TOKENIZATION WITH FORMAL GUARANTEES: FROM OPTIMAL COMPRESSION TO PROVABLE BOUNDS

Anonymous authors
Paper under double-blind review

ABSTRACT

Tokenization maps text into sequences of tokens from a fixed vocabulary and is a core component of modern NLP pipelines. Recent complexity-theoretic work formalizes tokenization as selecting tokens that optimize a compression objective over a corpus, and shows that computing an optimal solution is NP-complete. This shows that widely used tokenization methods such as Byte-Pair Encoding (BPE) are inherently heuristic and do not guarantee optimality. While this line of work has largely remained theoretical, Lim et al. (2025) recently took a practical step by introducing a mixed-integer programming (MIP) formulation and a greedy approximation algorithm for this objective. However, their MIP formulation suffers from a parameter explosion that makes it impractical for modern solvers such as Gurobi, while the greedy algorithm provides no optimality guarantee. In this work, we introduce a new *graph-flow MIP formulation* for tokenization that substantially reduces the number of decision variables and constraints, making optimal compression-based token selection tractable for state-of-the-art solvers. This enables, for the first time at this scale, the computation of *provably optimal* compression of token vocabularies for large corpora. To further improve scalability, we introduce a *Lagrangian relaxation* based on a shortest-path selection formulation. Our experiments show that our exact MIP algorithms achieve provably optimal compression and substantially outperform existing compression-based tokenization schemes, even on corpora with millions of candidate tokens. Moreover, our Lagrangian relaxation scales to significantly larger corpora, substantially improving over the state of the art in compression, and providing *provable upper and lower bounds* on the optimal value. Overall, this work takes a significant step toward turning optimal token selection from a theoretical objective into a practical algorithmic tool for building highly compressed vocabularies for real-world data.

1 INTRODUCTION

Tokenization is the process of mapping raw text into sequences of discrete symbols from a finite vocabulary, and forms the foundation of modern natural language processing pipelines. Virtually all large language models rely on tokenization as the interface between text and computation, making the quality of the tokenizer a first-order factor in both efficiency and modeling performance. Among existing methods, Byte-Pair Encoding (BPE) (Gage, 1994; Sennrich et al., 2016; Kaddour et al., 2023) has emerged as the dominant paradigm due to its simplicity and scalability, constructing vocabularies through a sequence of greedy merges over character pairs. Despite its widespread adoption, BPE is fundamentally heuristic: it makes locally optimal merge decisions without directly optimizing the global compression objective induced over the corpus.

Recent theoretical work has begun formalizing tokenization as an explicit combinatorial optimization problem, asking for a vocabulary of bounded size that minimizes a compression objective (Whittington et al., 2025; Kastreva et al., 2026; Lim et al., 2025; Kozma & Voderholzer, 2024; Zouhar et al., 2023). This line of work establishes a theoretical connection between tokenization and compression, and shows that computing an optimal compression token vocabulary is NP-hard (Whittington et al., 2025; Lim et al., 2025; Kozma & Voderholzer, 2024). These results provide an important conceptual shift: compression-based tokenization is not merely a preprocessing heuristic, but a computationally hard optimization problem with a well-defined objective. At the same time, they imply that widely used tokenization methods such as BPE are inherently heuristic and may be far from optimal.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

A major open challenge, however, is turning this new theoretical perspective into practical optimization methods. Most existing approaches remain heuristic and provide no guarantees on optimality. In contrast, a recent work of Lim et al. (2025) introduced the first mixed-integer programming (MIP) formulation for optimal compression-based tokenization, which then serves as the basis for a greedy algorithm for scaling to larger corpora. While this constitutes an important practical bridge from theory to implementation, the exact MIP formulation suffers from a parameter explosion that limits its scalability for modern solvers, and the greedy algorithm, while effective in practice, provides neither optimality guarantees nor provable approximation guarantees.

Our Contributions. In this work, we revisit the optimization structure underlying compression-based tokenization and introduce a new practical optimization framework with provable guarantees. Our central observation is that tokenization can be interpreted as a *path-selection problem* in a directed acyclic graph (DAG), where each valid path corresponds to a feasible partition of a word into tokens. Building on this perspective, we develop a new *flow-based MIP formulation* that avoids the parameter explosion of prior exact formulations and makes exact optimization substantially more tractable for modern MIP solvers like Gurobi (Gurobi Optimization, LLC, 2026). This enables, to the best of our knowledge, the first computation of *provably optimal* compression of token vocabularies at scales that were previously intractable.

Beyond exact optimization, we develop a *Lagrangian relaxation* (Lemaréchal, 2001; Fisher, 1981) of the flow-based formulation. By dualizing the token-selection consistency constraints, the problem decomposes into shortest-paths over weighted DAGs, enabling scalable, parallel optimization across large corpora. Importantly, this relaxation maintains *provable lower and upper bounds* on the optimal compression, yielding certified approximations while scaling substantially beyond exact MIP solving.

Our empirical evaluation demonstrates that these formulations significantly improve the practical tractability of optimal compression of tokenization. Our exact flow-based MIP solves instances that are intractable for prior exact formulations while certifying optimality. At larger scales, our Lagrangian relaxation achieves substantially stronger compression than existing tokenization methods, while simultaneously providing certified tight upper and lower bounds on the optimal solution.

Overall, our results take a significant step toward closing the gap between the theoretical understanding of optimal compression for tokenization and practical algorithms for computing highly compressed token vocabularies with provable guarantees. Due to space constraints, we discuss the central ideas behind selected propositions in the main text and defer the full proofs to Appendix B.

2 PRELIMINARIES

Notation. Let Σ be a fixed alphabet (e.g., Unicode), and let a corpus be denoted by $\mathcal{C} := \langle \mathcal{W}, f \rangle$, where $\mathcal{W}$ is a set of distinct words and $f : \mathcal{W} \to \mathbb{N}$ is a frequency function assigning each word its number of occurrences in the corpus. For a word $w \in \mathcal{W}$, let $|w|$ denote its length. We denote by $\mathcal{B}(\mathcal{C}) \subseteq \Sigma$ the set of singleton characters, typically those appearing in the corpus or all of Σ. Let $\mathcal{T} \subseteq \Sigma^+$ be a set of candidate tokens containing $\mathcal{B}(\mathcal{C})$, typically taken as all substrings of words in $\mathcal{W}$. Following (Lim et al., 2025), for a word $w \in \Sigma^+$ and a token set $\mathcal{S} \subseteq \mathcal{T}$, let partition$(w, \mathcal{B}(\mathcal{C}) \cup \mathcal{S})$ denote the minimum number of tokens in $\mathcal{B}(\mathcal{C}) \cup \mathcal{S}$ whose concatenation forms w. This induces the corpus compression objective:

$$\text{Comp}(\mathcal{C}, \mathcal{S}) := \sum_{w \in \mathcal{W}} f(w) \cdot \text{partition}(w, \mathcal{B}(\mathcal{C}) \cup \mathcal{S}). \tag{1}$$

Minimizing $\text{Comp}(\mathcal{C}, \mathcal{S})$ under a token budget $k \in \mathbb{N}$ bounding the size of $|\mathcal{S}|$ is equivalent to the partition-minimization / coverage-maximization formulation of Lim et al. (2025), and coincides with the *direct tokenization* objective studied by recent works (Whittington et al., 2025; Kastreva et al., 2026). Particularly, under a budget $k \in \mathbb{N}$, we define this optimal compression problem as:

Optimal Compression for Tokenization (Problem Statement):
Input: A corpus $\mathcal{C} = \langle \mathcal{W}, f \rangle$, a set $\mathcal{T}$ of candidate tokens (e.g., all substrings of words in $\mathcal{W}$), and a token budget $k \in \mathbb{N}$.
Output: A subset $\mathcal{S} \subseteq \mathcal{T} \setminus \mathcal{B}(\mathcal{C})$ of candidate tokens such that $|\mathcal{S}| \leq k$, for which $\text{Comp}(\mathcal{C}, \mathcal{S})$ is minimized. We denote this optimal compression value for corpus $\mathcal{C}$ under budget k as $\text{Comp}^*_k(\mathcal{C}, \mathcal{T}) := \min_{\mathcal{S} \subseteq \mathcal{T} \setminus \mathcal{B}(\mathcal{C}), |\mathcal{S}| \leq k} \text{Comp}(\mathcal{C}, \mathcal{S})$.

3 A MIP WITH $\Theta((\sum_{w\in\mathcal{W}}|w|)\cdot|\mathcal{T}|)$ VARIABLES AND CONSTRAINTS

Lim et al. (2025) develop a MIP formulation of tokenization as a compression problem over adjacent character positions, used mainly as an auxiliary construct for deriving a greedy algorithm rather than as a model to be solved directly. The formulation selects a vocabulary of at most k tokens and maximizes the total number of adjacent merges across the corpus, weighted by word frequency. Additional variables indicate whether a selected token covers a particular span in a word, and the constraints ensure that token usages are contiguous and non-overlapping, thus inducing a valid partition of each word.

Formally, let $\mathcal{W}$ be the corpus, $\mathcal{T}$ the set of candidate tokens, and k the token budget. For a word $w = (w_0, \ldots, w_{|w|-1})$, the formulation introduces the following binary variables:

- $\mathbf{x}_t \in \{0,1\}$ for each $t \in \mathcal{T}$, indicating whether token t is selected;
- $\mathbf{m}^w_{i,i+1} \in \{0,1\}$ for each $w \in \mathcal{W}$ and $i \in \{0,\ldots,|w|-2\}$, indicating whether adjacent symbols w_i and w_{i+1} are merged;
- $\mathbf{m}^{w,t}_{i,i+1} \in \{0,1\}$ for each $w \in \mathcal{W}$, $t \in \mathcal{T}$, and $i \in \{0,\ldots,|w|-2\}$, indicating whether token t covers the pair (w_i, w_{i+1}).

The objective is: $\min \sum_{w\in\mathcal{W}} f(w)\cdot\left(|w| - \sum_{i=0}^{|w|-2}\mathbf{m}^w_{i,i+1}\right)$ subject to

- $\sum_{t\in\mathcal{T}\setminus\mathcal{B}(\mathcal{C})}\mathbf{x}_t \le k$,
- $\mathbf{x}_t \ge \mathbf{m}^{w,t}_{i,i+1}$ $\quad \forall w\in\mathcal{W}, \forall t\in\mathcal{T}, \forall 0\le i<|w|-1,$ s.t. $(w_i, w_{i+1})\subseteq t$,
- $\sum_{t\in\mathcal{T}}\mathbf{m}^{w,t}_{i,i+1} \ge \mathbf{m}^w_{i,i+1}$, $\quad \forall w\in\mathcal{W}, \forall 0\le i<|w|-1$,
- $\sum_{t\in\mathcal{T}}\mathbf{m}^{w,t}_{i,i+1} \le 1$ $\quad \forall w\in\mathcal{W}, \forall 0\le i<|w|-1$,
- $\mathbf{m}^{w,t}_{i,i+1} = \mathbf{m}^{w,t}_{i+1,i+2}$ $\quad \forall w\in\mathcal{W}, \forall t\in\mathcal{T}, \forall 0\le i<|w|-2$ s.t. $(w_i, w_{i+1}, w_{i+2})\subseteq t$,
- $\sum_{t\in\mathcal{T}}\mathbf{m}^{w,t}_{s-1,s} \le 1-\mathbf{m}^{w,t}_{s,s+1}$ $\quad \forall w\in\mathcal{W}, \forall t\in\mathcal{T}, \forall 1\le s<|w|-1,$ s.t. t starts at w_s,
- $\sum_{t\in\mathcal{T}}\mathbf{m}^{w,t}_{e,e+1} \le 1-\mathbf{m}^{w,t}_{e-1,e}$ $\quad \forall w\in\mathcal{W}, \forall t\in\mathcal{T}, \forall 1\le e<|w|-1,$ s.t. t ends at w_e.

Together, these constraints enforce a valid tokenization of each word: the first constraint limits the vocabulary size, the second ensures that only selected tokens may be used, the next two link merges to covering tokens and enforce unique coverage, the equality constraints enforce contiguity of token spans, and the boundary constraints prevent overlaps between distinct tokens. More details appear in Appendix A. A central limitation of this formulation, however, is its *size*: it introduces variables indexed by words, tokens, and positions. This leads to the following bound.

