m\in\mathcal{M}_{\text{env}}^{(x)}}$ are on the lower-left concave boundary of $\{(\boldsymbol{R}_{x,m}, \boldsymbol{D}_{x,m})\}_{m\in\mathcal{M}_x}$ ;
3      $\left\{m_1^{(x)}, m_2^{(x)}, \ldots, m_{k_x}^{(x)}\right\} \leftarrow \mathcal{M}_{\text{env}}^{(x)}$ ordered such that $\boldsymbol{R}_{x,m_{k_x}^{(x)}} > \cdots > \boldsymbol{R}_{x,m_1^{(x)}}$;
4      for $i = 1, \ldots, k_x - 1$ do $\lambda_i^{(x)} \leftarrow \frac{\boldsymbol{D}_{x,m_i^{(x)}} - \boldsymbol{D}_{x,m_{i+1}^{(x)}}}{\boldsymbol{R}_{x,m_{i+1}^{(x)}} - \boldsymbol{R}_{x,m_i^{(x)}}}$;
5      $\lambda_0^{(x)} \leftarrow +\infty$; $\lambda_{k_x}^{(x)} \leftarrow 0$;
6      $\Lambda^{(x)} \leftarrow \left\{\lambda_0^{(x)}, \lambda_1^{(x)}, \ldots, \lambda_{k_x-1}^{(x)}, \lambda_{k_x}^{(x)}\right\}$;
7  $\left\{\widetilde{\lambda}_0, \ldots, \widetilde{\lambda}_k\right\} \leftarrow \bigcup_{x\in\mathcal{X}} \Lambda^{(x)}$ with $+\infty = \widetilde{\lambda}_0 > \widetilde{\lambda}_1 > \cdots > \widetilde{\lambda}_{k-1} > \widetilde{\lambda}_k = 0$;
8  for $x \in \mathcal{X}$ do
9      for $j = 1, \ldots, k$ do
10         Set $\widetilde{m}_j^{(x)} \leftarrow m_i^{(x)}$ where $i \in \{1, \ldots, k_x\} : \left(\lambda_i^{(x)}, \lambda_{i-1}^{(x)}\right) \supseteq \left(\widetilde{\lambda}_j, \widetilde{\lambda}_{j-1}\right)$
11 for $j = 1, \ldots, k$ do
12     if $\sum_{x\in\mathcal{X}} \boldsymbol{R}_{x,\widetilde{m}_j^{(x)}} > R$ then $\lambda_j \leftarrow \widetilde{\lambda}_{j-1}$ else $\lambda_j \leftarrow \widetilde{\lambda}_j$ ;
13     $D_j \leftarrow -\lambda_j R + \sum_{x\in\mathcal{X}} \left[\boldsymbol{D}_{x,\widetilde{m}_j^{(x)}} + \lambda_j \boldsymbol{R}_{x,\widetilde{m}_j^{(x)}}\right]$;
14 Return $\max_{j=1,\ldots,k} D_j$;
```

For a fixed value of x, the positive real numbers $\boldsymbol{D}_{x,m}$ can be arbitrary, but $\boldsymbol{R}_{x,m}$ must be an integral multiple of the constant $\frac{\mathsf{P}_X(x)}{\text{len}(x)}$. Hence, for a given x, $\boldsymbol{R}_{x,m}$ takes at most $\text{len}(x)$ possible values. This turns out to be extremely beneficial in the first step, namely identifying the points on the lower-left concave boundary.

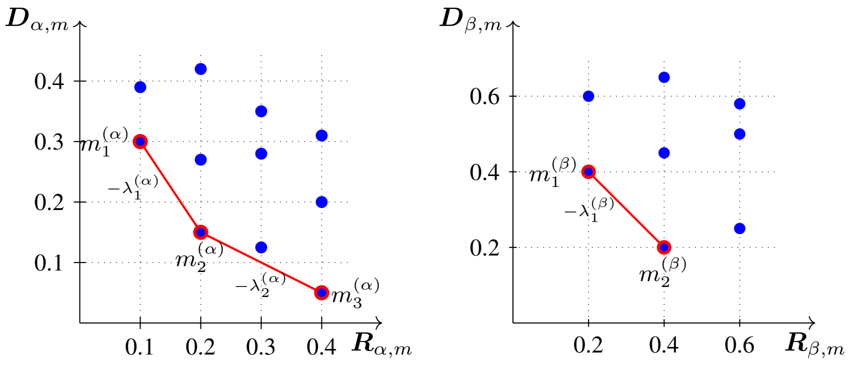


Figure 3: Scatter plots showing the points $\{(\boldsymbol{R}_{\alpha,m}, \boldsymbol{D}_{\alpha,m})\}_{m\in\mathcal{M}_\alpha}$ and $\{(\boldsymbol{R}_{\beta,m}, \boldsymbol{D}_{\beta,m})\}_{m\in\mathcal{M}_\beta}$ in blue. The associated lower-left concave boundaries $\mathcal{M}_{\text{bd}}^{(\alpha)} = \{m_1^{(\alpha)}, m_2^{(\alpha)}, m_3^{(\alpha)}\}$ and $\mathcal{M}_{\text{bd}}^{(\beta)} = \{m_1^{(\beta)}, m_2^{(\beta)}\}$ are in red; $\lambda_1^{(\alpha)}$, $\lambda_2^{(\alpha)}$ and $\lambda_1^{(\beta)}$ are the magnitudes of the slopes of the associated line segments.

For $x \in \mathcal{X}$, lines 2–3 of the algorithm identify the k_x points $m_1^{(x)}, \ldots, m_{k_x}^{(x)}$ that lie on the lower-left concave boundary of $\{(\boldsymbol{R}_{x,m}, \boldsymbol{D}_{x,m})\}_{m\in\mathcal{M}_x}$. The lower-left concave boundaries are given by the red lines and the points lying on the boundary are outlined in red. Observe that $k_\alpha = 3$ and $k_\beta = 2$. The quantities computed in line 4 are simply the magnitudes of the slopes of the line segments on the boundary. A simple computation gives the result of line 6 of the algorithm in our example to be $\Lambda^{(\alpha)} = \{+\infty, 1.5, 0.5, 0\}$ and $\Lambda^{(\beta)} = \{+\infty, 1, 0\}$. Clearly, for a given value of $x \in \mathcal{X}$ and $\lambda \in \left[\lambda_j^{(x)}, \lambda_{j-1}^{(x)}\right)$, we have that $m_i^{(x)}$ minimizes $\boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m}$ over all $m \in \mathcal{M}_x$, by virtue of the fact that these points come from the lower-left concave boundary. Hence, for $\lambda \in \left[\lambda_j^{(x)}, \lambda_{j-1}^{(x)}\right)$, we have

$$\sum_{x\in\mathcal{X}} \min_{m\in\mathcal{M}_x} [\boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m}] = \sum_{x\in\mathcal{X}} \left[\boldsymbol{D}_{x,m_j^{(x)}} + \lambda \boldsymbol{R}_{x,m_j^{(x)}}\right].$$

We cannot simplify this further in its current state since λ in the above expression depends on x. Hence, we must remove the dependence of the range $\left[\lambda_j^{(x)}, \lambda_{j-1}^{(x)}\right)$ on x. To do so, observe that each $\Lambda^{(x)}$ is a partition of $\mathbb{R}_+$ on which $m_i^{(x)}$ is the minimizer of $\boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m}$. Line 7 of the algorithm simply constructs the union of all these partitions, with k elements, denoted by the $\widetilde{\lambda}$ variables; here we have $k = 4$ and the union is $\{+\infty, 1.5, 1, 0.5, 0\}$. For each x, the minimizer on each interval $\left[\widetilde{\lambda}_j, \widetilde{\lambda}_{j-1}\right)$ of the finer partition is known exactly to be one of the $m_i^{(x)}$'s; lines 8–10 associate to each interval the corresponding minimizer, given by $\widetilde{m}_j^{(x)}$. There is no computation involved in these steps, only notational rewriting. The corresponding values obtained for our example are given in the table below (with $\widetilde{\lambda}_0 = +\infty$); observe that $\widetilde{m}_j^{(x)}$ minimizes $\boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m}$ over $m \in \mathcal{M}_x$ for $\lambda \in \left[\widetilde{\lambda}_j, \widetilde{\lambda}_{j-1}\right)$.

