Tokenisation is NP-Complete

Anonymous Authors[1]

Abstract

In this work, we prove the NP-completeness of two variants of tokenisation, defined here as the problem of compressing a dataset to at most δ symbols by either finding a vocabulary directly (*direct* tokenisation), or selecting a sequence of merge operations (*bottom-up* tokenisation).

1. Introduction

Tokenisation is at the heart of natural language processing (NLP) being the first step required to use a language model. Given a string of **characters** $\mathbf{c}$, a tokeniser converts it into a string of **subwords** $\mathbf{s}$. Language models are then trained to estimate distributions over subword-strings—never seeing the original character-strings. Despite its prominent role, however, much remains unknown about tokenisation. We still do not know, for instance, what makes a good tokeniser (Gowda & May, 2020; Cognetta et al., 2024): which characteristics should its produced subwords $\mathbf{s}$ have to be a good starting point for language modelling? If we knew this, then we could define an **objective function** with which we could evaluate tokenisers.

Another open question is how to—given such an objective function—efficiently find a tokeniser which maximises it. Byte pair encoding (BPE; Gage, 1994; Sennrich et al., 2016), for instance, is a greedy solution to find a tokeniser which maximises a text's compression. UnigramLM (Kudo, 2018) is a heuristic method to find a tokeniser that maximises its tokenised text's unigram log-probability. Both these methods, however, are approximate: they do not necessarily find an optimal tokeniser according to their objective function. This raises the question of whether finding such optimal tokenisers efficiently is even possible.

In this paper, we answer this question (at least partially), proving the NP-completeness of several variants of this tokenisation problem. Specifically, we focus on finding tokenisers that maximise the **compression** of a text.[1] Given this objective, we then define the **tokenisation problem** as the task of finding a tokeniser which compresses a dataset to at most δ symbols. Notably, prior work imposes different constraints on how tokenisers are defined; here we consider two variants. In **direct tokenisation**, the desired compression must be reached by choosing a vocabulary (i.e., a set of subwords) which is directly used to represent the text. In **bottom-up tokenisation**, the desired compression must be reached by finding a sequence of merge operations instead, which we apply to the input text.

We prove the NP-hardness of both of these tokenisation problems (as well as of some variants thereof) by reducing from the **max 2-satisfiability** problem.[2] Practically speaking, our results imply that we are unlikely to discover an efficient algorithm for the problem of finding optimal tokenisers, and that we should focus on approximate algorithms (such as BPE or UnigramLM) instead.

2. How to Choose a Tokeniser?

Given a tokeniser, any character-level distribution has an equivalent subword-level distribution (Pimentel & Meister, 2024; Phan et al., 2024; Giulianelli et al., 2024). Ergo, despite the distribution we may wish to language model, a sufficiently expressive model should be able to represent it exactly; this is true regardless which tokeniser is used. In theory, thus, a researcher's choice of tokeniser should not influence their language model's quality.

In practice, however, a bad choice of tokeniser can have undesirable effects on downstream applications. For instance, performing standard arithmetic tasks (e.g., $317 + 421$) can be difficult even for large models (Nogueira et al., 2021; Muffo et al., 2022) due to the arbitrary splitting of numbers into subwords. Indeed, simple changes in how numbers are

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]The compression achieved by a tokeniser correlates with downstream language modelling performance (Gallé, 2019; Zouhar et al., 2023a) and computational efficiency.

[2]We note two related concurrent works. Kozma & Voderholzer (2024) also prove the NP-completeness of bottom-up tokenisation; in fact, they prove something stronger: its APX-hardness. Lim et al. (2025) prove the NP-completeness of a restricted variant of direct tokenisation, in which a set of candidate tokens is previously specified.

tokenised can improve performance in such tasks (Singh & Strouse, 2024). Similar issues arise when prompting language models to count letters, where even advanced models such as `GPT-4` infamously cannot correctly count the number of occurrences of the letter r in the word strawberry.

This raises the question of how to select a good tokeniser. Ideally, we would choose the tokeniser which maximises downstream language modelling performance. Unfortunately, we do not know how to measure such performance without fully training a model, making its direct maximisation computationally infeasible. Rather, we thus optimise proxy objectives—assumed to correlate with downstream performance. Among these are unigram log-probability (Kudo, 2018), Rényi efficiency (Zouhar et al., 2023a), and compression (Gallé, 2019).