Proposition 1. *The number of binary variables defined by the MIP of Lim et al. (2025) is* $\Theta(|\mathcal{T}| + \sum_{w\in\mathcal{W}}|w|\cdot|\mathcal{T}|) = \Theta(\sum_{w\in\mathcal{W}}|w|\cdot|\mathcal{T}|)$. *This bound also applies to the number of constraints.*

4 A FLOW-BASED MIP WITH $\Theta(\sum_{w\in\mathcal{W}}|w|^2)$ VARIABLES AND CONSTRAINTS

In this section, we propose a new MIP encoding that yields the same optimal solution as (Lim et al., 2025), but with typically substantially fewer binary variables and constraints, making it better suited for modern MIP solvers and significantly improving practical runtime.

Formally, let $\mathcal{C} = \langle\mathcal{W}, f\rangle$ be the corpus, let $\mathcal{T}$ be the set of candidate tokens, and let $k \in \mathbb{N}$ be the token budget. We construct a graph $G := \langle V, E\rangle$ as follows. For each word $w \in \mathcal{W}$, we create vertices $v^w_0, \ldots, v^w_{|w|}$, where v^w_i marks the position right before character w_i. For every pair of positions $0 \le i < j \le |w|$ such that the substring $w[i:j] := w_i \cdots w_{j-1}$ is a candidate token, we add a directed edge $e^w_{i,j} = (v^w_i, v^w_j)$, representing the use of this token. Each word thus induces a DAG $G_w = \langle V_w, E_w\rangle$ (complete when $\mathcal{T}$ contains all substrings), and G is their disjoint union. We denote by $\mathrm{IN}(v^w_j)$ and $\mathrm{OUT}(v^w_j)$ the sets of incoming and outgoing edges of v^w_j, respectively.

Now, for a word $w = (w_0, \ldots, w_{|w|-1})$, we introduce the following binary variables:

- $\mathbf{x}_t \in \{0,1\}$ for each $t \in \mathcal{T}$, indicating whether token t is selected.
- $\mathbf{e}^w_{i,j} \in \{0,1\}$ for each edge $e^w_{i,j} \in E$, indicating whether edge $e^w_{i,j}$ carries flow.

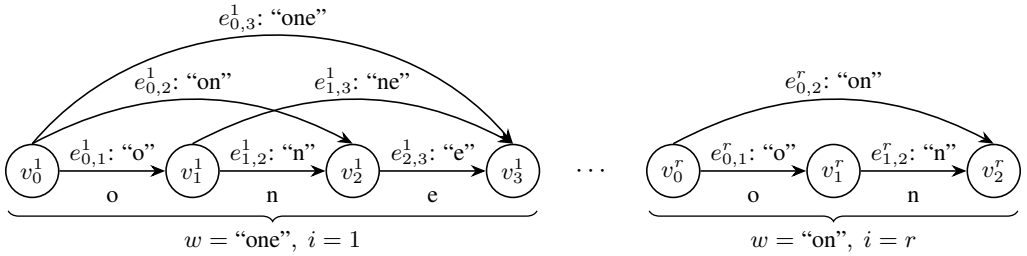

Figure 1: An illustration of the graph encoding G of a given corpus $\mathcal{C} = \langle \mathcal{W}, f \rangle$, where w="one"$\in \mathcal{W}$ and w="on"$\in \mathcal{W}$. All vertices and edges are shown in the illustration.

The objective is:

$$\min \sum_{w \in \mathcal{W}} f(w) \cdot \left(\sum_{i=0}^{|w|-1} \left(\sum_{j=i+1}^{|w|} \mathbf{e}^w_{i,j} \right) \right) \tag{2}$$

subject to

- $\sum_{t \in \mathcal{T} \setminus \mathcal{B}(\mathcal{C})} \mathbf{x}_t \leq k,$
- $\mathbf{x}_b = 1$ $\quad \forall b \in \mathcal{B}(\mathcal{C}),$
- $\mathbf{e}^w_{i,j} \leq \mathbf{x}_{w[i:j]}$ $\quad \forall w \in \mathcal{W},\ \forall e^w_{i,j} \in E_w,$
- $\sum_{e^w_{i,j} \in \mathrm{OUT}(v^w_0)} \mathbf{e}^w_{i,j} = 1$ $\quad \forall w \in \mathcal{W},$
- $\sum_{e^w_{i,j} \in \mathrm{IN}(v^w_l)} \mathbf{e}^w_{i,j} = \sum_{e^w_{i,j} \in \mathrm{OUT}(v^w_l)} \mathbf{e}^w_{i,j}$ $\quad \forall w \in \mathcal{W},\ \forall\, 0 < l < |w|.$

Here, the constraint $\sum_{t \in \mathcal{T} \setminus \mathcal{B}(\mathcal{C})} \mathbf{x}_t \leq k$ allows at most k tokens to be selected beyond the singletons, and $\mathbf{x}_b = 1$ ensures that every singleton $b \in \mathcal{B}(\mathcal{C})$ is selected. The constraint $\mathbf{e}^w_{i,j} \leq \mathbf{x}_{w[i:j]}$ allows flow on an edge only if its token is selected. The last two constraints send one unit of flow out of the source v^w_0 and conserve it at every internal vertex, so the sink $v^w_{|w|}$ receives exactly one unit and each word graph carries exactly one path of unit flow: edges on this path are set to 1 and all others to 0. This path is the "coverage" of the word by selected tokens.

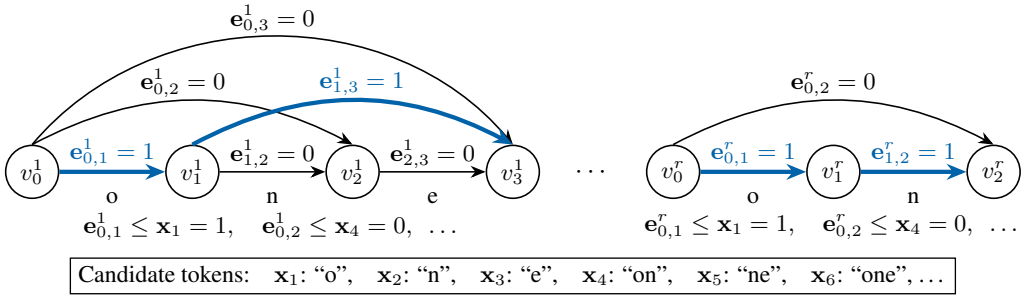

Figure 2: An illustration of a flow in a graph encoding G of a given corpus $\mathcal{C}$ with words "one"$\in \mathcal{W}$ and "on"$\in \mathcal{W}$. In the flow (**bold blue**), tokens "o" and "ne" cover "one", and "o" and "n" cover "on".

Proposition 2. *Given a corpus $\langle \mathcal{W}, f \rangle$, a token budget $k \in \mathbb{N}$, a graph $G = \langle V, E \rangle$ representing $\mathcal{C}$, and a set of candidate tokens $\mathcal{T} \subseteq \Sigma^+$, then solving the encoding described in Equation 2 such that all $\mathbf{x}$ variables are chosen as indicators for the candidate tokens in $\mathcal{T}$ and all $\mathbf{e}$ variables are chosen as indicator variables for the edges in E, obtains an optimal compression solution $\mathrm{Comp}^*_k(\mathcal{C}, \mathcal{T})$.*

Number of binary variables and constraints. Unlike the previous MIP encoding proposed by (Lim et al., 2025), and presented in Section 3, which uses $\Theta(\sum_{w \in \mathcal{W}} |w| \cdot |\mathcal{T}|)$ binary variables and $\Theta(\sum_{w \in \mathcal{W}} |w| \cdot |\mathcal{T}|)$ constraints, our encoding avoids the $\sum_{w \in \mathcal{W}} |w| \cdot |\mathcal{T}|$ factor, i.e., it does not introduce a separate token indicator for each substring occurrence. Instead, it depends only on the set of distinct substrings. We formalize this in the following proposition:

Proposition 3. *The number of binary variables defined by the MIP described in Section 4 is $\Theta(|\mathcal{T}| + \sum_{w \in \mathcal{W}} |w|^2) = \Theta(\sum_{w \in \mathcal{W}} |w|^2)$. This bound also applies to the number of constraints.*

5 A LAGRANGIAN RELAXATION OF THE MIP FORMULATION

Despite the substantial reduction in the size of our MIP encoding in Section 4, applying modern MIP solvers to LLM-scale tokenization data may still be challenging. *Lagrangian relaxation* is a common technique for simplifying difficult MIPs by moving selected hard constraints into the objective as penalty terms weighted by Lagrange multipliers (Lemaréchal, 2001; Fisher, 1981). This often makes the problem easier to solve or split into simpler subproblems, while still providing *provable bounds* on the original optimum. The multipliers are then updated iteratively to improve the bounds and guide the search toward good feasible solutions (Geoffrion, 1974; Nemhauser & Wolsey, 1988).

We show that our formulation is well suited for Lagrangian relaxation by relaxing precisely its coupling constraints: the token-consistency constraints $\mathbf{e}^w_{i,j} \le \mathbf{x}_{w[i:j]}$, which connect the local path choices in each word graph with the global vocabulary variables. Let $\lambda : E \to \mathbb{R}_{\ge 0}$ be a multiplier function assigning a non-negative penalty to every edge. For a fixed λ, define the relaxed objective

$$\min \left[\sum_{w \in \mathcal{W}} f(w) \cdot \left(\sum_{i=0}^{|w|-1} \left(\sum_{j=i+1}^{|w|} \mathbf{e}^w_{i,j} \right) \right) + \sum_{w \in \mathcal{W}} \left(\sum_{i=0}^{|w|-1} \left(\sum_{j=i+1}^{|w|} \lambda(e^w_{i,j})(\mathbf{e}^w_{i,j} - \mathbf{x}_{w[i:j]}) \right) \right) \right] \quad (3)$$

subject to

- $\sum_{t \in \mathcal{T} \setminus \mathcal{B}(\mathcal{C})} \mathbf{x}_t \le k,$
- $\mathbf{x}_b = 1 \qquad \forall b \in \mathcal{B}(\mathcal{C}),$
- $\sum_{e^w_{i,j} \in \mathrm{OUT}(v^w_0)} \mathbf{e}^w_{i,j} = 1 \qquad \forall w \in \mathcal{W},$
- $\sum_{e^w_{i,j} \in \mathrm{IN}(v^w_l)} \mathbf{e}^w_{i,j} = \sum_{e^w_{i,j} \in \mathrm{OUT}(v^w_l)} \mathbf{e}^w_{i,j} \qquad \forall w \in \mathcal{W},\ \forall\, 0 < l < |w|.$

The first term in the optimization objective is the original compression objective, while the second penalizes using an edge whose token is not selected. Thus, $\lambda(e^w_{i,j})$ is the penalty for violating the corresponding consistency constraint. We relax only the constraints $\mathbf{e}^w_{i,j} \le \mathbf{x}_{w[i:j]}$, while keeping the budget and flow constraints. For a fixed multiplier function λ, let $q(\lambda)$ denote the optimal value of the relaxed problem in Equation 3. Then every fixed multiplier function yields a valid lower bound:

Proposition 4. *Given a corpus $\mathcal{C}$, a graph $G = \langle V, E \rangle$ representing it, and a set of candidate tokens $\mathcal{T}$, then for every multiplier function $\lambda : E \to \mathbb{R}_{\ge 0}$, the value $q(\lambda)$ is a lower bound on the optimal value of the original flow-based MIP in Equation 2, i.e., $q(\lambda) \le \mathit{Comp}^*_k(\mathcal{C}, \mathcal{T})$.*

For a fixed multiplier function λ, the relaxed objective in Equation 3 can be optimized efficiently. After rewriting, each edge $e^w_{i,j}$ has weight $c_\lambda(e^w_{i,j}) = f(w) + \lambda(e^w_{i,j})$, so minimizing over the flow variables $\mathbf{e}$ reduces, for each G_w, to finding a shortest path under these weights (Figure 3). The selected path determines the binary edge variables: edges on the path are set to 1, and all others to 0.