Table 1: The outputs produced by lines 7–10 of Alg. 1 with $(\boldsymbol{R}_\alpha, \boldsymbol{D}_\alpha)$ and $(\boldsymbol{R}_\beta, \boldsymbol{D}_\beta)$ as given in Fig. 3.

j	$\widetilde{\lambda}_j$	$\widetilde{m}_j^{(\alpha)}$	$\widetilde{m}_j^{(\beta)}$
1	1.5	$m_1^{(\alpha)}$	$m_1^{(\beta)}$
2	1	$m_2^{(\alpha)}$	$m_1^{(\beta)}$
3	0.5	$m_2^{(\alpha)}$	$m_2^{(\beta)}$
4	0	$m_3^{(\alpha)}$	$m_2^{(\beta)}$

At this point, we have for $\lambda \in \left[\widetilde{\lambda}_j, \widetilde{\lambda}_{j-1}\right)$,

$$\begin{aligned}\sum_{x\in\mathcal{X}} \min_{m\in\mathcal{M}_x} [\boldsymbol{D}_{x,m} + \lambda \boldsymbol{R}_{x,m}] &= \sum_{x\in\mathcal{X}} \left[\boldsymbol{D}_{x,\widetilde{m}_j^{(x)}} + \lambda \boldsymbol{R}_{x,\widetilde{m}_j^{(x)}}\right] \\ &= \left(\textstyle\sum_{x\in\mathcal{X}} \boldsymbol{D}_{x,\widetilde{m}_j^{(x)}}\right) + \lambda \left(\textstyle\sum_{x\in\mathcal{X}} \boldsymbol{R}_{x,\widetilde{m}_j^{(x)}}\right).\end{aligned}$$

Hence, the right-hand side of (dual-LP) is simply

$$\begin{aligned}&\max_{j=1,\ldots,k} \sup_{\lambda\in[\widetilde{\lambda}_j,\widetilde{\lambda}_{j-1})} \left\{\left(\textstyle\sum_{x\in\mathcal{X}} \boldsymbol{D}_{x,\widetilde{m}_j^{(x)}}\right) + \lambda \left(\textstyle\sum_{x\in\mathcal{X}} \boldsymbol{R}_{x,\widetilde{m}_j^{(x)}} - R\right)\right\} \\ &= \max_{j=1,\ldots,k} \left\{\left(\textstyle\sum_{x\in\mathcal{X}} \boldsymbol{D}_{x,\widetilde{m}_j^{(x)}}\right) + \sup_{\lambda\in[\widetilde{\lambda}_j,\widetilde{\lambda}_{j-1})} \lambda \left(\textstyle\sum_{x\in\mathcal{X}} \boldsymbol{R}_{x,\widetilde{m}_j^{(x)}} - R\right)\right\},\end{aligned}$$

where the first equality follows since $\left\{\widetilde{\lambda}_j\right\}_{j=0}^k$ is a partition of $\mathbb{R}_+$. Consider the term $\sup_{\lambda\in[\widetilde{\lambda}_j,\widetilde{\lambda}_{j-1})} \lambda \left(\sum_{x\in\mathcal{X}} \boldsymbol{R}_{x,\widetilde{m}_j^{(x)}} - R\right)$. If $\boldsymbol{R}_{x,\widetilde{m}_j^{(x)}} > R$, this supremum occurs in the limit as $\lambda \to \widetilde{\lambda}_{j-1}$, otherwise it is achieved at $\lambda = \widetilde{\lambda}_j$. Hence, defining λ_j to be $\widetilde{\lambda}_{j-1}$ or $\widetilde{\lambda}_j$ accordingly as in line 12, we have that the above expression is simply $\max_{j=1,\ldots,k} \left\{\left(\sum_{x\in\mathcal{X}} \boldsymbol{D}_{x,\widetilde{m}_j^{(x)}}\right) + \lambda_j \left(\sum_{x\in\mathcal{X}} \boldsymbol{R}_{x,\widetilde{m}_j^{(x)}} - R\right)\right\}$, which is exactly what line 14 returns. Hence, we have that the algorithm correctly computes the distortion-rate function $D^*(R)$.

E. Connections to information theory literature

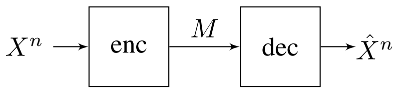
Rate-distortion theory is an area of information theory introduced by Shannon (1959) to study the fundamental limits of source compression. The simplest rate-distortion setup is shown in Fig. 4a: We are given a source which generates samples $X_1, \ldots, X_n$ independently and identically distributed (i.i.d.) according to the distribution P_X on the set $\mathcal{X}$. We are also given a reconstruction alphabet $\hat{\mathcal{X}}$, which may or may not be equal to $\mathcal{X}$. The goal is to compress X^n to a sequence of k elements from an alphabet $\mathcal{V}$, such that a reconstruction onto $\hat{\mathcal{X}}^n$ is as "faithful" as possible, while keeping k as small as possible (in the information theory literature, $\mathcal{V} = \{0,1\}$ typically). The fidelity of representation is quantified by a *distortion function* $\mathsf{d} : \mathcal{X} \times \hat{\mathcal{X}} \to [0, \infty]$. For example, the problem of compressing images with p real-valued pixels into bit sequences can be cast in this formulation by taking $\mathcal{X} = \hat{\mathcal{X}} = \mathbb{R}^p$, $\mathcal{V} = \{0,1\}$, and the squared-loss distortion function $\mathsf{d}(x, \hat{x}) = \|x - \hat{x}\|_2^2$.

Formally, the goal is to construct an encoder $\mathrm{enc} : \mathcal{X}^n \to \mathcal{V}^k$ and a decoder $\mathrm{dec} : \mathcal{V}^k \to \hat{\mathcal{X}}^n$ such that: (1) the *rate* k/n, and (2) the *(average) distortion* $\mathbb{E}\left[\mathsf{d}(X^n, \mathrm{dec}(\mathrm{enc}(X^n))\right]$, are both as small as possible. We say that the rate-distortion pair (R, D) is *achievable* for the source P_X under the distortion function d if there exists an $(\mathrm{enc}, \mathrm{dec})$ pair with rate at most R and average distortion at most D. If the pair (R, D) is achievable, then clearly, for $\widetilde{R} \geq R$ and $\widetilde{D} \geq D$, the pair $(\widetilde{R}, \widetilde{D})$ is also achievable. Thus, the quantity of interest to us is the lower boundary of the set of achievable (R, D) pairs. This is given by the *distortion-rate function* D^*, which is defined as follows: the distortion-rate function at rate R is the smallest distortion D such that the pair (R, D) is achievable, or equivalently,

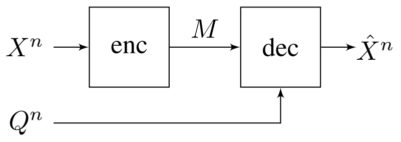
$$\begin{aligned} D^*(R) &\triangleq \inf\{D \geq 0 \mid (R, D) \text{ is achievable}\} \\ &= \inf\{D \geq 0 \mid \text{there exists } (\mathrm{enc}, \mathrm{dec}) \text{ with rate} \leq R \text{ and distortion} \leq D\}. \end{aligned} \tag{7}$$

Note that the distortion-rate function depends on the source P_X and the choice of distortion measure. Closed form expressions are known in some cases, the reader is encouraged to refer to classical texts on information theory (Berger, 1971; Cover & Thomas, 2006; El Gamal & Kim, 2011) for examples. It is important to note that the distortion-rate function is a fundamental limit; no choice of encoder and decoder can give a lower rate and a lower distortion. Thus, the distortion-rate function characterizes the Pareto-optimal front of the trade-off between rate and distortion. It is more common in the information theory literature to define the *rate-distortion function* $R^*(D)$, which is the smallest rate R such that the pair (R, D) is achievable. The two functions trace the same curve when plotted on the same two-dimensional plane.