We focus on compression in this paper. Denoting our tokenisation's **objective function** as $\mathfrak{G}$, we write this objective as: $\mathfrak{G}(\mathbf{s}) = -|\mathbf{s}|$. Improved compression leads to: (i) more efficient training and inference, due to shortened inputs;[3] (ii) improved downstream performance, at least to a certain extent (Gallé, 2019; Rust et al., 2021; Zouhar et al., 2023a; Goldman et al., 2024);[4] and (iii) fairer multilingual treatment—assuming similar compression across languages—given models' limited context lengths and the per-token costs to use proprietary models (Petrov et al., 2023; Ahia et al., 2023).

Our Notation's Colour-coding

- Blue for raw data (i.e., characters $\mathbf{c} \in \Sigma^*$);
- Magenta for tokeniser-specific data (i.e., subwords $\mathbf{s} \in \mathcal{S}^*$ and merges $\mathbf{m} \in \mathcal{M}^*$);
- Orange for functions (e.g., $\texttt{tok}$).

3. Defining a Tokeniser

A tokeniser can be defined as a 3-tuple $\langle \mathcal{S}, \texttt{tok}, \texttt{detok} \rangle$, composed of a vocabulary, a tokenisation and a detokenisation function. Before defining these terms, however, we require some notation. Let $\mathbf{c} \in \Sigma^*$ be a **character-string**,[5] i.e., a sequence of characters c from alphabet Σ, which we write as $\mathbf{c} = c_1 c_2 \cdots c_{|\mathbf{c}|}$. Further, let $\mathcal{D} = \{\mathbf{c}_n\}_{n=1}^N$ be a dataset of character-strings.[6] A subword $s \in \mathcal{S}$ represents a non-empty character-string $\mathbf{c}$ (where sequence $\mathbf{c}$ can have length one). Finally, let $\mathbf{s} \in \mathcal{S}^*$ be a **subword-string**. Just like a single subword, a subword-string $\mathbf{s} = \langle s_1, s_2, \cdots, s_{|\mathbf{s}|} \rangle$ represents a character-string via the concatenation of its subwords' characters:

$$\texttt{concat}(\mathbf{s}) = s_1 \circ s_2 \circ \ldots \circ s_{|\mathbf{s}|} \tag{1}$$

and we say that a pair of character and subword strings are equivalent if:

$$\mathbf{c} \overset{\circ}{=} \mathbf{s} \iff \mathbf{c} = \texttt{concat}(\mathbf{s}) \tag{2}$$

Given the notation above, we can now define the items in tuple $\langle \mathcal{S}, \texttt{tok}, \texttt{detok} \rangle$. A tokeniser's **vocabulary** is a set of subwords $\mathcal{S} \subset \Sigma^+$ such that $\Sigma \subseteq \mathcal{S}$;[7] we say its size is $|\mathcal{S}| = |\Sigma| + K$. Further, a **detokenisation function** is defined as $\texttt{detok} : \mathcal{S}^* \to \Sigma^*$ and given a subword-string it outputs the character-string it represents. This function thus is simply defined as $\texttt{detok}(\mathbf{s}) \overset{\text{def}}{=} \texttt{concat}(\mathbf{s})$.

Finally, we are left with defining a **tokenisation function** $\texttt{tok} : \Sigma^* \to \mathcal{S}^*$, which maps from character- to subword-strings. Notably, these functions always ensure the equivalence $\mathbf{c} \overset{\circ}{=} \mathbf{s}$ for $\mathbf{s} = \texttt{tok}(\mathbf{c})$. Several tokenisation functions, however, are compatible with this constraint, as given a vocabulary, many subword-strings may be equivalent to the same character-string. For instance, given $\mathcal{S} = \{a, c, t, at\}$, the string $\mathbf{c} = c, a, t$ could be tokenised as $\mathbf{s} = \langle c, a, t \rangle$ or as $\mathbf{s} = \langle c, at \rangle$. Most researchers define tokenisation functions in one of two ways, which we term direct and bottom-up tokenisation functions; we define these next.