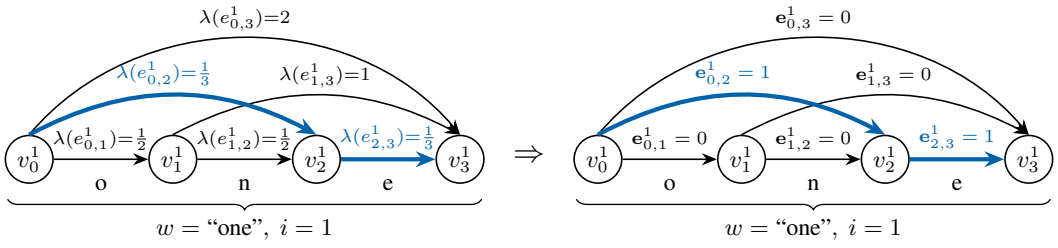

Figure 3: Illustration of the graph encoding G for $\mathcal{C} = \langle \mathcal{W}, f \rangle$, focusing on $w =$ "one" $\in \mathcal{W}$. The left figure shows the **shortest root-to-sink path** under weights $f(w) + \lambda$ (here $f(w) = 1$), and the right figure shows the induced binary edge assignment. Here, the path using "on" and "e" is selected.

Proposition 5. *Given a corpus $\mathcal{C}$, a graph $G = \langle V, E \rangle$ representing it, and a set of candidate tokens $\mathcal{T}$, then for every multiplier function $\lambda : E \to \mathbb{R}_{\ge 0}$, the assignments to $\mathbf{e}$ and $\mathbf{x}$ computed in lines 3–6 of Algorithm 1 attain the optimal value LB of the relaxed problem in Equation 3.*

Proposition 5 is significant since it shows that, for any fixed λ, the relaxed problem can be solved *optimally* and efficiently: each word graph G_w is a DAG, so its shortest path can be computed in

linear time, and the relaxed optimum LB is thus computed in time linear in the graph size. Moreover, since word graphs are independent, these computations can be performed fully in parallel across the corpus, making the relaxation highly scalable in practice.

Algorithm 1 Lagrangian relaxation of the flow-based MIP

Input: $G = \langle V, E \rangle$ representing $\mathcal{C}$, budget k, number of iterations n, step parameter $\mu \in (0, 2]$.
Initialize $\lambda(e^w_{i,j}) \leftarrow 0$ for all $e^w_{i,j} \in E$, $\mathrm{LB}^* \leftarrow -\infty$, and $\mathrm{UB}^* \leftarrow +\infty$
for n iterations **do**
  $\mathbf{e} \leftarrow$ a shortest source-to-sink path in each G_w under edge weights $f(w) + \lambda(e^w_{i,j})$
  $s_\lambda(t) \leftarrow \sum_{e^w_{i,j} \in E:\, w[i:j]=t} \lambda(e^w_{i,j})$ for each token $t \in \mathcal{T}$
  $\mathcal{S} \leftarrow$ the k tokens $t \in \mathcal{T} \setminus \mathcal{B}$ with the largest values of $s_\lambda(t)$
  Set $\mathbf{x}_t = 1$ for $t \in \mathcal{S} \cup \mathcal{B}$, and $\mathbf{x}_t = 0$ otherwise
  $\mathrm{LB} \leftarrow$ the value of Eq. 3 at $(\mathbf{x}, \mathbf{e})$
  $\mathrm{LB}^* \leftarrow \max(\mathrm{LB}^*, \mathrm{LB})$
  $g^w_{i,j} \leftarrow \mathbf{e}^w_{i,j} - \mathbf{x}_{w[i:j]}$ for all $e^w_{i,j} \in E$ ▷ Subgradient
  $\hat{\mathbf{e}} \leftarrow$ a fewest-edge source-to-sink path in each G_w using only edges with $\mathbf{x}_{w[i:j]} = 1$
  $\mathrm{UB} \leftarrow$ the value of Eq. 2 at $(\mathbf{x}, \hat{\mathbf{e}})$; **if** $\mathrm{UB} < \mathrm{UB}^*$ **then** $\mathrm{UB}^* \leftarrow \mathrm{UB}$, $\mathbf{x}^* \leftarrow \mathbf{x}$
  if $\mathrm{UB}^* = \mathrm{LB}^*$ **then break** ▷ Certified optimal
  $\alpha \leftarrow \mu \cdot \frac{\mathrm{UB}^* - \mathrm{LB}}{\sum_{e^w_{i,j} \in E} |g^w_{i,j}|}$
  $\lambda(e^w_{i,j}) \leftarrow \max\{0,\ \lambda(e^w_{i,j}) + \alpha \cdot g^w_{i,j}\}$ for all $e^w_{i,j} \in E$ ▷ Projected Polyak step
end for
return $\mathbf{x}^*$, LB^*, UB^*

Since Proposition 4 guarantees that every multiplier function induces a valid lower bound, this naturally gives rise to the Lagrangian dual objective:

$$\max_{\lambda: E \to \mathbb{R}_{\geq 0}} q(\lambda),$$

namely, searching for the multiplier function that yields the tightest lower bound. We optimize it by projected subgradient ascent (Geoffrion, 1974; Nemhauser & Wolsey, 1988), where the subgradient $g^w_{i,j} = \mathbf{e}^w_{i,j} - \mathbf{x}_{w[i:j]}$ measures the violation of the relaxed consistency constraint, with a Polyak-type step size α (Polyak, 1969; Nemhauser & Wolsey, 1988; Hazan & Kakade, 2019). To obtain a feasible solution, we fix $\mathbf{x}$ and recompute, in each G_w, a shortest source-to-sink path using only edges with $\mathbf{x}_{w[i:j]} = 1$; its objective value defines UB. Algorithm 1 summarizes this iterative process.

Proposition 6. *At every iteration of Algorithm 1, LB is the optimal value of the relaxed problem for the current multiplier function and therefore a valid lower bound, while UB is the value of a feasible solution to the original flow-based MIP and therefore a valid upper bound. Consequently, $LB^* \leq Comp^*_k(\mathcal{C}, \mathcal{T}) \leq UB^*$, and the algorithm maintains a certified optimality gap $UB^* - LB^*$.*

6 Experimental Evaluation

Experimental setup. We evaluate our methods on four real-world text corpora: UN (Jankin et al., 2025), ArXiv (Clement et al., 2019), PubMed (Canese & Weis, 2013), and Wikitext (Merity et al., 2016). Dataset statistics appear in Table 1, with further details in Appendix C. We define $\mathcal{B}(\mathcal{C})$ as the 256 singleton byte characters, $\mathcal{W}$ as the set of space-delimited UTF-8 byte strings, and $\mathcal{T}$ as all substrings of words in $\mathcal{W}$. We report the compression ratio $\mathrm{CR}(\mathcal{C}, \mathcal{S}) := \mathrm{Comp}(\mathcal{C}, \mathcal{S}) / \sum_{w \in \mathcal{W}} f(w)$ (tokens per word; lower is better), normalizing LB and UB likewise. We compare our MIP encoding against (Lim et al., 2025), solving both with the state-of-the-art MIP solver Gurobi (Gurobi Optimization, LLC, 2026), and compare our more scalable Lagrangian relaxation to BPE (Gage, 1994) and GreedTok (Lim et al., 2025). For BPE, we start from the singleton byte vocabulary and add tokens until reaching size $|\mathcal{B}| + k$; GreedTok and our methods use a budget of k tokens. Unless stated otherwise, experiments run with 128 CPU cores, 128GB memory, and a 24-hour timeout (PubMed: 512GB, 48 hours; details in Appendix C); runtimes are wall-clock time.

Table 1: Dataset statistics

Dataset	$\sum_{w \in \mathcal{W}} f(w)$	$\lvert\mathcal{T}\rvert$	$\lvert\mathcal{W}\rvert$
Wikitext	2M	1.4M	80K
UN	37M	884K	105K
arXiv	38M	8M	1M
PubMed	4149M	121M	6527K

6.1 Provably Optimal Tokenization via our Flow-based MIP vs. prior MIP

Comparing the encoding size. We compare the encoding size of our flow-based MIP formulation for optimal tokenization (Section 4) with the MIP encoding of (Lim et al., 2025) (Section 3). We plot the number of binary variables induced by each encoding on the UN, arXiv, and PubMed datasets in Figure 4. The results show that replacing the $\Theta(\sum_{w\in\mathcal{W}} |w| \cdot |\mathcal{T}|)$ variables required by (Lim et al., 2025) (Proposition 1) with the $\Theta(\sum_{w\in\mathcal{W}} |w|^2)$ variables in our formulation (Proposition 3) substantially reduces the number of binary variables, with the gap growing as dataset size and $|\mathcal{W}|$ increase.

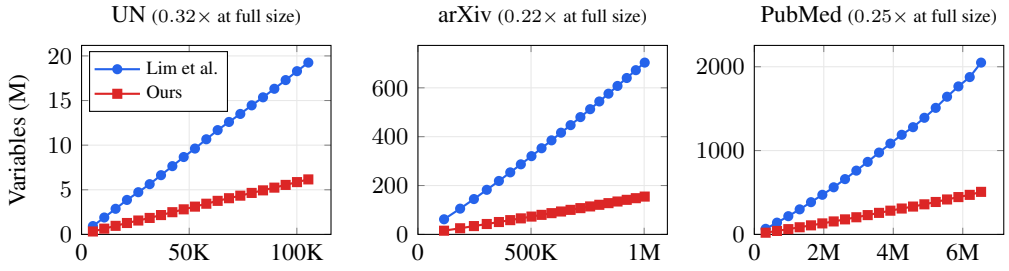

Figure 4: Number of variables in the MIP of Lim et al. (2025) (Section 3) and in our flow-based MIP (Section 4) as a function of the number of unique words $|\mathcal{W}|$. Panel titles give the ratio (ours/theirs) on the full corpus.

Runtime comparison between MIP solving of the encodings. Having shown that our flow-based MIP encoding is substantially more compact than the encoding of (Lim et al., 2025), we next evaluate the practical runtime of solving it with the state-of-the-art solver Gurobi. We run both encodings on different slices of the WikiText dataset, using the configuration described in Appendix C, and report the results in Table 2. The results show a clear gap in both runtime and memory scalability. On the 1%–15% slices, our encoding is 3.24×–7.12× faster, and from the 20% slice onward the previous MIP encoding runs out of memory, whereas our formulation terminates successfully and certifies a provably optimal compression value.

Table 2: Runtime of provably optimal tokenization on Wikitext using Gurobi with our flow-based MIP or the MIP of (Lim et al., 2025). OOM denotes out-of-memory (128GB RAM, 128 CPUs).

Runtime (sec) / Corpus (%)	1%	5%	10%	15%	20%	40%	60%	80%	100%
Our flow-Based MIP	**5.48**	**28.38**	**63.59**	**133.37**	**109.59**	**179.72**	**7840.17**	**50041.93**	**49582.94**
Previously suggested MIP	17.75	201.93	372.14	685.16	OOM	OOM	OOM	OOM	OOM
Improvement	3.24×	7.12×	5.85×	5.14×	–	–	–	–	–

6.2 Evaluating the Compression of the Lagrangian Relaxation

Certified lower and upper bounds across iterations. Beyond its scalability, a key advantage of the Lagrangian relaxation is that it provides *provable lower and upper bounds* on the optimal compression value. These bounds certify the approximation quality not only of the solution produced by the Lagrangian algorithm, but also of any heuristic baseline. Figure 5 illustrates this process across iterations: the Lagrangian upper bound decreases over time, eventually improving upon the compression values achieved by BPE and GreedTok, while approaching the provable lower bound. Thus, the method yields *certified approximation guarantees* for BPE, GreedTok, and the Lagrangian solution itself. We provide a more detailed analysis of these dynamics in Appendix I.

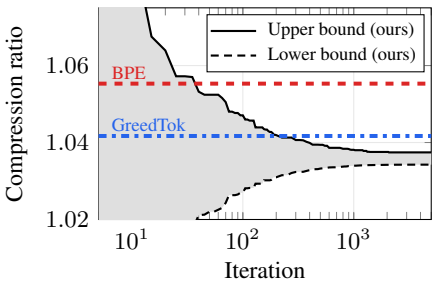

Figure 5: LRX bounds over iterations (UN, k=15K); shading: certified gap.