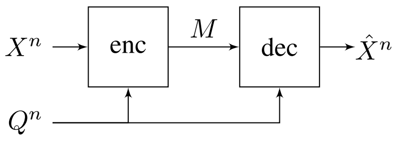
Several variants of this problem can be defined by introducing the notion of *side-information*, where we have i.i.d. samples $(X_1, Q_1), \ldots, (X_n, Q_n)$ of a pair of correlated random variables $(X, Q) \sim \mathsf{P}_{XQ} \in \mathscr{P}(\mathcal{X} \times \mathcal{Q})$. A natural question to ask is what improvement is possible in terms of the rate-distortion trade-off for X^n when either the encoder or decoder or both have access to this side-information Q^n, which is correlated with X. If only the encoder has access to Q^n, then no improvement can be obtained. If both the encoder and decoder have access to Q^n as shown in Fig. 4c and studied by Gray (Gray, 1972), then, clearly, an improvement is possible. Surprisingly, we can obtain nontrivial improvements when the decoder has access to Q^n as shown in Fig. 4b and studied by Wyner and Ziv (Wyner & Ziv, 1976; Wyner, 1978).



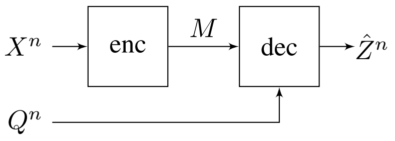
(a) No side-information (Shannon, 1959)



(b) Side-information at only the decoder (Wyner & Ziv, 1976; Wyner, 1978)



(c) Side-information at the encoder and the decoder (Weissman & El Gamal, 2006)



(d) For function computation, $Z = f(X, Q)$ (Yamamoto, 1982)

Figure 4: Rate-distortion models of compression.

These models resemble our setups for query-aware and query-agnostic prompt compression respectively, with a key difference being that the decoder in our problem is the pretrained LLM, which is fixed. An rate-distortion setup that is closer to our problem in this sense is that of compression for function computation, introduced by (Yamamoto, 1982). Here, the goal is to recover an estimate $\hat{Z}^n$ that is close to Z^n, with $Z_i = f(X_i, Q_i)$ for some desired function f. At first glance, it might appear that this setup is exactly our model for prompt compression, but this turns out to be false — the desired output is an estimate of $f(X, Q)$, but the decoder can be designed to compute *any arbitrary function* of M and Q. In prompt compression, we have the constraint that the function computed by the decoder is fixed to be ϕ_{LLM}, in addition to requiring that the output be close to some function of X and Q. Thus, our model for prompt compression actually corresponds to a rate-distortion problem for function computation with side-information *with a fixed decoder*, which has not been studied before, to the best of our knowledge. The distortion-rate function $D^*(R)$ for this setup is given by (LP) and (dual-LP). A closed form solution $D^*(R)$ requires further assumptions on ϕ_{LLM}; nonetheless, $D^*(R)$ can be computed by solving Alg. 1.

F. Experimental setup

As described in Sec. 3, we provide a detailed explanation of our experimental setup in this section. In all our results, we compare the performance of existing prompt compression methods (that are compatible with the black-box model setting we consider in this work) with the optimal curve for a synthetic dataset. We report our results with the log loss and 0/1 loss metrics, which are defined respectively as

$$\mathsf{d}_{\log}(y, \mathsf{P}_{\hat{Y}}) = \log \frac{1}{\mathsf{P}_{\hat{Y}}(y)} \quad \text{and} \quad \mathsf{d}_{0/1}(y, \mathsf{P}_{\hat{Y}}) = \mathbb{1}\Big\{ y \neq \operatorname*{argmax}_{\hat{y}} \mathsf{P}_{\hat{Y}}(\hat{y}) \Big\}. \tag{8}$$

We include an ablation study on the impact of tokenization of the prompt compression problem, as tokenization is lossy since it groups together multiple symbols into a single symbol before passing it to an LLM. We study the effect of tokenization on the prompt compression problem by forcing the tokenizer on the encoder and decoder side to tokenize the bits of the binary string prompts in our dataset individually, which we refer to as "forced tokenization." We run experiments in this setting and with the regular "standard tokenization." Technical details regarding the system parameters for these experiments can be found in App. H.

Baseline methods. We compare the rate-distortion trade-off of the optimal strategy (both query-aware and query-agnostic) with prompt compression methods that can be used to compress prompts for a black-box target LLM: Selective Context (Li et al., 2023), LLMLingua (Jiang et al., 2023c), LLMLingua Query (Jiang et al., 2023b), LLMLingua-2 (Pan et al., 2024). As such, we do not consider methods like Gist Tokens (Mu et al., 2023), In-Context Autoencoder (Ge et al., 2024), and AutoCompressor (Chevalier et al., 2023) since they require special training methods generally not compatible with black-box target LLMs. Selective Context uses $-\log P(x_i \mid x_0, x_1, \ldots, x_{i-1})$ to score the i-th token, and retains the tokens whose score is larger than the p-percentile, where $p \in [0, 1]$ is the ratio parameter. LLMLingua uses a similar method, but they first partition the input prompt into segments and condition on previously compressed segments to compress the current segment. They later extended their method to perform query-aware compression, which is what we use for LLMLingua Query. While these methods use a decoder-style (causal) transformer LLM to do prompt compression, this approach makes an independence assumption on the influence of future tokens have on the i-th token. LLMLingua-2 instead uses an encoder-style (bidirectional) LLM to perform a token classification task, where their model predicts whether a given token should be kept or removed.

Our proposed methods. We add two novel contributions over the LLMLingua-2 work: (1) we adapt LLMLingua-2 to the query-aware setting, whereas the original work only proposed the query-agnostic approach, which we call "LLMLingua-2 Query," and (2) we further adapt this query-aware approach into a variable-rate approach we refer to as "LLMLingua-2 Dynamic." This approach which lets the encoder model decide which tokens to keep based on the confidence over a specified threshold. In other words, LLMLingua-2 and LLMLingua-2 Query accept a rate parameter to determine the compression ratio, but LLMLingua-2 Dynamic replaces the rate parameter with a threshold parameter. As a result, the encoder model predicts the probability of keeping a particular token, and the token is kept if the predicted probability is above a given threshold, resulting in a variable-rate compression of the prompt. This is an important aspect of the prompt compression problem, as some prompts are more compressible than others, and vice versa.

Models. We use Mistral 7B Instruct v0.2 (Jiang et al., 2023a) as our black-box target LLM (decoder side), which is fine-tuned on the training set partition of our synthetic dataset. Critically, this model is fixed after fine-tuning and no prompt compression methods have access to any part of it. All prompt compression methods use an LLM as part of their compression algorithm (encoder side); we use deduplicated Pythia 1B (Biderman et al., 2023) for Selective Context, LLMLingua, and LLMLingua Query and RoBERTa Base (Liu et al., 2020) for LLMLingua-2 methods. For each method, we finetune on the training set partition to enable the best performance possible for that method. More information on how we trained these methods and the data we used is in App. H. For all models, including the target LLM, we fine-tune with LoRA (Hu et al., 2022) and conduct a hyperparameter grid search. We choose the configuration with the best performance on a test set that is different from the validation set. More details on the hyperparameter search are provided in App. H.2.