3.1. Direct Tokenisation Functions

In direct tokenisation, a character-string is directly replaced by an optimal subword-string. To implement this, one must thus first define what *optimal* means; this is done through an objective function $\mathfrak{G}$ which, given a subword-string, returns a score. Given a previously chosen vocabulary $\mathcal{S}$ (we discuss how to find $\mathcal{S}$ in §5), a direct tokenisation function then encodes string $\mathbf{c}$ as:

$$\texttt{tok}_{\diamond}[\mathcal{S}](\mathbf{c}) = \arg\max_{\mathbf{s} \in \mathcal{S}^*} \mathfrak{G}(\mathbf{s}) \quad \text{s.t. } \mathbf{s} \overset{\circ}{=} \mathbf{c} \tag{3}$$

[3]Recent work tries to improve the computational efficiency of byte-level models (Yu et al., 2023; Pagnoni et al., 2024).

[4]Although, see Ali et al. (2024), who argue that compression might be a necessary but not sufficient condition for good downstream performance, and Schmidt et al. (2024), who argue that compression and downstream performance have a more complex relationship than prior work suggests.

[5]We note that Σ^* denotes the Kleene star of Σ (i.e., $\cup_{i=0}^{\infty} \Sigma^i$), and Σ^+ denotes its Kleene plus (i.e., $\cup_{i=1}^{\infty} \Sigma^i$).

[6]We note that we use set notation here, but our datasets are actually multisets—datasets can include the same string $\mathbf{c}$ multiple times. We show that tokenisation is still NP-complete for datasets with no repetitions in §6.3. Further, we impose no constraint on the kind of string present in these datasets: each $\mathbf{c}_n$ can be either a raw or pre-tokenised character-string (i.e., either a full document or a whitespace-separated word).

[7]$\Sigma \subseteq \mathcal{S}$ is typically enforced to guarantee that every $\mathbf{c} \in \Sigma^*$ can be represented by at least one subword-string $\mathbf{s} \in \mathcal{S}^*$.

In words, given a vocabulary $\mathcal{S}$, function $\texttt{tok}_{\diamond}$ returns the optimal subword-string $\mathbf{s} \in \mathcal{S}^*$ which is equivalent to the input character-string $\mathbf{c}$. We then set $\texttt{tok}(\mathbf{c}) \overset{\text{def}}{=} \texttt{tok}_{\diamond}[\mathcal{S}](\mathbf{c})$. Different choices of $\mathfrak{G}$ recover methods such as UnigramLM's tokenisation function (Kudo, 2018) or PathPiece (Schmidt et al., 2024). Notably, in general, this function is not efficiently computable.[8]

In this paper, we are concerned with tokenisers that use compression as their objective: that is, for which $\mathfrak{G}(\mathbf{s}) = -|\mathbf{s}|$. In this case, we can rewrite the direct tokenisation function as:

$$\texttt{tok}_{\diamond}[\mathcal{S}](\mathbf{c}) = \underset{\mathbf{s}\in\mathcal{S}^*}{\arg\min}\, |\mathbf{s}| \quad \text{s.t. } \mathbf{s} \overset{\circ}{=} \mathbf{c} \tag{4}$$

Importantly, in the case of compression, this equation can be computed efficiently (as shown in §5.1).

3.2. Bottom-up Tokenisation Functions

In bottom-up tokenisation, one starts with a set of character-strings, and merges their symbols bottom-up, one pair at a time.[9] Formally, let $m \in \mathcal{M}$ be a **merge**, defined as a pair of subwords: $m = \langle s_1, s_2 \rangle$. Further, let $\mathcal{M} \overset{\text{def}}{=} \Sigma^+ \times \Sigma^+$. Now, let $\texttt{merge}$ be a functional; given merge $m = \langle s_1, s_2 \rangle$, it returns a function $\texttt{merge}[m] : \mathcal{S}^* \to (\mathcal{S} \cup \{s_1 \circ s_2\})^*$ which operates on string $\mathbf{s}$ left-to-right, replacing every occurrence of s_1 followed by s_2 in it with subword $s' = s_1 \circ s_2$. E.g., given $\mathbf{s} = \langle wo, r, ld \rangle$ and $m = \langle wo, r \rangle$, the output of $\texttt{merge}[m](\mathbf{s})$ is $\langle wor, ld \rangle$.