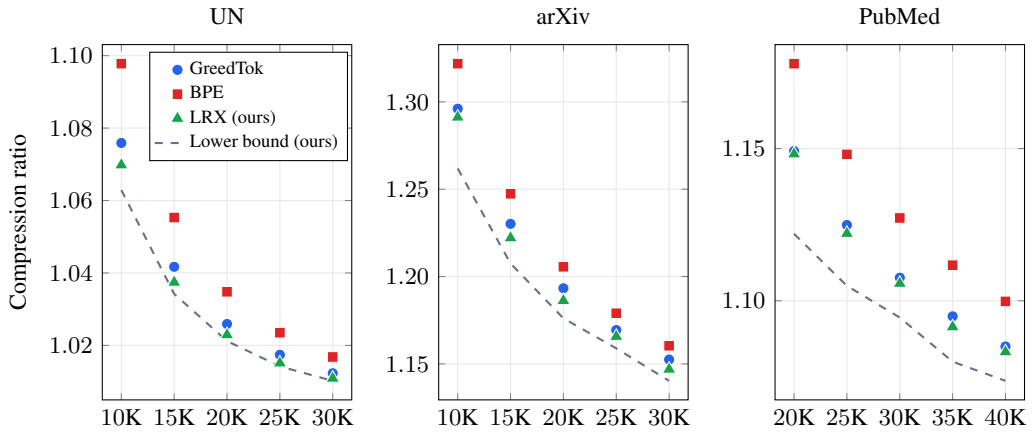

Figure 6: Compression ratio (lower is better) of our Lagrangian relaxation (LRX), its certified lower bound, BPE, and GreedTok, as a function of the token budget k.

Compression ratio across benchmarks and budgets. We next compare the final compression ratio of the Lagrangian relaxation (Section 5) against BPE (Gage, 1994) and GreedTok (Lim et al., 2025) on the three larger benchmarks and a range of budgets k (Figure 6). Our Lagrangian relaxation achieves substantially improved compression ratios across all benchmarks and all values of k. As expected, as k increases, the ratio approaches 1, and the gap over both GreedTok and BPE narrows.

6.3 Beyond Compression: Language Modeling and Generalization

Language modeling under the selected vocabularies. While our focus is provably optimal compression, where the previous sections show substantial improvements, we complement these results with evaluations of language-modeling quality, generalization, and online encoding. We train identical GPT-2-style models (22M parameters, five seeds; Karpathy, 2022) on arXiv, WikiText, and UN, tokenized with the $k = 30$K vocabularies of BPE and LRX, and compare held-out bits per byte (BPB), which, unlike perplexity, is comparable across tokenization schemes. LRX attains a lower mean BPB than BPE on all three corpora at every checkpoint, most clearly on WikiText (Table 3; details in Appendix D), so its compression gains do not come at the expense of modeling quality.

Table 3: Validation bits per byte (BPB; lower is better; mean $\pm$ std over five seeds) of identical 22M-parameter language models trained on each corpus with a BPE or LRX vocabulary ($k = 30$K), at 25/50/75/100% of training. Best mean is in bold.

Corpus	Tokenization scheme	25%	50%	75%	100%
arXiv	BPE-based	1.4584 $\pm$ .018	1.2921 $\pm$ .011	1.2176 $\pm$ .008	1.1691 $\pm$ .006
	LRX-based (ours)	**1.4476** $\pm$ .030	**1.2864** $\pm$ .019	**1.2143** $\pm$ .015	**1.1687** $\pm$ .012
WikiText	BPE-based	2.2672 $\pm$ .016	2.1800 $\pm$ .023	2.0960 $\pm$ .021	2.0390 $\pm$ .019
	LRX-based (ours)	**2.2006** $\pm$ .026	**2.1057** $\pm$ .021	**2.0291** $\pm$ .019	**1.9768** $\pm$ .018
UN	BPE-based	1.2893 $\pm$ .018	1.0840 $\pm$ .009	1.0053 $\pm$ .004	0.9647 $\pm$ .004
	LRX-based (ours)	**1.2755** $\pm$ .027	**1.0805** $\pm$ .012	**1.0028** $\pm$ .006	**0.9630** $\pm$ .004

Online encoding, generalization, and morphology. Our methods operate *offline*: they select the vocabulary, over which any online tokenizer can then segment text at inference time. Since optimal online segmentation for a fixed vocabulary can be computed efficiently by shortest-path dynamic programming (Schmidt et al., 2024), we pair our vocabularies with this scheme, and LRX remains ahead of BPE even when BPE uses it as well (Appendix G). The gains also extend to *held-out* text: a vocabulary selected on an arXiv training split compresses the unseen test split better than BPE (Appendix E), with the largest gains on low-frequency, rare, and unseen words, about 0.2 fewer tokens per word on words never seen in training (Appendix F). Finally, beyond compression, LRX also yields segmentations that are better aligned with morphological boundaries (Appendix H).

7 Related Work

Compression-based tokenization. BPE (Gage, 1994; Sennrich et al., 2016) is the dominant tokenizer, commonly viewed as a greedy compression procedure (Gallé, 2019). Its variants add stochastic merges, pruning, alternative encodings, or whitespace-spanning tokens (Provilkov et al., 2020; Schmidt et al., 2024; Chizhov et al., 2024; Schmidt et al., 2025; Liu et al., 2025; Xu et al., 2021). Unigram (Kudo, 2018) instead casts tokenization as probabilistic segmentation, but compresses less effectively (Schmidt et al., 2024). Closest to us, Lim et al. (2025) propose a MIP encoding and the greedy GreedTok heuristic; we instead obtain provably optimal solutions through a substantially smaller MIP, and certified bounds through a scalable Lagrangian relaxation. Concurrent work (Schmidt et al., 2026) solves an integer program restricted to segmentations induced by fixed split trees, whereas we optimize over *all* segmentations in the full substring graph.

Theoretical foundations of tokenization. A growing line of work studies optimal-compression tokenization as a formal computational problem. Whittington et al. (2025) prove NP-completeness for both direct and merge-based (i.e., "bottom-up") tokenization, and Lim et al. (2025) give an alternative NP-hardness proof for the direct formulation. Kozma & Voderholzer (2024) prove APX-completeness for the merge-based setting, and Kastreva et al. (2026) prove hardness over bounded alphabets. Other works study further structural and theoretical aspects of tokenizers (Gastaldi et al., 2025; Vieira et al., 2025). We complement this line by casting direct tokenization as a path-selection problem, which yields a practical framework for exact and provably approximate solutions.

String covering and optimization frameworks. Tokenization is also related to string covering (Mhaskar & Smyth, 2023; Iliopoulos et al., 1996) and dictionary compression (Kieffer & Yang, 2000; Charikar et al., 2005), and our methods build on standard combinatorial optimization tools: network-flow formulations (Ahuja & Orlin, 2002; Conforti et al., 2009), MIP (Wolsey, 2007; Conforti et al., 2014), and Lagrangian relaxation (Fisher, 1985; Geoffrion, 1974). More closely, Canzar et al. (2012) apply a graph-flow formulation with a Lagrangian relaxation for string covering. They minimize cover size, whereas we minimize corpus-level token count under a fixed vocabulary budget k.

8 Limitations and Future Work

Since optimal compression-based tokenization is NP-hard (Whittington et al., 2025; Lim et al., 2025; Kozma & Voderholzer, 2024; Kastreva et al., 2026), our exact approach remains bounded by MIP solver scalability. However, our flow-based formulation substantially extends the reach of exact solving, and continued MIP solver advances (Gurobi Optimization, LLC, 2026; Achterberg & Wunderling, 2013; Koch et al., 2022; Achterberg et al., 2020) will translate directly into larger provably optimal instances. Our Lagrangian relaxation scales much further but remains slower than lightweight heuristics such as BPE. In exchange, it achieves stronger compression and provides certified lower and upper bounds that quantify how far any tokenizer, including ours, is from optimal. Finally, compression is only one facet of tokenizer quality (Sennrich et al., 2016; Schmidt et al., 2024). Prior evidence that fewer tokens do not yield better models concerns reducing token counts at encoding time over a *fixed* vocabulary (Schmidt et al., 2024). In contrast, when the vocabulary itself is optimized, we find that language-modeling quality is preserved, and on some corpora even improved (Section 6.3). Future work could jointly optimize compression with other objectives, such as morphological consistency or predictive utility, while retaining provable guarantees.

9 Conclusion

We presented a new framework for compression-based tokenization, taking a step toward tractable *provably optimal* token selection at larger scales. Using a graph-theoretic view, we introduced a flow-based MIP formulation that substantially reduces the problem size compared to prior exact approaches, enabling modern solvers to obtain provably optimal tokenization for corpuses that were previously intractable. We further improve scalability via a Lagrangian relaxation that decomposes into parallel shortest-path subproblems while providing provable lower and upper bounds on the optimal compression value. Together, these results help bridge the gap between the theory of tokenization as a hard combinatorial optimization problem and its practical algorithmic realization, suggesting a broader optimization-theoretic path toward tokenizer design with provable guarantees.

AI USE STATEMENT

In this work, we used generative AI tools for implementing basic methods, such as helper functions within our pipeline, visualizations such as plotting figures, baselines or data loading. We have not used generative AI tools for developing theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, or interpreting results. Generating synthetic data sets, assisting with translation, cleaning and reformatting datasets, and supporting qualitative and thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for creating or editing software code, creating or modifying scientific figures or images, and editing the paper to improve readability (rephrasing and polishing). Three authors carefully reviewed, edited, and tested all generated code, and carefully reviewed and edited all generated text. We take full responsibility for all content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Tobias Achterberg and Roland Wunderling. Mixed Integer Programming: Analyzing 12 Years of Progress. In *Facets of combinatorial optimization: Festschrift for martin grötschel*, pp. 449–481. Springer, 2013.

Tobias Achterberg, Robert E Bixby, Zonghao Gu, Edward Rothberg, and Dieter Weninger. Presolve Reductions in Mixed Integer Programming. *INFORMS Journal on Computing*, 32(2):473–506, 2020.

Ravindra K Ahuja and James B Orlin. Combinatorial Algorithms for Inverse Network Flow Problems. *Networks: An International Journal*, 40(4):181–187, 2002.

Kathi Canese and Sarah Weis. Pubmed: the bibliographic database, the ncbi handbook. *Bethesda: National Center for Biotechnology Information (US)*, 2013.

Stefan Canzar, Tobias Marschall, Sven Rahmann, and Chris Schwiegelshohn. Solving the Minimum String Cover Problem. In *14th Annual Meeting on Algorithm Engineering and Experiments, ALENEX 2012*, pp. 75–83, 2012.

Moses Charikar, Eric Lehman, Ding Liu, Rina Panigrahy, Manoj Prabhakaran, Amit Sahai, and Abhi Shelat. The Smallest Grammar Problem. *IEEE Transactions on Information Theory*, 51(7): 2554–2576, 2005.

Pavel Chizhov, Catherine Arnett, Elizaveta Korotkova, and Ivan P Yamshchikov. BPE Gets Picky: Efficient Vocabulary Refinement During Tokenizer Training. In *Proc. Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 16587–16604, 2024.

Colin B Clement, Matthew Bierbaum, Kevin P O'Keeffe, and Alexander A Alemi. On the Use of arXiv as a Dataset. *arXiv preprint arXiv:1905.00075*, 2019.

Michele Conforti, Marco Di Summa, Friedrich Eisenbrand, and Laurence A Wolsey. Network formulations of mixed-integer programs. *Mathematics of Operations Research*, 34(1):194–209, 2009.

Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. Integer Programming Models. In *Integer Programming*, pp. 45–84. Springer, 2014.

Marshall L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science*, 27(1):1–18, 1981.

Marshall L Fisher. An Applications Oriented Guide to Lagrangian Relaxation. *Interfaces*, 15(2): 10–21, 1985.

Philip Gage. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38, 1994.

Matthias Gallé. Investigating the effectiveness of BPE: The power of shorter sequences. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 1375–1381, 2019.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*, 2020.

Juan Luis Gastaldi, John Terilla, Luca Malagutti, Brian DuSell, Tim Vieira, and Ryan Cotterell. The Foundations of Tokenization: Statistical and Computational Concerns. In *Proc. 13th Int. Conf. on Learning Representations (ICLR)*, 2025.

Arthur M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Study*, 2:82–114, 1974.

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2026. URL `https://www.gurobi.com`.

Elad Hazan and Sham Kakade. Revisiting the polyak step size. *arXiv preprint arXiv:1905.00313*, 2019.

Costas S Iliopoulos, Dennis WG Moore, and Kunsoo Park. Covering a String. *Algorithmica*, 16(3): 288–297, 1996.

Slava Jankin, Alexander Baturo, and Niheer Dasandi. Words to Unite Nations: The Complete United Nations General Debate Corpus, 1946–present. *Journal of Peace Research*, 62(4):1339–1351, 2025.

Jean Kaddour, Joshua Harris, Maximilian Mozes, Herbie Bradley, Roberta Raileanu, and Robert McHardy. Challenges and Applications of Large Language Models. *arXiv preprint arXiv:2307.10169*, 2023.