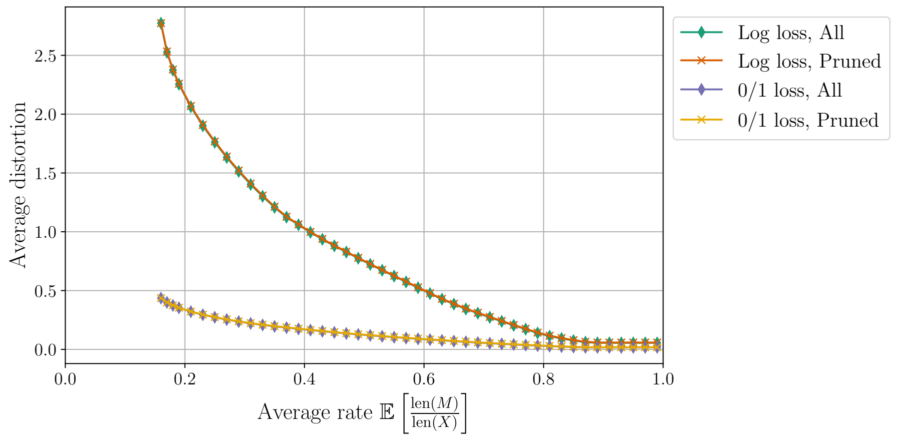


Figure 5: Query-agnostic distortion-rate curves plotted for log loss and 0/1 loss distortion measures. The curves marked with a 'diamond' are computed using all possible shorter sequences, while those marked with an '$\times$' are computed using only pruned versions of the original prompt. They are nearly identical, which suggests that a good approximation to the optimal distortion-rate curve can be obtained by considering pruned prompts only.

G. Limitations

Restriction to binary prompts. A major limitation of our paper is that the distortion-rate curves are computed using Alg. 1 for only binary prompts, and not natural language prompts. The decisive bottleneck in running Alg. 1 turns out to be obtaining $\boldsymbol{D}_x$ for each $x \in \mathcal{X}$, i.e., the input to the algorithm. We require one inference call for each possible compressed prompt $m \in \mathcal{M}_x$ to compute $\boldsymbol{D}_x$ for a particular prompt x and query. Taking $\mathcal{M}_x$ to be all sequences of length smaller than $\text{len}(x)$, we see that that the size of $\mathcal{M}_x$ is $\sum_{i=1}^{\text{len}(x)-1} |\mathcal{V}|^i$, where $\mathcal{V}$ is the vocabulary of the LLM. The model used in our experiments, Mistral 7B Instruct v0.2 (Jiang et al., 2023a), has $|\mathcal{V}| = 32{,}000$. Clearly, it is then virtually impossible to consider prompts with more than 2 tokens, and in fact, makes it difficult to consider even medium length prompts (50 tokens) for a vocabulary of size more than 2.

A key first step towards extending our algorithm for natural language prompts is the observation that all prompt compression methods in the literature work by *pruning* tokens, i.e., (1) they are non-generative, i.e., work by removing tokens from the original prompt and therefore do not generate any new tokens, and (2) they preserve the order of the tokens as they appear in the input sequence. Hence, to compute the fundamental limit for the schemes that compress the prompt by pruning, it is enough to consider $\mathcal{M}_x$ to be the sequences that are obtained from x by removing some number of (not necessarily contiguous) tokens. Then, we have $|\mathcal{M}_x| = 2^{\text{len}(x)}$, irrespective of the vocabulary size $|\mathcal{V}|$.

In Fig. 5, we observe that the distortion-rate curves obtained by restricting $\mathcal{M}_x$ to be only those sequences obtained from x from via pruning, are nearly identical to the original curves, where we take $\mathcal{M}_x$ to be all shorter sequences. This suggests two things: (1) there is no fundamental drawback to considering compression schemes are not generative, i.e., work by pruning the original prompt, and (2) we can approximate the optimal distortion-rate function reasonably well by considering only pruned versions of the prompt as possible compressed prompts. Thus, in principle, we can replicate experiments with natural language prompts of the same lengths (4 to 10) as the binary prompts in our experiments above, with the same computational cost. However, it is difficult to identify sufficiently rich natural language prompts of such short lengths for which compression is a reasonable problem, and hence, use binary prompts (generated from a Markov chain, to model the dependence between tokens) with natural language queries (since there is no computational restriction on the vocabulary of the queries) to run experiments such as those in Sec. 3 and App. I at scale.

LLMLingua Query seems to offer nearly no improvement over LLMLingua. While LLMLingua Query is a query-aware prompt compression model and should, in principle, perform better than LLMLingua, in all of our experimental results in Sec. 3 and App. I, we see that their performance is comparable at best. In order to confirm that this is simply a consequence of our synthetic dataset with binary prompts, we construct a synthetic dataset (see App. H.1 for details) with natural language prompts of length up to 15, so that we can also compute the optimal distortion-rate curves. This requires $2^{15} = 32{,}768$ inference calls per prompt and query. The advantage of Alg. 1 is that only requires these quantities to be

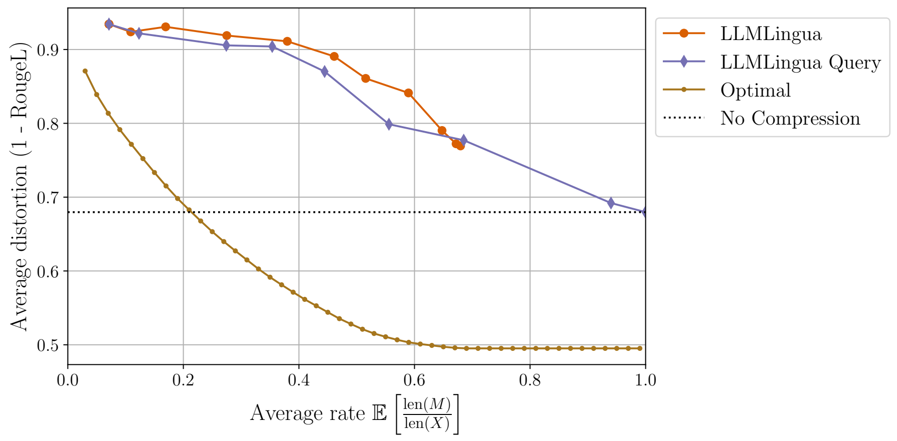


Figure 6: Query-agnostic distortion-rate function for the natural language dataset described in App. H.1 using the RougeL metric. Since a higher RougeL metric is better, we plot "1− the computed average distortion").

computed once per dataset, and the distortion-rate function can then be computed at any rate R with minimal overhead computational cost.

H. Experiment details

We provide additional details regarding our synthetic dataset and how we fine-tuned all models. Experiments were run on three different machines, two of which are identical machines with an AMD Ryzen Threadripper PRO 5975WX CPU (32 cores), 256 GB of system RAM, and 2x Nvidia RTX 4090 GPUs with 24 GB each. We also ran experiments on a DGX machine with an AMD EPYC 7742 64-Core Processor, 512 GB of system RAM, and 4x 80GB SMX4 A100 GPUs. LLM fine-tuning varies in length, depending on the model being fine-tuned. In general, it takes 10 to 30 minutes for a single fine-tuning run. Running the code necessary to reproduce all plots takes several hours.

We use code from the LLMLingua and Selective Context GitHub repos, which is released under the MIT license. In our experiments, we use the following models: Mistral 7B Instruct v0.2 (Apache-2.0), RoBERTa Base (MIT), and Pythia 1B deduped (Apache-2.0).

Each method requires a rate or threshold parameter r, for which we use $r \in \{0.04, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.96, 0.99, 1.0\}$ in our experiments. However, the length of the returned compressed prompt might not be faithful to this rate parameter, so, for each r, we report the average rate and average distortion on the examples in our validation dataset. LLMLingua and LLMLingua Query have one additional parameter controlling the size of each segment the prompt is broken into before compressing each segment. We found that using a segment size of 2 works best for our synthetic dataset.

H.1. Dataset

Solving for the optimal distortion-rate curve requires a substantial amount of compute as mentioned in App. G. To overcome this, we construct a synthetic dataset consisting of binary string prompts, natural language queries, and numerical and yes/no answers, for a total of 1400 examples (7 queries, 200 examples per query) in the validation set. We show a few examples of the validation partition of our synthetic dataset in Table 2. The binary prompts are generated according to a Markov chain with probability of switching 0.9 and probability of remaining in the same state 0.1.