Consider now $\mathbf{m} \in \mathcal{M}^*$, a sequence of merges. Given a character-string $\mathbf{c} \in \Sigma^*$, a bottom-up tokenisation function compresses it as:

$$\texttt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c}) = \left(\bigodot_{z=1}^{|\mathbf{m}|} \texttt{merge}[m_z] \right)(\mathbf{c}) \tag{5}$$

where $\bigodot$ represents function composition, e.g., $\bigodot_{z=1}^{2} \texttt{merge}[m_z] = \texttt{merge}[m_2] \odot \texttt{merge}[m_1]$. Bottom-up tokenisers then set $\texttt{tok} \overset{\text{def}}{=} \texttt{tok}_{\uparrow}[\mathbf{m}]$. Further, a merge sequence $\mathbf{m}$ is also used to set a bottom-up tokeniser's vocabulary as:

$$\mathcal{S} = \Sigma \cup \{s_1 \circ s_2 \mid \langle s_1, s_2 \rangle \in \mathbf{m}\} \tag{6}$$

where $|\mathbf{m}| = K$ implies this vocabulary has size $|\mathcal{S}| = |\Sigma| + K$, as before.

[8] In fact, Geh et al. (2024) shows that it is NP-complete for $\mathfrak{G}(\mathbf{s}) = \sum_{t=1}^{|\mathbf{s}|} \log p_{\boldsymbol{\theta}}(s_t \mid \mathbf{s}_{<t})$, where $p_{\boldsymbol{\theta}}$ is a language model.

[9] Currently, this is likely the most common tokenisation function, being used in popular tokenisers such as, e.g., GPT-4's (OpenAI, 2023), LLaMA's (Touvron et al., 2023a;b), and Pythia's (Biderman et al., 2023).

4. Maximum 2-Satisfiability

Our paper's goal is to prove the NP-completeness of tokenisation. To show this, we must reduce an NP-hard problem to tokenisation in polynomial time. We will rely on the **maximum 2-satisfiability** problem (`max-2-SAT`) for this, whose definition we provide here. The NP-hardness of `max-2-SAT` was proven by Garey et al. (1974).

Definition 1. *Let $\mathcal{X} = \{X_j\}_{j=1}^J$ be a set of variables; each of these variables are assigned values $x_j \in \{\texttt{F}, \texttt{T}\}$, and we write $\chi = \{x_j\}_{j=1}^J \in \{\texttt{F}, \texttt{T}\}^J$. Let $\mathcal{C} = \{(L_i^1 \vee L_i^2)\}_{i=1}^I$ be a set of clauses,[10] where each literal L represents either a variable X_j or its negation $\neg X_j$. The **`max-2-SAT` decision problem** requires deciding whether there exists an assignment for which at least γ clauses are satisfied:*

$$\gamma \leq \max_{\chi \in \{\texttt{F},\texttt{T}\}^J} \sum_{i=1}^{I} \mathbb{1}_{\chi}\{L_i^1 \vee L_i^2\} \tag{7}$$

where $\mathbb{1}_{\chi}$ is an indicator function which evaluates the clause and returns one if the clause is satisfied by χ and zero otherwise.

For mathematical convenience, we will write $\text{M2S}(\mathcal{X}, \mathcal{C}, \gamma)$ for a function which returns $\texttt{T}$ if its input is satisfiable under a `max-2-SAT` decision problem, and $\texttt{F}$ otherwise. As a concrete example, consider the set of variables $\mathcal{X} = \{X_1, X_2\}$ and the set of clauses $\mathcal{C} = \{X_1 \vee X_2, \neg X_1 \vee X_2, X_1 \vee \neg X_2, \neg X_1 \vee \neg X_2\}$. The assignment $x_1 = \texttt{T}, x_2 = \texttt{T}$ leads to 3 clauses being satisfied, which is the optimum. For this example, we thus have that $\text{M2S}(\mathcal{X}, \mathcal{C}, 3) = \texttt{T}$, but that $\text{M2S}(\mathcal{X}, \mathcal{C}, 4) = \texttt{F}$.

5. Finding Optimal Direct Tokenisers

We are now left with the task of finding an optimal tokeniser. We do this by selecting either: its vocabulary in direct tokenisation, since $\texttt{tok} = \texttt{tok}_{\diamond}[\mathcal{S}]$; or its merge sequence in bottom-up tokenisation, since $\texttt{tok} = \texttt{tok}_{\uparrow}[\mathbf{m}]$ and since its vocabulary is chosen according to Eq. (6). (Note that in §3, we only showed how to apply tokenisers at inference time, but not how to find them.) In this section, we focus on direct tokenisation, defining its optimisation and decision problems; we then prove its NP-completeness. The optimisation problem is defined as follows.