Richard M Karp. Reducibility Among Combinatorial Problems. In *50 Years of Integer Programming 1958-2008: from the Early Years to the State-of-the-Art*, pp. 219–241. Springer, 2009.

Andrej Karpathy. nanoGPT. `https://github.com/karpathy/nanoGPT`, 2022. Software repository.

Violeta Kastreva, Philip Whittington, Dennis Komm, and Tiago Pimentel. Tokenisation over Bounded Alphabets is Hard. *Proc. 14th Int. Conf. on Learning Representations (ICLR)*, 2026.

Leonid Genrikhovich Khachiyan. A Polynomial Algorithm in Linear Programming. In *Doklady Akademii Nauk*, volume 244, pp. 1093–1096. Russian Academy of Sciences, 1979.

John C Kieffer and En-Hui Yang. Grammar-Based Codes: A New Class of Universal Lossless Source Codes. *IEEE Transactions on Information Theory*, 46(3):737–754, 2000.

Thorsten Koch, Timo Berthold, Jaap Pedersen, and Charlie Vanaret. Progress in Mathematical Programming Solvers from 2001 to 2020. *EURO Journal on Computational Optimization*, 10: 100031, 2022.

László Kozma and Johannes Voderholzer. Theoretical Analysis of Byte-Pair Encoding. *arXiv preprint arXiv:2411.08671*, 2024.

Taku Kudo. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In *Proc. 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 66–75, 2018.

Claude Lemaréchal. Lagrangian relaxation. In *Computational combinatorial optimization: optimal or provably near-optimal solutions*. Springer, 2001.

Jia Peng Lim, Shawn Tan, Davin Choo, and Hady W Lauw. A Partition Cover Approach to Tokenization. In *Proc. 39th Conf. on Neural Information Processing Systems (NeurIPS)*, 2025.

Alisa Liu, Jonathan Hayase, Valentin Hofmann, Sewoong Oh, Noah A. Smith, and Yejin Choi. SuperBPE: Space travel for language models. In *2nd Conf. on Language Modeling (COLM)*, 2025.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer Sentinel Mixture Models. *arXiv preprint arXiv:1609.07843*, 2016.

Neerja Mhaskar and William F Smyth. String covering: A survey. *Fundamenta Informaticae*, 190, 2023.

George L. Nemhauser and Laurence A. Wolsey. *Integer and Combinatorial Optimization*. Wiley, 1988.

Boris Teodorovich Polyak. Minimization of Unsmooth Functionals. *USSR Computational Mathematics and Mathematical Physics*, 9(3):14–29, 1969.

Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. BPE-Dropout: Simple and Effective Subword Regularization. In *Proc. of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1882–1892, 2020.

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

Craig W Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. Tokenization is More than Compression. In *Proc. of the 2024 Conf. on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 678–702, 2024.

Craig W Schmidt, Varshini Reddy, Chris Tanner, and Yuval Pinter. Boundless Byte Pair Encoding: Breaking the Pre-tokenization Barrier. In *2nd Conf. on Language Modeling (COLM)*, 2025.

Craig W. Schmidt, Michael Krumdick, Adam Wiemerslage, Seth Ebner, Varshini Reddy, Yuval Pinter, and Chris Tanner. Tokenization with split trees. *arXiv preprint arXiv:2605.22705*, 2026.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proc. 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 1715–1725, 2016.

Omri Uzan, Craig W Schmidt, Chris Tanner, and Yuval Pinter. Greed is all you need: An evaluation of tokenizer inference methods. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 813–822, 2024.

Tim Vieira, Tianyu Liu, Clemente Pasti, Yahya Emara, Brian DuSell, Benjamin LeBrun, Mario Giulianelli, Juan Luis Gastaldi, Timothy J O'Donnell, and Ryan Cotterell. Language Models over Canonical Byte-Pair Encodings. In *Proc. 42nd Int. Conf. on Machine Learning (ICML)*, 2025.

Philip Whittington, Gregor Bachmann, and Tiago Pimentel. Tokenisation is NP-complete. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28133–28153, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.1365.

Laurence A Wolsey. Mixed Integer Programming. *Wiley Encyclopedia of Computer Science and Engineering*, pp. 1–10, 2007.

Jingjing Xu, Hao Zhou, Chun Gan, Zaixiang Zheng, and Lei Li. Vocabulary Learning via Optimal Transport for Neural Machine Translation. In *Proc. 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, pp. 7361–7373, 2021.

Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Tim Vieira, Mrinmaya Sachan, and Ryan Cotterell. A Formal Perspective on Byte-Pair Encoding. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 598–614, 2023.

Appendix

The appendix provides background, formal definitions and proofs referenced throughout the paper.

Appendix A provides additional background on mixed integer programing (MIP), lagrangian relaxation, and the MIP encoding mentioned in (Lim et al., 2025).
Appendix B contains the full proofs of the propositions mentioned within the main text.
Appendix C contains additional details on the datasets and algorithmic configurations.
Appendix D contains an evaluation of downstream language-modeling performance.
Appendix E contains an evaluation of generalization to held-out text.
Appendix F reports compression as a function of word frequency, including rare and unseen words.
Appendix G contains an evaluation under different online tokenization schemes.
Appendix H contains an evaluation of morphological alignment.
Appendix I contains ablation and supplementary results.

A Additional Background

Mixed-Integer Programming (MIP). A mixed-integer program (MIP) is an optimization problem in which the objective and constraints are linear, but some variables are restricted to take integral values (typically binary). Formally, a MIP can be written as:

$$\min_{\mathbf{x}\in\mathbb{R}^n,\ \mathbf{y}\in\{0,1\}^m} c^\top \mathbf{x} + d^\top \mathbf{y}$$

subject to

$$A\mathbf{x} + B\mathbf{y} \leq b.$$

where c, d are objective-cost vectors, A, B are constraint matrices, and b is the constraint vector.

MIPs provide a general framework for modeling combinatorial optimization problems by encoding discrete decisions through binary variables and structural constraints. While linear programs can be solved in polynomial time (Khachiyan, 1979), integer programming is NP-hard in general (Karp, 2009), which makes exact solving computationally challenging in the worst case. Modern solvers such as Gurobi (Gurobi Optimization, LLC, 2026) address this using branch-and-bound, cutting planes, and primal heuristics, often certifying global optimality via upper and lower bounds on the objective. In our setting, token selection and token usage naturally correspond to binary decisions, making MIP a suitable exact optimization framework for compression-based tokenization.

In practice, the scalability of MIP solving is strongly influenced by the size of the formulation, especially the number of binary variables and the number of constraints (Wolsey, 2007; Conforti et al., 2014). Binary variables encode discrete choices and therefore drive the size of the branch-and-bound search tree, whose worst-case size can grow exponentially in the number of integer variables. Constraints, in turn, determine the size and cost of the linear relaxations solved at each node, and can also affect the strength of the relaxation and the effectiveness of cutting planes and presolve procedures. Thus, even for the same underlying optimization problem, a more compact formulation can have a substantial practical impact on solver performance. This is central to our work: by replacing the prior token-based encoding with a flow-based formulation, we substantially reduce both the number of binary variables and the number of constraints, making exact optimization feasible at larger scales.

Lagrangian Relaxation. Lagrangian relaxation is a standard technique for improving the scalability of constrained optimization problems by dualizing difficult constraints into the objective with penalty multipliers. Given an optimization problem of the form

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{s.t.} \quad g_i(\mathbf{x}) \leq 0 \ \ \forall i,$$

its Lagrangian relaxation replaces these constraints with the objective

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \sum_i \lambda_i g_i(\mathbf{x}),$$

where $\lambda_i \geq 0$ are dual variables. For any fixed multiplier vector λ, optimizing $L(\mathbf{x}, \lambda)$ yields a lower bound on the original minimization problem. The resulting dual problem seeks the tightest such

bound by maximizing over λ. In practice, this often decomposes the original problem into simpler independent subproblems, substantially improving scalability while preserving provable bounds on optimality. In our formulation, dualizing the token-consistency constraints yields a shortest-path decomposition over word-specific DAGs, enabling efficient large-scale optimization.

Additional Details on the MIP Formulation of Lim et al. (2025). For completeness, we provide additional details on the MIP formulation introduced by (Lim et al., 2025), which we use as the baseline exact formulation in our comparison. We note that their formulation is presented as maximizing token *coverage*, but they also show that this representation is equivalent to compression-minimization under the partition formulation. Recall that given a corpus $\mathcal{C} = \langle \mathcal{W}, f \rangle$, a candidate token set $\mathcal{T}$, and a token budget $k \in \mathbb{N}$, the goal is to select at most k tokens that minimize the corpus compression objective:

$$\text{Comp}(\mathcal{C}, \mathcal{S}) = \sum_{w \in \mathcal{W}} f(w) \cdot \text{partition}(w, \mathcal{B}(\mathcal{C}) \cup \mathcal{S}).$$

The formulation of (Lim et al., 2025) is stated as maximizing the number of adjacent character merges induced by the selected token set. This objective is equivalent to minimizing the partition size. To see this, observe that for any word $w \in \mathcal{W}$, each merge of two adjacent symbols reduces the total number of resulting tokens by exactly one. Consequently, if a tokenization of w consists of r tokens, it induces exactly $|w| - r$ merges. Thus:

$$\text{partition}(w, \mathcal{B}(\mathcal{C}) \cup \mathcal{S}) = |w| - \text{coverage}(w, \mathcal{S}),$$

where $\text{coverage}(w, \mathcal{S})$ denotes the number of adjacent pairs covered by selected tokens.

Summing over the corpus gives:

$$\text{Comp}(\mathcal{C}, \mathcal{S}) = \sum_{w \in \mathcal{W}} f(w) \cdot |w| - \sum_{w \in \mathcal{W}} f(w) \cdot \text{coverage}(w, \mathcal{S}).$$

Since the first term is constant with respect to $\mathcal{S}$, minimizing compression is equivalent to maximizing coverage. Hence the MIP formulation described in (Lim et al., 2025) and in Section 3 can equivalently be formulated as:

$$\max \sum_{w \in \mathcal{W}} f(w) \cdot \left(\sum_{i=0}^{|w|-2} \mathbf{m}_{i,i+1}^{w} \right). \tag{4}$$

subject to

- $\sum_{t \in \mathcal{T} \setminus \mathcal{B}(\mathcal{C})} \mathbf{x}_t \leq k,$
- $\mathbf{x}_t \geq \mathbf{m}_{i,i+1}^{w,t}$ $\quad \forall w \in \mathcal{W}, \forall t \in \mathcal{T}, \forall 0 \leq i < |w| - 1, \text{ s.t. } (w_i, w_{i+1}) \subseteq t,$
- $\sum_{t \in \mathcal{T}} \mathbf{m}_{i,i+1}^{w,t} \geq \mathbf{m}_{i,i+1}^{w}$ $\quad \forall w \in \mathcal{W}, \forall 0 \leq i < |w| - 1,$
- $\sum_{t \in \mathcal{T}} \mathbf{m}_{i,i+1}^{w,t} \leq 1$ $\quad \forall w \in \mathcal{W}, \forall 0 \leq i < |w| - 1,$
- $\mathbf{m}_{i,i+1}^{w,t} = \mathbf{m}_{i+1,i+2}^{w,t}$ $\quad \forall w \in \mathcal{W}, \forall t \in \mathcal{T}, \forall 0 \leq i < |w| - 2 \text{ s.t. } (w_i, w_{i+1}, w_{i+2}) \subseteq t,$
- $\sum_{t \in \mathcal{T}} \mathbf{m}_{s-1,s}^{w,t} \leq 1 - \mathbf{m}_{s,s+1}^{w,t}$ $\quad \forall w \in \mathcal{W}, \forall t \in \mathcal{T}, \forall 1 \leq s < |w| - 1, \text{ s.t. } t \text{ starts at } w_s,$
- $\sum_{t \in \mathcal{T}} \mathbf{m}_{e,e+1}^{w,t} \leq 1 - \mathbf{m}_{e-1,e}^{w,t}$ $\quad \forall w \in \mathcal{W}, \forall t \in \mathcal{T}, \forall 1 \leq e < |w| - 1, \text{ s.t. } t \text{ ends at } w_e.$

Constraint interpretation. Here, we explain the meaning of each constraint in more detail. The constraint $\sum_{t \in \mathcal{T} \setminus \mathcal{B}(\mathcal{C})} \mathbf{x}_t \leq k$ enforces the vocabulary budget. The constraint $\mathbf{x}_t \geq \mathbf{m}_{i,i+1}^{w,t}$ ensures that a token may cover a span only if it is selected into the vocabulary. The constraint $\sum_{t \in \mathcal{T}} \mathbf{m}_{i,i+1}^{w,t} \geq \mathbf{m}_{i,i+1}^{w}$ links token-level coverage to the aggregate merge variable, ensuring that a merge is counted only if some selected token realizes it. The constraint $\sum_{t \in \mathcal{T}} \mathbf{m}_{i,i+1}^{w,t} \leq 1$ enforces unique coverage, meaning that each adjacent pair is covered by at most one token. The constraint $\mathbf{m}_{i,i+1}^{w,t} = \mathbf{m}_{i+1,i+2}^{w,t}$ enforces contiguity: if a token covers one adjacent pair inside its span, it must cover all adjacent

pairs inside that span. The constraints $\sum_{t\in\mathcal{T}} \mathbf{m}^{w,t}_{s-1,s} \leq 1 - \mathbf{m}^{w,t}_{s,s+1}$ and $\sum_{t\in\mathcal{T}} \mathbf{m}^{w,t}_{e,e+1} \leq 1 - \mathbf{m}^{w,t}_{e-1,e}$ enforce boundary consistency and prevent overlapping token spans.