We curated a small natural language dataset to generate results shown in Fig. 6 by prompting GPT-4 (OpenAI, 2024) to provide short natural language prompts of 15 tokens or less, provide four questions about each prompt, and give the answer. Afterward, we modified some of the questions and prompts slightly when the generated prompt by GPT-4 was too long or the questions and answers contained too much overlap with each other for a given prompt. In total, our dataset consists of ten prompts with four questions each. A few examples of our dataset are shown in Table 3.

Table 2: One example of each query from the validation set of our synthetic dataset

Prompt	Query	Answer
110011111	Count the number of 1s.	7
11111	Count the number of 0s.	0
00000111	Compute the parity.	1
11011111	What is the length of the longest subsequence of 0s or 1s?	5
0110	Is the binary string a palindrome?	Yes
1100111100	Count the number of transitions from 0 to 1 and 1 to 0.	3
111111	Predict the next bit.	1

Table 3: One example of each prompt from our natural language dataset.

Prompt	Query	Answer
After dinner, the cat chased a mouse around the house.	What was the cat doing?	The cat was chasing a mouse.
The dog barked loudly at the passing mailman on a quiet street.	Where did the barking occur?	On a quiet street.
After school, the child played with toys in the cozy living room.	When was the child playing?	After school.
At the art gallery, the artist painted a colorful mural on the wall.	Where was the painting done?	On the wall at the art gallery.

H.2. LLM fine-tuning

Given that our synthetic dataset of binary prompts is not naturally in the distribution of training data of LLMs, we use Mistral 7B Instruct v0.2 (Jiang et al., 2023a) as our black-box model, and fine-tune it on tuples of (prompt, query, answer). This is also known as “instruction fine-tuning;” we only compute the loss over the answer.

Each prompt compression method requires an LLM as part of its compression algorithm; we fine-tune Pythia 1B deduplicated (Biderman et al., 2023) for Selective Context and LLMLingua-based methods. Selective Context and LLMLingua only use negative log-likelihood scores over the prompt, so for these methods we fine-tune with the next-word prediction over the prompts. For LLMLingua Query, we place the (query, prompt, answer) tuple into context and then perform next token prediction over the entire context. We place only the query and prompt into the context for the prompt compression step (inference time).

LLMLingua-2 methods require an additional label set for every prompt as ground-truth answers to teach the model to predict which tokens should be kept. For our dataset, gathering the labels for each prompt is deterministic if the query is known, so it is easy to assemble the label set for query-aware LLMLingua-2 methods. For example, for the query “Is the binary string a palindrome?” we can easily choose the shortest sequence of tokens from the input that is also a palindrome (if the answer is “yes”) as the ground-truth compressed prompt. For LLMLingua-2 Query and LLMLingua-2 Dynamic, both of which are query-aware, we put the query and prompt into context and then train the LLM to predict which tokens to keep using the constructed label set. This process is less straightforward for query-agnostic LLMLingua-2 since it is not clear how to assign the labels without the query. In this case, we choose the ground-truth compressed prompt to consist of the highest entropy tokens. Given the Markov chain from which our prompts were generated, these tokens contain the transitions between bits. For all LLMLingua-2 methods, we fine-tune RoBERTa Base (Liu et al., 2020).

We conduct a grid search over a set of hyperparameters before fine-tuning the final model used for each method. Specifically, we use the training set to fine-tune a model with all combinations of hyperparameters, evaluate the final performance on each model with a test set, and choose the combination of hyperparameters leading to the best performance. We then merge the train and test set and train with the chosen hyperparameters and do a final evaluation on the validation, which is the dataset used in the results of this paper.

All models are searched over the same learning rate {5e-6, 1e-5, 5e-5, 1e-4}, batch size {16, 32}, LoRA rank {16 32 64 128}, and LoRA alpha {16 32 64 128} hyperparameters. For the number of training epochs, we search over {1, 2, 4} for Mistral 7B Instruct v0.2 and Pythia 1B deduplicated, and {8, 12} for RoBERTa Base.

We report our final set of hyperparameters used to fine-tune the LLM used for each prompt compression method in Table 4.

Table 4: Final set of hyperparameters used to train the LLM used in each prompt compression method.

Method	Tokenization	Epochs	Batch Size	Learning Rate	LoRA Rank	LoRA Alpha
Selective Context	Standard	1	16	5e-5	32	32
Selective Context	Forced	1	16	5e-5	128	64
LLMLingua	Standard	1	16	5e-5	32	32
LLMLingua	Forced	1	16	5e-5	128	64
LLMLingua Query	Standard	4	32	1e-4	128	128
LLMLingua Query	Forced	4	16	1e-4	64	128
LLMLingua-2	Standard	12	32	1e-4	128	128
LLMLingua-2	Forced	12	32	1e-4	64	128
LLMLingua-2 Query	Standard	12	32	1e-4	128	128
LLMLingua-2 Query	Forced	12	32	1e-4	64	128
LLMLingua-2 Dynamic (Ours)	Standard	12	32	1e-4	128	128
LLMLingua-2 Dynamic (Ours)	Forced	12	32	1e-4	64	128
Black-box target LLM	Standard	4	16	5e-5	16	16
Black-box target LLM	Forced	4	16	5e-6	16	64

I. Additional experimental results

As described in 3, Fig. 2 summarizes our experimental contributions. We observe a *large gap* between the optimal curve and existing prompt compression methods. Thus, we propose LLMLingua-2 as a *query-aware, variable-rate modification* of LLMLingua-2. Our results show that LLMLingua-2 Dynamic achieves the best performance and, in fact, is the only method to outperform the optimal query-agnostic strategy. We also note that the optimal distortion-rate curves eventually fall below the baseline performance of using the full prompt (no compression). This observation is especially interesting because it shows that compressing prompts can improve performance on downstream tasks, as observed on natural language datasets in previous prompt compression works (Jiang et al., 2023c;b; Pan et al., 2024). We accredit the performance of LLMLingua-2 Dynamic to variable-rate compression, where we allow the compressor to choose how much it should compress based on the query and prompt as input (see App. C, Remark 1 for a formal explanation of variable-rate compression). Even though this approach relinquishes explicit control over the rate, our experiments show that variable-rate compression is the closest to optimality. Our remaining empirical results are as follows.

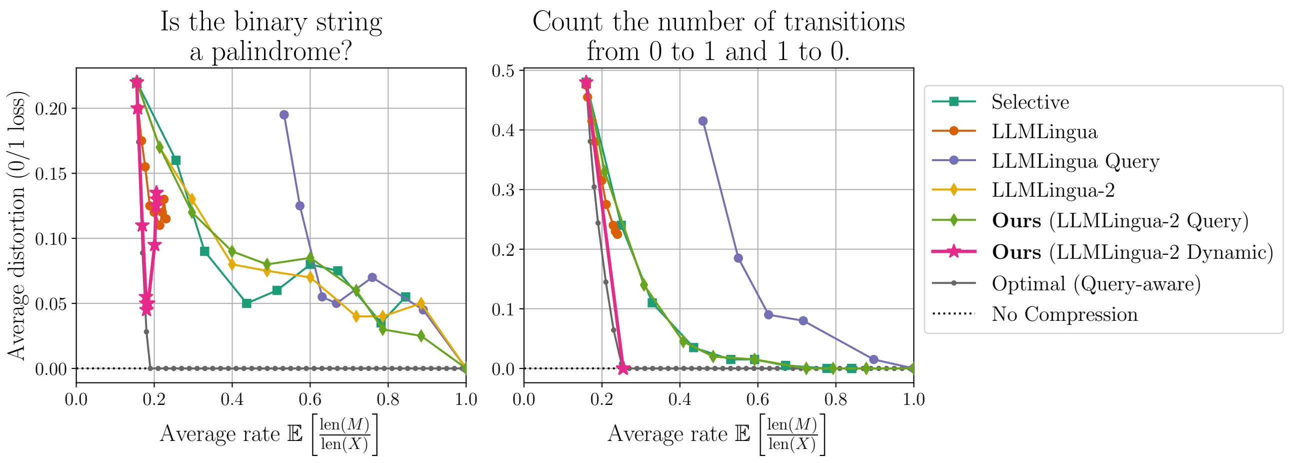


Figure 7: We highlight the distortion-rate curves for two of the seven queries in the validation partition of our synthetic dataset. Our method, LLMLingua-2 Dynamic, is able to match the performance of the optimal query-aware strategy **(left)**. Some queries naturally incur less distortion than others with the target LLM, even with a query-agnostic approach, if the query is aligned well with the data generation process for the prompt **(right)**. Note that LLMLingua-2 Query covers the line of LLMLingua-2 as their performance is identical for this query.

Gap from optimality depends on the query. In Fig. 7, we highlight the distortion-rate curves for two out of seven of the queries in our synthetic dataset. Despite the fact that Fig. 2 shows a gap in average performance between the query-aware optimal strategy and LLMLingua-2 Dynamic, Fig. 7 **(left)** shows that LLMLingua-2 Dynamic can match the performance of the optimal query-aware compression scheme. Comparing Fig. 7 **(left)** and **(right)**, we see that the prompt compression problem is easier (methods are closer to optimality) for certain tasks or queries depending on how the prompts were generated. For our synthetic dataset, all prompts are generated from a Markov chain with a transition probability of 0.1 and a probability of 0.9 for remaining in the same state. This means the tokens with the highest entropy are those that are part of a transition, and those tokens are the most important for answering this query. As a result, we see that methods that use the negative log-likelihood as a means for compression (Selective Context, LLMLingua, and LLMLingua Query) perform well, even without conditioning on the query. An exception here is the performance of LLMLingua Query, which we find has mixed performance compared to vanilla LLMLingua for token-level prompt compression on our dataset. LLMLingua Query performs markedly worse for this query (refer to Fig. 10 for results on all queries).

Effect of tokenization. The results of our ablation study on the effects of tokenization are provided in Fig. 8, which shows a direct comparison between the trade-off for methods using standard and forced tokenization. Interestingly, the optimal curves are nearly identical, suggesting that tokenization does not play a role in attaining the best possible trade-off. Furthermore, we see that, for a fixed rate, the standard tokenization performance often matches or exceeds the performance of forced tokenization. However, the standard tokenization approach does not allow for average rates below 0.6 due to the limited size of the prompts in our synthetic dataset, so the comparison is somewhat limited. In particular, standard tokenization allows for compression of at most four tokens (but usually only two or three tokens), whereas forced tokenization allows for compression of at most ten tokens. Fig. 9 is similar to Fig. 2, but shows the result when the prompt compression method uses standard tokenization rather than forced tokenization.

Trade-off for each query. Finally, Fig. 10, Fig. 11, Fig. 12, and Fig. 13 show the rate-distortion trade-off curves for each of the seven queries in our synthetic dataset. Fig. 10 shows forced tokenization with 0/1 loss, Fig. 11 shows forced tokenization with log loss, Fig. 12 shows standard tokenization with 0/1 loss, and Fig. 13 shows standard tokenization with log loss.

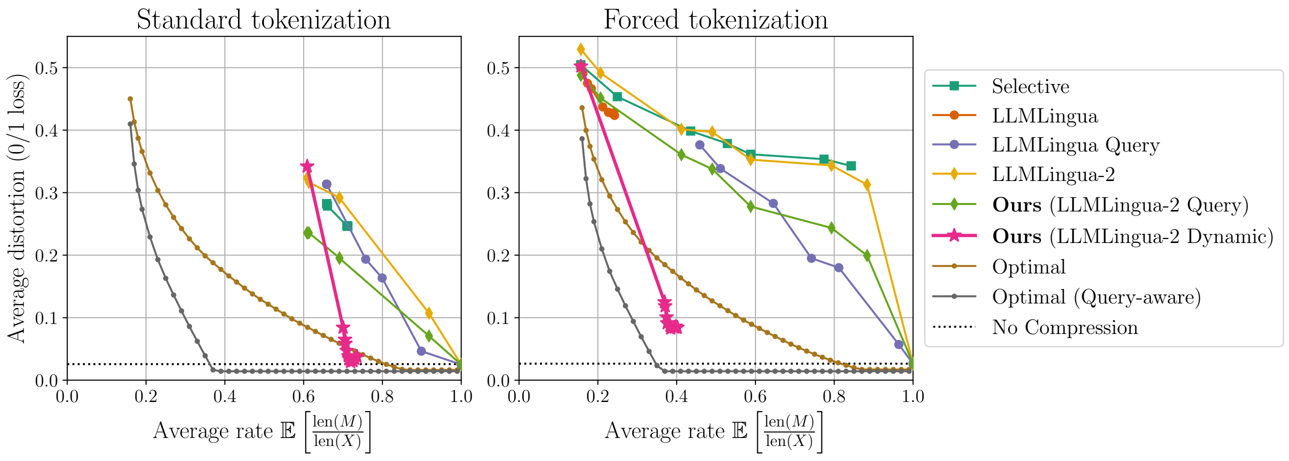


Figure 8: Performance comparison when standard tokenization **(left)** and forced tokenization **(right)** is used on our synthetic dataset. Interestingly, the optimal performance is nearly equivalent between the two, and, for a given rate, methods with standard tokenization match or improve upon the performance of a method that forced separate tokenization of every bit. However, standard tokenization results in compression of 1 to 4 tokens on our dataset, whereas forced tokenization compresses up to 10 tokens, allowing for a greater range of rates.

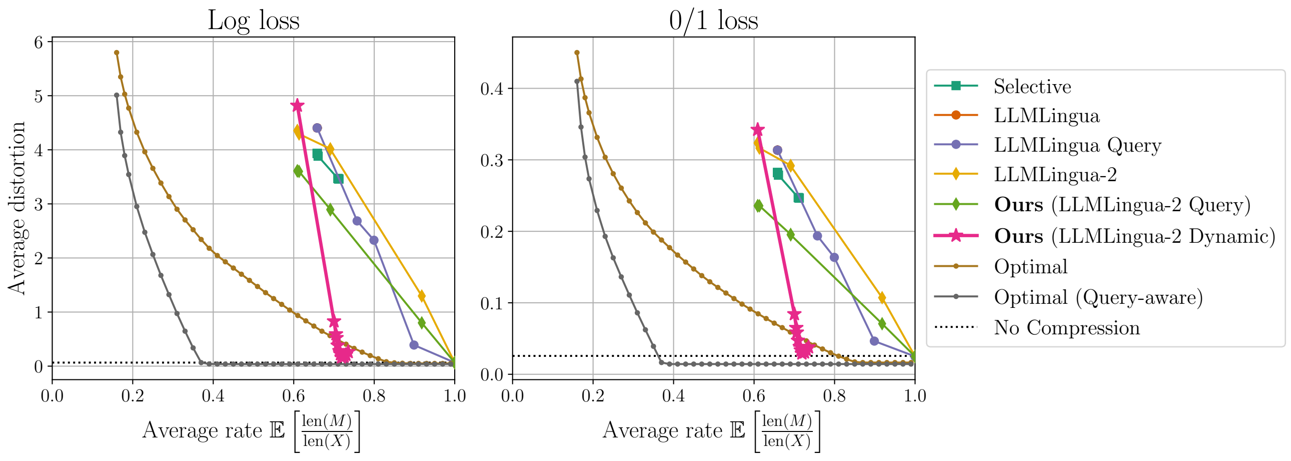


Figure 9: The distortion-rate curves of all prompt compression methods and the optimal strategy attained by solving our dual LP formulation when standard tokenization is used for the prompt. All methods are compared with the log loss **(left)** and 0/1 loss **(right)** metrics.