The GreedTok algorithm. Building on this formulation, (Lim et al., 2025) propose GreedTok, a greedy token-selection heuristic inspired by weighted set cover. At each iteration, the algorithm selects the token with the largest marginal gain in corpus coverage. While this heuristic scales substantially better than solving the exact MIP, it does not directly optimize the global compression objective and does not provide a provable approximation guarantee for optimal compression.

B Full Proofs

Proposition 1. *The number of binary variables defined by the MIP suggested by (Lim et al., 2025) is* $\Theta(|\mathcal{T}| + \sum_{w\in\mathcal{W}} |w| \cdot |\mathcal{T}|) = \Theta(\sum_{w\in\mathcal{W}} |w| \cdot |\mathcal{T}|)$. *This bound also applies to the number of constraints.*

Proof. The count follows by summing the variables of each type. First, there is one binary variable $\mathbf{x}_t$ for each candidate token $t \in \mathcal{T}$, contributing $|\mathcal{T}|$ variables. Second, for each word $w \in \mathcal{W}$ and each adjacent pair of positions, there is one aggregate merge variable $\mathbf{m}^{w}_{i,i+1}$, contributing

$$\sum_{w\in\mathcal{W}} (|w| - 1) = \Theta(\sum_{w\in\mathcal{W}} |w|)$$

variables. Finally, for each word, each adjacent position, and each candidate token, there is a token-specific merge variable $\mathbf{m}^{w,t}_{i,i+1}$, contributing

$$\sum_{w\in\mathcal{W}} (|w| - 1) \cdot |\mathcal{T}| = \Theta(\sum_{w\in\mathcal{W}} |w| \cdot |\mathcal{T}|)$$

variables. This last term dominates, yielding the overall bound of:

$$|\mathcal{T}| + \sum_{w\in\mathcal{W}} (|w| - 1) + \sum_{w\in\mathcal{W}} (|w| - 1) \cdot |\mathcal{T}| = \Theta(\sum_{w\in\mathcal{W}} |w| \cdot |\mathcal{T}|).$$

For the constraints, the budget constraint contributes one constraint. The token-selection constraints

$$\mathbf{x}_t \geq \mathbf{m}^{w,t}_{i,i+1}$$

are indexed by words, adjacent positions, and candidate tokens, and therefore contribute

$$\Theta\left(\sum_{w\in\mathcal{W}} (|w| - 1) \cdot |\mathcal{T}|\right) = \Theta\left(\sum_{w\in\mathcal{W}} |w| \cdot |\mathcal{T}|\right)$$

constraints. The remaining constraint families contribute no more than this order: the merge-linking constraints are indexed only by words and positions, while the contiguity and boundary constraints are indexed by words, positions, and tokens. Hence, the overall number of constraints is

$$\Theta\left(\sum_{w\in\mathcal{W}} |w| \cdot |\mathcal{T}|\right).$$

□

Proposition 2. *Given a corpus* $\langle \mathcal{W}, f\rangle$, *a token budget* $k \in \mathbb{N}$, *a graph* $G = \langle V, E\rangle$ *representing* $\mathcal{C}$, *and a set of candidate tokens* $\mathcal{T} \subseteq \Sigma^+$, *then solving the encoding described in Equation 2 such that all* $\boldsymbol{x}$ *variables are chosen as indicators for the candidate tokens in* $\mathcal{T}$ *and all* $\boldsymbol{e}$ *variables are chosen as indicator variables for the edges in* E, *obtains an optimal compression solution* $\mathit{Comp}^*_k(\mathcal{C}, \mathcal{T})$.

Proof. We prove that the feasible solutions of the flow-based MIP are in one-to-one correspondence with choices of a vocabulary together with valid tokenizations of the corpus, and that the objective values coincide with the compression objective.

Let $(\mathbf{x}, \mathbf{e})$ be any feasible solution of the MIP, and define $\mathcal{S}_{\mathbf{x}} := \{t \in \mathcal{T} \setminus \mathcal{B}(\mathcal{C}) : \mathbf{x}_t = 1\}$. By the budget constraint, $|\mathcal{S}_{\mathbf{x}}| \leq k$, and by the singleton constraints all singleton tokens are available. Fix a word $w \in \mathcal{W}$. The flow constraints force the selected edges with $\mathbf{e}^w_{i,j} = 1$ to form a unit v^w_0-to-$v^w_{|w|}$ path in the DAG G_w. Since the graph is acyclic and all variables are binary, this path can be written as

$$v^w_{i_0}, v^w_{i_1}, \ldots, v^w_{i_r}, \qquad 0 = i_0 < i_1 < \cdots < i_r = |w|.$$

Each selected edge $(v^w_{i_\ell}, v^w_{i_{\ell+1}})$ corresponds to the substring $w[i_\ell : i_{\ell+1}]$. Moreover, the consistency constraint $\mathbf{e}^w_{i_\ell, i_{\ell+1}} \leq \mathbf{x}_{w[i_\ell:i_{\ell+1}]}$ implies that this substring is a selected token, or a required singleton. Hence the path induces a valid partition of w using tokens from $\mathcal{B} \cup \mathcal{S}_{\mathbf{x}}$. The number of tokens in this partition is exactly the number of selected edges in the path, i.e.,

$$\sum_{0 \leq i < j \leq |w|} \mathbf{e}^w_{i,j}.$$

Therefore the MIP objective equals the total number of tokens used by the induced tokenizations, weighted by word frequency.

Conversely, let $\mathcal{S} \subseteq \mathcal{T} \setminus \mathcal{B}(\mathcal{C})$ be any feasible vocabulary with $|\mathcal{S}| \leq k$. Set $\mathbf{x}_t = 1$ iff $t \in \mathcal{S} \cup \mathcal{B}$, and $\mathbf{x}_t = 0$ otherwise. For each word $w \in \mathcal{W}$, choose an optimal partition of w into tokens from $\mathcal{B} \cup \mathcal{S}$:

$$w = t_1 t_2 \cdots t_r.$$

This partition defines a path from v^w_0 to $v^w_{|w|}$ whose edges correspond to the consecutive spans of $t_1, \ldots, t_r$. Set $\mathbf{e}^w_{i,j} = 1$ exactly for the edges on this path, and set all other edge variables to 0. The resulting assignment satisfies the flow constraints and the consistency constraints, since every used edge corresponds to a token in $\mathcal{B} \cup \mathcal{S}$.

Thus every feasible MIP solution induces a feasible vocabulary and tokenization with the same objective value, and every feasible vocabulary and tokenization induces a feasible MIP solution with the same objective value. Minimizing the MIP objective is therefore exactly equivalent to minimizing $\mathrm{Comp}(\mathcal{C}, \mathcal{S})$ over all vocabularies $\mathcal{S} \subseteq \mathcal{T} \setminus \mathcal{B}(\mathcal{C})$ with $|\mathcal{S}| \leq k$. Hence the optimal value of the MIP is $\mathrm{Comp}^*_k(\mathcal{C}, \mathcal{T})$. □

Proposition 3. *The number of binary variables defined by the MIP described in Section 4 is* $\Theta(|\mathcal{T}| + \sum_{w \in \mathcal{W}} |w|^2) = \Theta(\sum_{w \in \mathcal{W}} |w|^2)$. *This bound also applies to the number of constraints.*

Proof. The count is obtained by summing the variables of each type. First, there is one binary variable $\mathbf{x}_t$ for every candidate token $t \in \mathcal{T}$. In addition, there is one variable $\mathbf{e}^w_{i,j}$ for every edge $e^w_{i,j}$ in the graph G representing the corpus $\mathcal{C}$.

For each word $w \in \mathcal{W}$, the corresponding graph is a complete DAG over $|w| + 1$ vertices, and therefore contains

$$\frac{|w| \cdot (|w| + 1)}{2}$$

edges. Hence, the total number of $\mathbf{e}^w_{i,j}$ variables is

$$\sum_{w \in \mathcal{W}} \frac{|w| \cdot (|w| + 1)}{2} = \Theta\left(\sum_{w \in \mathcal{W}} |w|^2\right).$$

The number of candidate tokens does not dominate this term. Indeed, $|\mathcal{T}|$ is at most the number of possible substrings in the corpus, and in our experimental setup it is exactly this set. Thus,

$$|\mathcal{T}| \leq \sum_{w \in \mathcal{W}} \frac{|w| \cdot (|w| + 1)}{2} \leq \sum_{w \in \mathcal{W}} |w|^2.$$

Therefore, the total number of binary variables in the encoding is

$$|\mathcal{T}| + \sum_{w \in \mathcal{W}} \frac{|w| \cdot (|w| + 1)}{2} = \Theta\left(|\mathcal{T}| + \sum_{w \in \mathcal{W}} |w|^2\right) = \Theta\left(\sum_{w \in \mathcal{W}} |w|^2\right).$$

We now count the constraints. The budget constraint contributes one constraint, and the singleton constraints contribute $|\mathcal{B}|$ constraints. The consistency constraints $\mathbf{e}^w_{i,j} \leq \mathbf{x}_{w[i:j]}$ contribute one constraint for every edge of the corpus graph, namely

$$\sum_{w\in\mathcal{W}} \frac{|w|\cdot(|w|+1)}{2} = \Theta\left(\sum_{w\in\mathcal{W}} |w|^2\right).$$

The source constraints contribute one constraint per word, and the flow-conservation constraints contribute at most one constraint per internal vertex, hence

$$\Theta(|\mathcal{W}|) + \Theta\left(\sum_{w\in\mathcal{W}} |w|\right)$$

additional constraints. Therefore, the total number of constraints is

$$1 + |\mathcal{B}| + \sum_{w\in\mathcal{W}} \frac{|w|\cdot(|w|+1)}{2} + \Theta\left(\sum_{w\in\mathcal{W}} |w|\right) = \Theta\left(\sum_{w\in\mathcal{W}} |w|^2\right),$$

where the quadratic edge-consistency constraints dominate. □

Proposition 4. *Given a corpus $\mathcal{C}$, a graph $G = \langle V, E\rangle$ representing it, and a set of candidate tokens $\mathcal{T}$, then for every multiplier function $\lambda : E \to \mathbb{R}_{\geq 0}$, the value $q(\lambda)$ is a lower bound on the optimal value of the original flow-based MIP in Equation 2, i.e., $q(\lambda) \leq \mathit{Comp}^*_k(\mathcal{C}, \mathcal{T})$.*

Proof. Fix any multiplier function $\lambda : E \to \mathbb{R}_{\geq 0}$, and let $q(\lambda)$ denote the optimal value of the relaxed problem. Let $(\mathbf{x}, \mathbf{e})$ be any feasible solution of the original flow-based MIP. Since $(\mathbf{x}, \mathbf{e})$ satisfies every consistency constraint

$$\mathbf{e}^w_{i,j} \leq \mathbf{x}_{w[i:j]},$$

we have

$$\mathbf{e}^w_{i,j} - \mathbf{x}_{w[i:j]} \leq 0$$

for every edge $e^w_{i,j} \in E$. Because $\lambda(e^w_{i,j}) \geq 0$, it follows that

$$\lambda(e^w_{i,j})\big(\mathbf{e}^w_{i,j} - \mathbf{x}_{w[i:j]}\big) \leq 0.$$

Therefore, for every feasible solution of the original MIP, the Lagrangian objective is at most the original MIP objective:

$$L(\mathbf{x}, \mathbf{e}, \lambda) \leq \sum_{w\in\mathcal{W}} f(w) \sum_{0\leq i<j\leq |w|} \mathbf{e}^w_{i,j}.$$

Moreover, the relaxed feasible region contains the original feasible region, since it is obtained by removing the consistency constraints. Hence, if $(\mathbf{x}^*, \mathbf{e}^*)$ is an optimal solution of the original MIP, then

$$q(\lambda) = \min_{(\mathbf{x},\mathbf{e})\in\mathcal{F}_{\mathrm{rel}}} L(\mathbf{x}, \mathbf{e}, \lambda) \leq L(\mathbf{x}^*, \mathbf{e}^*, \lambda) \leq \mathrm{Comp}^*_k(\mathcal{C}, \mathcal{T}).$$

Thus $q(\lambda)$ is a valid lower bound on the optimal value of the original flow-based MIP. □

Proposition 5. *Given a corpus $\mathcal{C}$, a graph $G = \langle V, E\rangle$ representing it, and a set of candidate tokens $\mathcal{T}$, then for every multiplier function $\lambda : E \to \mathbb{R}_{\geq 0}$, the assignments to **e** and **x** computed in lines 3–6 of Algorithm 1 attain the optimal value LB of the relaxed problem in Equation 3.*

Proof. Fix $\lambda : E \to \mathbb{R}_{\geq 0}$. Rewriting the relaxed objective gives

$$\sum_{w\in\mathcal{W}} \sum_{e^w_{i,j}\in E_w} \big(f(w) + \lambda(e^w_{i,j})\big)\mathbf{e}^w_{i,j} - \sum_{t\in\mathcal{T}} \mathbf{x}_t \sum_{e^w_{i,j}\in E : w[i:j]=t} \lambda(e^w_{i,j}).$$

The first term depends only on the edge variables **e**, while the second term depends only on the token variables **x**. Therefore, after the consistency constraints have been relaxed, the minimization over **e** and the minimization over **x** separate, except for the remaining budget and singleton constraints on **x**.

For the edge variables, fix a word w. The remaining flow constraints require the selected edges in G_w to form one unit path from the source v_0^w to the sink $v_{|w|}^w$. The contribution of such a path is the sum of its edge weights

$$c_\lambda(e_{i,j}^w) := f(w) + \lambda(e_{i,j}^w).$$

Thus, minimizing over the edge variables for this word is exactly the shortest-path problem in the DAG G_w with edge weights c_λ. Since the word graphs are disjoint, the optimal assignment to all edge variables is obtained by independently selecting a shortest path in each G_w.

For the token variables, define the aggregate multiplier score

$$s_\lambda(t) := \sum_{e_{i,j}^w \in E : w[i:j]=t} \lambda(e_{i,j}^w).$$

The part of the objective involving $\mathbf{x}$ is

$$-\sum_{t\in\mathcal{T}} s_\lambda(t)\mathbf{x}_t.$$

Subject to the budget constraint and the requirement that all singleton tokens are selected, this is minimized by selecting, in addition to the mandatory singleton tokens, the remaining tokens with the largest values of $s_\lambda(t)$, up to the budget. This is exactly the assignment performed in Algorithm 1.

Since the algorithm independently minimizes the $\mathbf{e}$-dependent part and the $\mathbf{x}$-dependent part of the relaxed objective, the assignment computed in lines 3–6 attains the optimal value of the relaxed problem. This value is therefore the lower bound $\mathrm{LB} = q(\lambda)$. □

Proposition 6. *At every iteration of Algorithm 1, LB is the optimal value of the relaxed problem for the current multiplier function and therefore a valid lower bound, while UB is the value of a feasible solution to the original flow-based MIP and therefore a valid upper bound. Consequently, $LB^* \le Comp_k^*(\mathcal{C},\mathcal{T}) \le UB^*$, and the algorithm maintains a certified optimality gap $UB^* - LB^*$.*

Proof. Fix an iteration of Algorithm 1, and let λ be the multiplier function at the beginning of that iteration. By Proposition 5, the value LB computed in the iteration is exactly the optimal value $q(\lambda)$ of the relaxed problem. By Proposition 4, $q(\lambda)$ is a lower bound on the optimal value of the original flow-based MIP. Therefore LB is a valid lower bound. Since LB^* is defined as the maximum lower bound encountered so far, LB^* is also a valid lower bound.

It remains to show that UB is a valid upper bound. The algorithm constructs UB from a feasible solution of the original MIP: the selected vocabulary satisfies the budget constraint and contains the required singleton tokens, and for each word w, the algorithm computes a source-to-sink path using only edges whose corresponding tokens are selected. Therefore the flow constraints are satisfied, and every selected edge satisfies the consistency constraint

$$\mathbf{e}_{i,j}^w \le \mathbf{x}_{w[i:j]}.$$

Hence the resulting assignment is feasible for the original flow-based MIP. Since the original problem is a minimization problem, the objective value of any feasible solution is an upper bound on the optimal value. Thus UB is a valid upper bound.

Since UB^* is the smallest such upper bound found so far, combining the two inequalities gives

$$\mathrm{LB}^* \le \mathrm{Comp}_k^*(\mathcal{C},\mathcal{T}) \le \mathrm{UB}^*.$$

Consequently, $\mathrm{UB}^* - \mathrm{LB}^*$ is a certified optimality gap. □

C Datasets and Algorithmic Configuration

In this section, we provide additional details on the datasets and technical specifications of our algorithms for reproducibility.

C.1 BENCHMARKS

In this subsection, we provide additional details on the benchmarks used throughout this work.

Wikitext. The WikiText corpus (Merity et al., 2016) is a large-scale language modeling benchmark introduced by Salesforce. We use WikiText-2 (raw-v1), a standard language modeling benchmark derived from high-quality, verified Wikipedia articles. The corpus contains roughly 2 million tokens and preserves the original raw text, including punctuation, casing, and formatting, without the aggressive normalization used in earlier benchmarks such as the Penn Treebank. Its diverse encyclopedic content and relatively rich vocabulary make it a useful benchmark for evaluating tokenization quality and compression performance under realistic natural language distributions.

arXiv. The arXiv corpus (Clement et al., 2019) consists of abstracts from scientific publications and preprints collected from the popular e-Print archive. It is released under a CC0 (Public Domain) license.

UN. The UN corpus (Jankin et al., 2025) consists of statements delivered by member states during the General Debate on major geopolitical issues between 1946 and 2022. The corpus is released under a Creative Commons CC0 (public domain) license. We use the same preprocessing of this dataset into words and frequencies as used in (Lim et al., 2025).

PubMed. The PubMed corpus (Canese & Weis, 2013) is a repository of scientific publications and preprints, primarily in health, medicine, biotechnology, and pharmaceuticals. Here, too, we use the same preprocessing of this dataset into words and frequencies as used in (Lim et al., 2025), which uses the Non-Commercial subset, consisting of articles licensed under CC BY-NC, CC BY-NC-SA, and CC BY-NC-ND, as well as under preprocessing done by removing citations and headers.

C.2 OPTIMIZATION AND CONFIGURATION DETAILS

Hardware. All experiments run on a Debian GNU/Linux x86_64 machine with Intel Xeon Gold 6248 CPUs.

Lagrangian Relaxation Algorithm. We parallelize the computation over 128 cores by decomposing the original DAG into independent sub-DAGs. This yields effective parallelism, as each subproblem requires approximately a $(\frac{1}{\rho})$ fraction of the memory required by the full problem, where $\rho \in \mathbb{N}$ is the number of processors.

For the Lagrangian relaxation algorithm (Algorithm 1), the pseudocode in the main paper uses a single step-size parameter μ for simplicity. In practice, however, to improve convergence, we follow the common convention of decaying μ over iterations. Specifically, all experiments use an initial value of $\mu_0 = 3.0$ and a decay factor $\mu_d = 0.9$, applied whenever the lower bound (LB) does not improve for *patience* $= 30$ iterations. We use this same fixed hyperparameter setting across all benchmarks, selected via a grid search over $\mu_d \in \{0.7, 0.75, 0.85, 0.9\}$ and *patience* $\in \{20, 30\}$, with $\mu_0 = 3.0$ fixed throughout.

MIP solving with Gurobi. We solve the MIP formulation using Gurobi (Gurobi Optimization, LLC, 2026). In addition to solving it directly, we also consider a variant initialized with the incumbent solution produced by GreedTok (Lim et al., 2025), and report the better of the two runs (with or without an incumbent), following a simple portfolio strategy. In all experiments, Gurobi is configured to use the maximum available number of threads.

D LANGUAGE MODELING QUALITY WITH THE SELECTED VOCABULARIES

To test whether the compression gains of our Lagrangian relaxation (LRX, Section 5) affect language modeling quality, we train identical language models from scratch on the same text tokenized by both BPE (which is a standard tokenization scheme used for pretraining) and LRX (our Lagrangian relaxation method) and compare how well the resulting models predict held-out documents.

Experimental setup. We use the arXiv, WikiText, and UN corpora of Table 1 with a token budget of $k = 30\text{K}$ for every method. As throughout, each method's vocabulary consists of the 256 singleton bytes plus at most k learned tokens, so all models have matched capacity. For each tokenizer we train the same 22M-parameter decoder-only Transformer following the GPT-2 architecture (Radford

et al., 2019), in the small configuration distributed with nanoGPT (Karpathy, 2022): 6 layers, 6 attention heads, embedding dimension 384, and context length 256, trained with AdamW (Loshchilov & Hutter, 2017) (learning rate 5×10^{-4}, batch size 32) for one epoch over the training documents, with 10% of the documents held out for validation. All runs use identical hardware (a single NVIDIA L40S GPU with 48 GB of memory), and every configuration is repeated over 5 random seeds. The document split is fixed across seeds, so every seed is evaluated on the same held-out text and only model initialization and batch order vary.

Evaluation metric. Because each tokenizer segments the same text into a different number of tokens, per-token perplexity is not comparable across tokenizers: a vocabulary that packs more bytes into each token faces a harder per-token prediction problem. Following standard practice (Gao et al., 2020), we report *bits per byte* (BPB), which normalizes the total negative log-likelihood of the held-out text by its size in UTF-8 bytes rather than by its number of tokens: $\mathrm{BPB} = \mathrm{NLL}/\left(\ln(2) \cdot n_{\mathrm{bytes}}\right)$, where NLL is measured in nats. BPB is directly comparable across tokenizers, and lower is better. We report validation BPB (mean $\pm$ standard deviation over the 5 seeds) at 25%, 50%, 75%, and 100% of the training epoch.

Results. Table 3 in the main text reports the results. LRX attains the lowest mean BPB in all twelve dataset/checkpoint settings. On arXiv and UN the two tokenization schemes are close, with LRX still ahead of BPE at every checkpoint; on WikiText the gap is significantly larger and persists throughout training, with LRX improving over BPE by roughly 0.06 BPB at the end of the epoch. Overall, the substantially better compression of LRX, which directly translates into a smaller memory footprint and lower storage and bandwidth cost, does not come at the expense of language-modeling quality, and we find that in some instances can even improve it.

E Held-Out Generalization under DP Online Encoding

Given a fixed vocabulary, optimal-compression tokenization can be found efficiently using shortest-path dynamic programming (DP), as explored, for example, by Schmidt et al. (2024). Our main contribution is different and instead addresses the substantially computationally harder, NP-hard problem of finding the optimally compressed vocabulary itself (Lim et al., 2025; Whittington et al., 2025; Kastreva et al., 2026), which is why our experiments mainly focus on this offline stage. Once a vocabulary has been constructed with our method, the choice of online-tokenization scheme is left to the user. In particular, DP online-encoding is a natural fit, since it directly matches our objective.

We ran an additional experiment measuring the held-out compression ratio of our Lagrangian relaxation method (LRX) under DP online-encoding, comparing it against BPE with its standard merge-rank encoding and against BPE combined with DP. We used the DP algorithm of Schmidt et al. (2024), implemented in Python, with ties broken by the longest-token rule ("PathPieceL") of Schmidt et al. (2024). The arXiv dataset was split 90/10, 80/20, and 70/30 into train–test partitions; on each training partition, we trained both BPE and our Lagrangian method with vocabulary size $k = 30{,}000$, and evaluated compression on the held-out portion. The results are reported in Table 4.