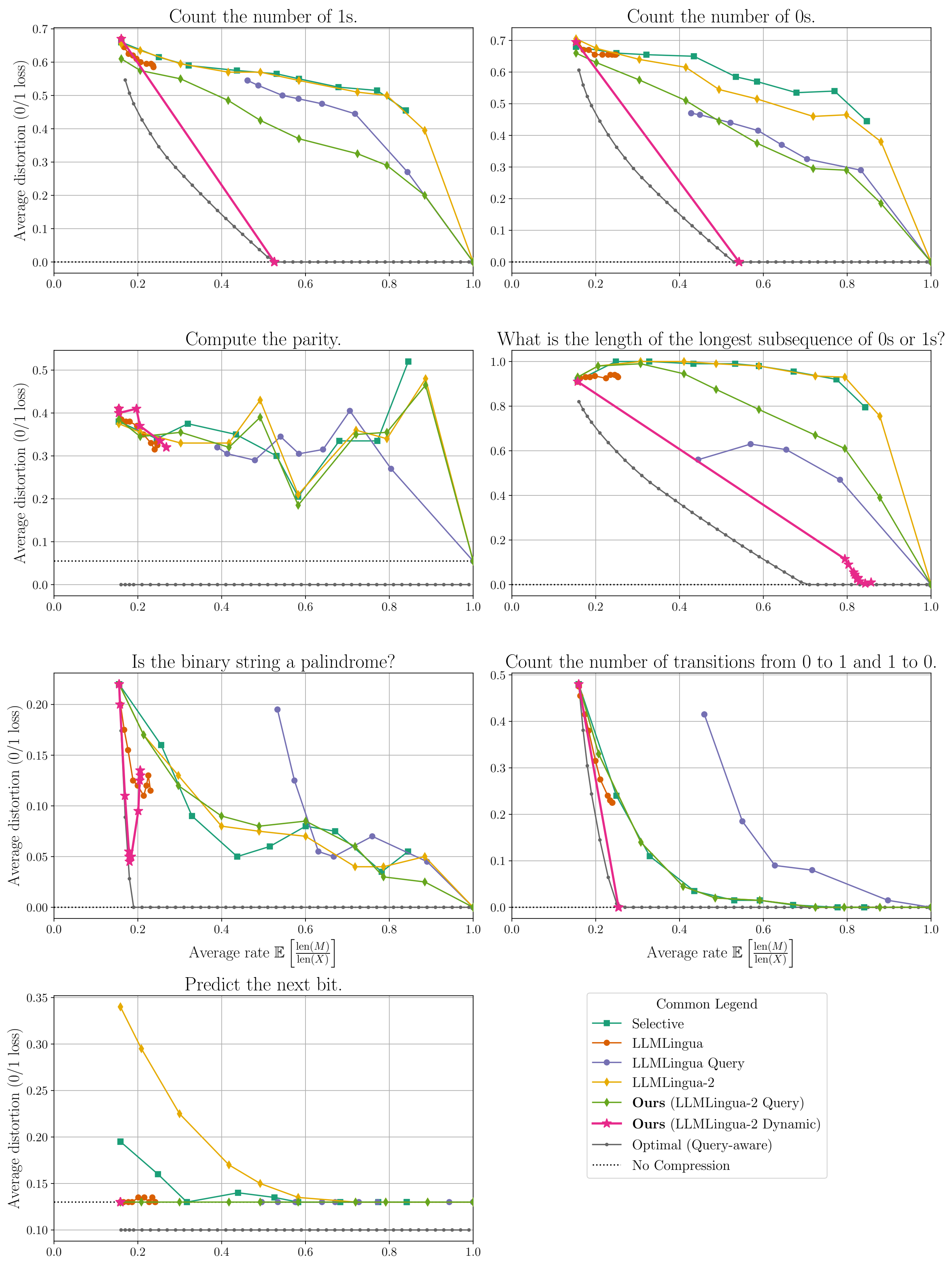


Figure 10: The rate-distortion trade-off of all methods on each individual query for forced tokenization and 0/1 loss.

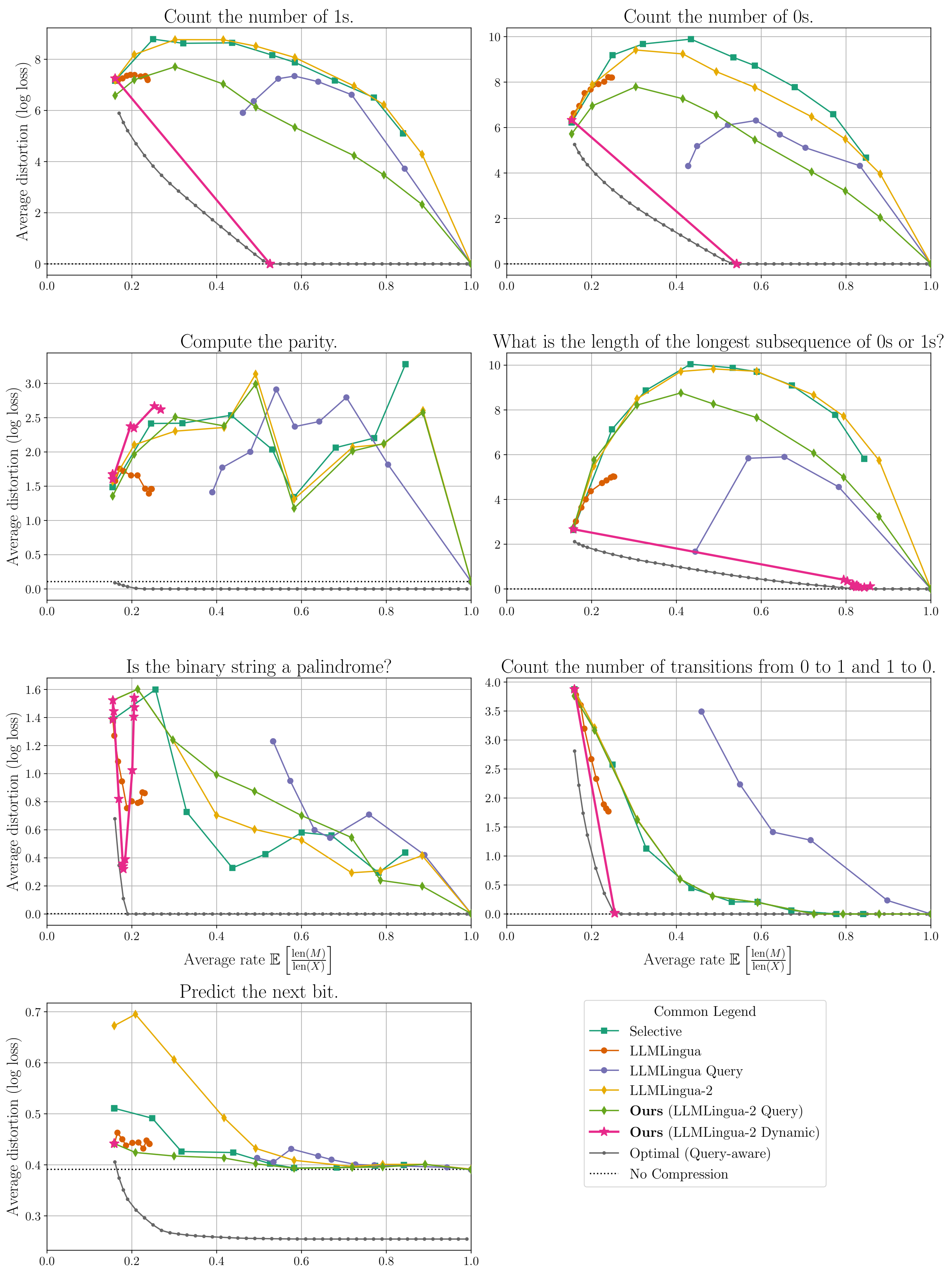


Figure 11: The rate-distortion trade-off of all methods on each individual query for forced tokenization and log loss.