Our results show that the Lagrangian relaxation approach generalizes to previously unseen text (the test set), achieving consistent gains in compression ratio on both the training and held-out sets in all settings.

F Rare-String Behavior: Compression as a Function of Word Frequency

To complement the results on the compression ratio, we also examine how the compression gains are distributed across words of different frequencies. In particular, we analyze whether they extend to rare and unseen words, which under a fixed budget are typically encoded as several shorter tokens. Following the experimental setup of Appendix E, we split the arXiv dataset into 90/10, 80/20, and 70/30 train–test partitions and measured the compression ratio achieved by each method on the held-out partition across five word-frequency groups, defined by the number of occurrences of each word in the training partition: high-frequency words occurring at least 1,000 times, medium-frequency words occurring 100–999 times, low-frequency words occurring 10–99 times, rare words occurring

Table 4: Held-out generalization on arXiv ($k = 30{,}000$). Compression ratio (tokens per word occurrence; lower is better) on the training partition and on the held-out partition for three train–test splits. Best value per split in bold.

Split (train/test)	Method	Train	Test (held-out)
90/10	BPE merge-rank	1.1603	1.1621
	BPE-DP	1.1546	1.1562
	LRX-DP (ours)	**1.1468**	**1.1488**
80/20	BPE merge-rank	1.1601	1.1620
	BPE-DP	1.1545	1.1562
	LRX-DP (ours)	**1.1461**	**1.1482**
70/30	BPE merge-rank	1.1600	1.1620
	BPE-DP	1.1543	1.1562
	LRX-DP (ours)	**1.1461**	**1.1483**

1–9 times, and unseen words with zero occurrences in the training split. The results are reported in Table 5.

The results show that our method achieves a better compression ratio than all competing approaches, with even larger gains for lower-frequency strings. For high-frequency words, all methods require close to one token per word; for unseen words, the gap between LRX-DP and BPE with merge-rank encoding is about 0.2 tokens per word.

Table 5: Compression ratio (lower is better) on the held-out partition of arXiv ($k = 30{,}000$) for three train–test splits, grouped by the number of occurrences of each word in the training partition. The "All" column is the overall held-out compression ratio. Best value per split and group in bold.

Split	Method	High ($\geq$1000)	Mid (100–999)	Low (10–99)	Rare (1–9)	Unseen (0)	All
90/10	BPE merge-rank	1.0006	1.0227	2.1552	2.7907	4.0186	1.1621
	BPE-DP	1.0006	1.0227	2.1203	2.7099	3.9070	1.1562
	LRX-DP (ours)	**1.0003**	**1.0100**	**2.0498**	**2.6815**	**3.8194**	**1.1488**
80/20	BPE merge-rank	1.0006	1.0106	2.0905	2.7627	3.9908	1.1620
	BPE-DP	1.0006	1.0106	2.0601	2.6842	3.8814	1.1562
	LRX-DP (ours)	**1.0003**	**1.0059**	**1.9668**	**2.6509**	**3.7912**	**1.1482**
70/30	BPE merge-rank	1.0006	1.0097	1.9866	2.7244	3.9461	1.1620
	BPE-DP	1.0006	1.0097	1.9596	2.6482	3.8381	1.1562
	LRX-DP (ours)	**1.0003**	**1.0052**	**1.8679**	**2.6216**	**3.7531**	**1.1483**

G Comparison against BPE with DP Encoding

We adopted the evaluation protocol of prior work on offline vocabulary compression, e.g., GreedTok (Lim et al., 2025), which compares against merge-rank BPE rather than DP-based decoding; this is the setting reported throughout the paper. However, the comparison against BPE combined with the DP online-encoding of Appendix E is also of interest, and we therefore repeated all comparisons of Figure 6 using BPE+DP in addition to regular BPE. The results are reported in Table 6; our method continues to achieve a better (i.e., lower) compression ratio.

These results highlight that the overall compression ratio depends critically on compressing the vocabulary itself, not just the online encoding given a fixed vocabulary.

Table 6: Compression ratio (lower is better) of BPE with its standard merge-rank encoding, BPE with DP encoding (BPE+DP), and LRX, for the datasets and vocabulary sizes k of Figure 6. BPE and LRX values match Figure 6; BPE+DP uses the BPE vocabulary of an earlier run. Best value per row in bold.

Dataset	k	BPE	BPE+DP	LRX (ours)
arXiv	10K	1.3219	1.3125	**1.2912**
	15K	1.2474	1.2397	**1.2221**
	20K	1.2056	1.1988	**1.1862**
	25K	1.1790	1.1729	**1.1657**
	30K	1.1604	1.1547	**1.1469**
UN	10K	1.0978	1.0942	**1.0698**
	15K	1.0553	1.0530	**1.0374**
	20K	1.0348	1.0332	**1.0229**
	25K	1.0235	1.0225	**1.0151**
	30K	1.0168	1.0160	**1.0109**
PubMed	20K	1.1779	1.1683	**1.1482**
	25K	1.1481	1.1397	**1.1221**
	30K	1.1272	1.1198	**1.1057**
	35K	1.1117	1.1051	**1.0914**
	40K	1.0998	1.0938	**1.0832**

H Morphological Alignment

We ran an additional experiment measuring morphological alignment following the protocol of Uzan et al. (2024), which reports the macro-averaged F1 score over a unified set of seven common morphological segmentation benchmarks. We train BPE, GreedTok, and our method (LRX) on the English morphology dataset across a range of vocabulary sizes k, and evaluate each setting. The results are reported in Table 7.

Table 7: Morphological alignment, measured as the macro-averaged F1 score over the seven morphological segmentation benchmarks of Uzan et al. (2024) (higher is better), for vocabulary sizes k. Best value per column in bold.

Method	$k = 10$K	$k = 15$K	$k = 20$K	$k = 25$K	$k = 30$K
LRX+DP (ours)	**0.8001**	**0.7881**	**0.7673**	**0.7471**	**0.7241**
BPE	0.6855	0.6685	0.6589	0.6546	0.6510
BPE+DP	0.7215	0.7074	0.6973	0.6891	0.6821
GreedTok	0.6847	0.6663	0.6579	0.6522	0.6497

Results indicate that our Lagrangian relaxation method has a higher morphological alignment than the competing methods.

I Ablation and Supplementary Results

As discussed in Appendix C.2, in our Lagrangian relaxation algorithm, we use two hyperparameters to control convergence: *mu-decay* (μ_d), and *patience*. The μ_d parameter governs the multiplicative decrease of the step size (penalty coefficient) across iterations, enabling a coarse-to-fine search that stabilizes the optimization as it progresses. The *patience* parameter determines how many consecutive iterations without improvement are allowed before triggering a decay of the step size. Together, these parameters balance exploration and stability: larger patience delays decay and encourages broader search, while more aggressive mu-decay leads to faster convergence but may risk premature stagnation.

We provide below several ablations of these choices, varying both the decay and the patience.

I.1 AN ABLATION RESULT ON THE μ_d PARAMETER IN THE LAGRANGIAN RELAXATION

Figure 7 presents an ablation over the decay parameter μ_d, considering the values $\{0.7, 0.85, 0.9\}$.

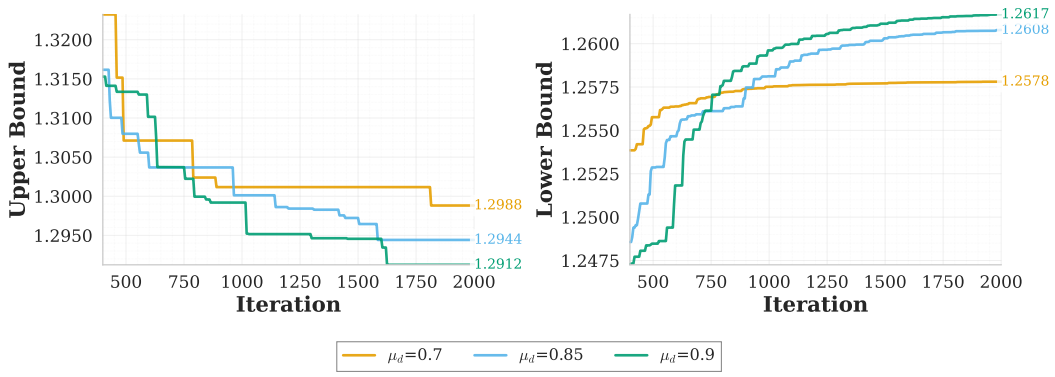

Figure 7: An ablation study on varying values of μ_d within the Lagrangian relaxation algorithm.

Overall, among the three tested values, $\mu_d = 0.9$ performs best. Although its bounds are initially looser than those obtained with smaller decay values, it eventually converges to the tightest upper and lower bounds.

I.2 AN ABLATION RESULT ON THE PATIENCE PARAMETER IN THE LAGRANGIAN RELAXATION

We now ablate the *patience* parameter under fixed values of μ_d. Figures 8 show this ablation for $\mu_d = 0.9$ and $\mu_d = 0.85$, respectively.

We can see in Figure 8 that, under $\mu_d = 0.9$, the best value for *patience* among the two tested options, $\{20, 30\}$, is 30. While it is initially worse, it eventually converges to tighter bounds.

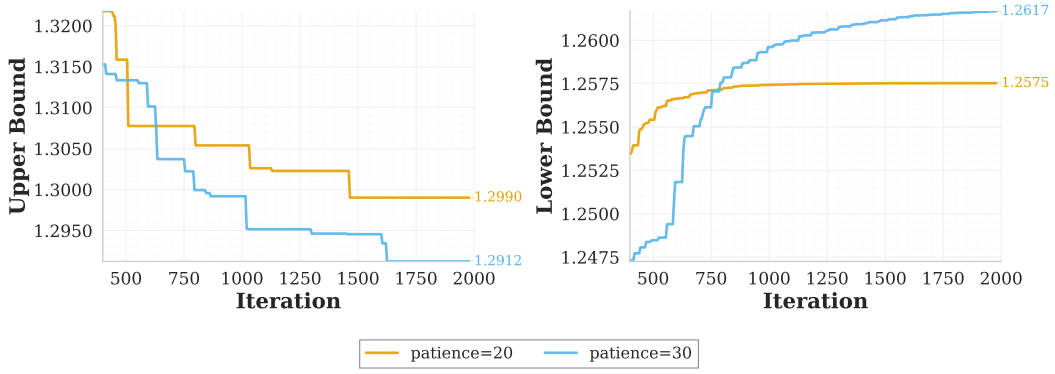

Figure 8: An ablation study for varying values of $patience$ within the Lagrangian relaxation algorithm (number of iterations for updating μ).

I.3 VARYING COMPRESSION RATIOS UNDER VARYING VOCABULARY SIZES k

To examine how the upper and lower bounds change as a function of the vocabulary size, we plot the results for multiple values in Figure 9.

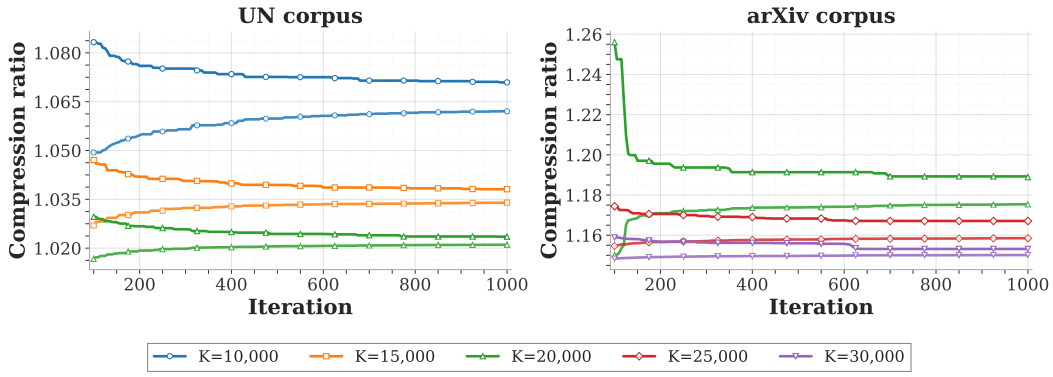

Figure 9: Ablation on vocabulary size.

As before, the bounds tighten as the iterations progress. We can see that larger vocabulary sizes lead to lower compression ratios.