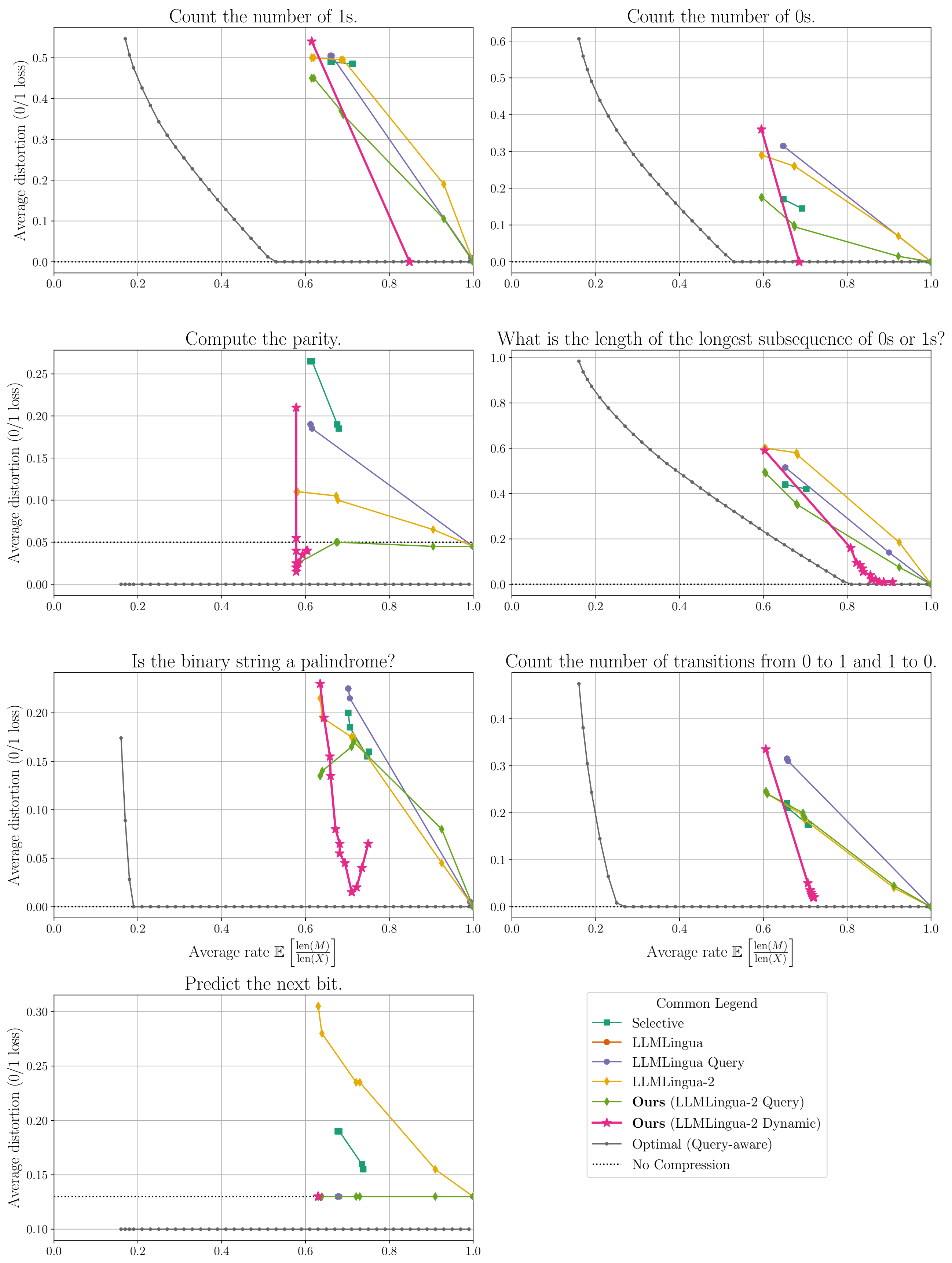


Figure 12: The rate-distortion trade-off of all methods on each individual query for standard tokenization and 0/1 loss.

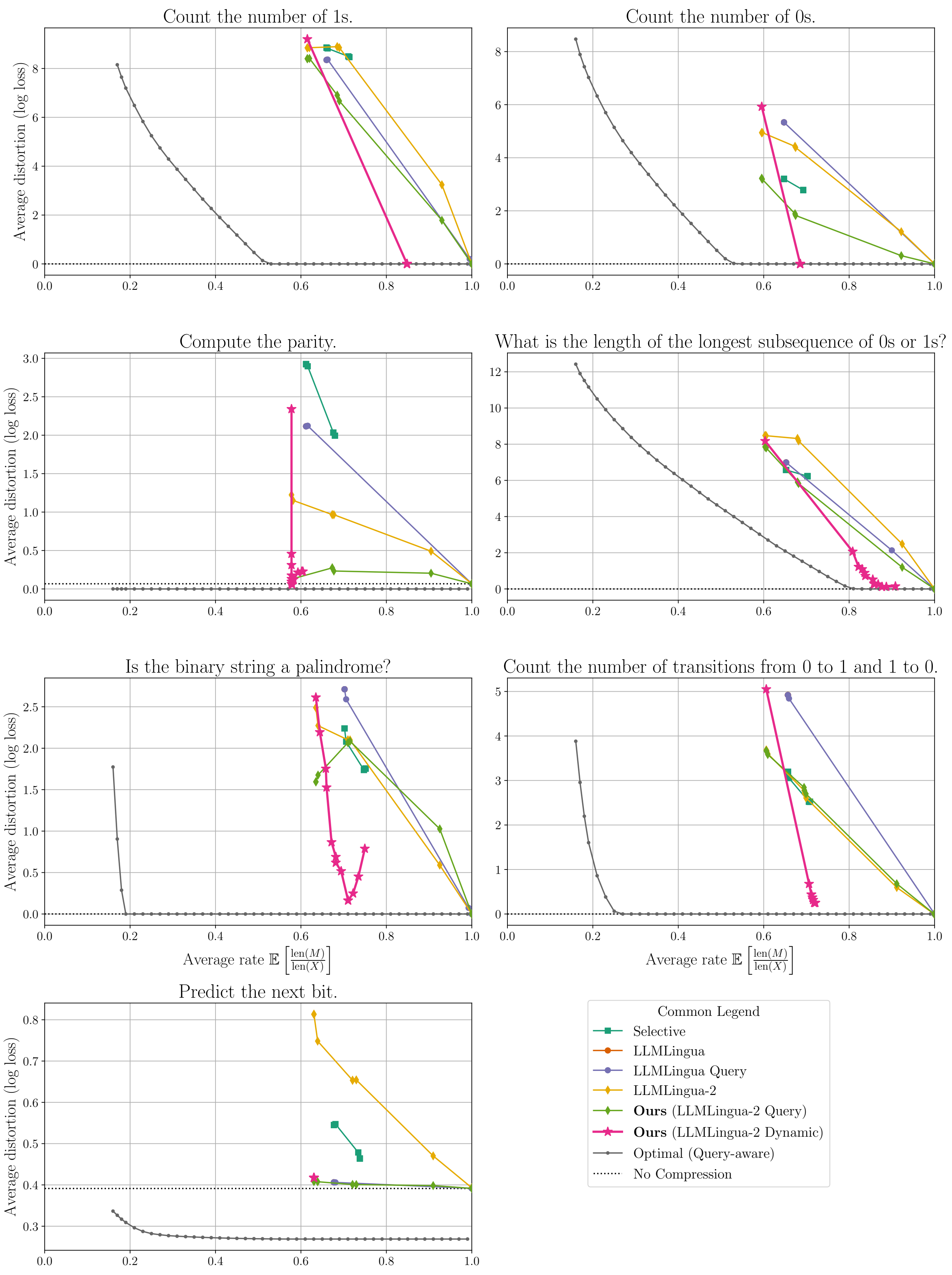


Figure 13: The rate-distortion trade-off of all methods on each individual query for standard tokenization and log loss.