Definition 2. *Given a dataset $\mathcal{D}$ and a vocabulary size K, the **direct tokenisation optimisation problem** is to find a*

[10] `max-2-SAT` also allows clauses to have a single literal L_i. In this case, we can always rewrite the clause as $(L_i \vee L_i)$ with no change to the solution of this decision problem.

vocabulary $\mathcal{S}_{\texttt{opt}} \subset \Sigma^+$ which maximally compresses $\mathcal{D}$:

$$\mathcal{S}_{\texttt{opt}} = \operatorname*{arg\,min}_{\mathcal{S}\subset\Sigma^+} \sum_{\mathbf{c}\in\mathcal{D}} |\texttt{tok}_{\Rightarrow}[\mathcal{S}](\mathbf{c})| \qquad (8)$$
$$\text{s.t. } |\mathcal{S}| = |\Sigma| + K$$

We can similarly define direct tokenisation's decision problem.

Definition 3. *Given a dataset $\mathcal{D}$ and a vocabulary size K, the* ***direct tokenisation decision problem*** *requires deciding whether there exists a vocabulary $\mathcal{S} \subset \Sigma^+$ which compresses $\mathcal{D}$ to at most δ symbols:*

$$\delta \geq \min_{\mathcal{S}\subset\Sigma^+} \sum_{\mathbf{c}\in\mathcal{D}} |\texttt{tok}_{\Rightarrow}[\mathcal{S}](\mathbf{c})| \qquad (9)$$
$$\text{s.t. } |\mathcal{S}| = |\Sigma| + K$$

We write $\text{Tok}_{\Rightarrow}(\mathcal{D}, K, \delta)$ for a function which returns $\texttt{T}$ if a direct tokenisation decision problem with those inputs is satisfiable, and $\texttt{F}$ otherwise. Note that, whenever $|\mathcal{D}| \leq K$, the solution to the problem above is trivial, as an optimal solution simply requires including all strings $\mathbf{c}_n$ in vocabulary $\mathcal{S}$. As we show next, however, in the general case the above decision problem is NP-complete. We now state this as a theorem, which we will prove in the next two sections.

Theorem 1. *The direct tokenisation decision problem, as in Definition 3, is NP-complete.*

Proof. A decision problem is considered to be NP-complete if: (i) it is in NP; (ii) it is NP-hard. We prove these conditions in §5.1 and §5.2. □

5.1. Direct Tokenisation is in NP

A decision problem is in the nondeterministic polynomial time class (NP) if, given a **certificate** of polynomial length, one can verify that certificate in polynomial time. Specifically, a certificate usually encodes a decision problem's solution, allowing us to verify its satisfiability. In the case of direct tokenisation, this certificate would be a vocabulary $\mathcal{S}$ which leads a dataset $\mathcal{D}$ to be compressed to at most δ symbols. Verifying this certificate simply requires computing the sum in Eq. (9), i.e.:

$$\sum_{\mathbf{c}\in\mathcal{D}} |\texttt{tok}_{\Rightarrow}[\mathcal{S}](\mathbf{c})| \qquad (10)$$

Lemma 1. *The direct tokenisation decision problem, as in Definition 3, is in NP.*

Proof. As noted above, whenever $|\mathcal{D}| \leq K$, each $\mathbf{c}_n \in \mathcal{D}$ can be included in the vocabulary $\mathcal{S}$ and fully compressed to a single symbol; we can thus verify the problem's satisfiability by simply checking that $\delta \geq |\mathcal{D}|$ as this is the best reachable compression. Assuming K to be bounded by $|\mathcal{D}|$—and therefore polynomial in the input—we have that the certificate $\mathcal{S}$ also has polynomial length. Given such a certificate $\mathcal{S}$, verifying it simply requires computing the sum in Eq. (10). In turn, computing this sum requires $|\mathcal{D}|$ calls to function $\texttt{tok}_{\Rightarrow}$. It follows that, if function $\texttt{tok}_{\Rightarrow}$ runs in polynomial time, then direct tokenisation is in NP. Luckily, this function can indeed be computed efficiently using Schmidt et al.'s (2024) PathPiece method, which runs in $O(|\mathbf{c}|^2)$ time. This is achieved by first converting $\mathbf{c}$ into a directed acyclic graph where nodes represent string positions $[0, 1, \ldots, |\mathbf{c}|]$ and where two nodes t, t' are connected if there exists a subword $s \in \mathcal{S}$ for which $\mathbf{c}_{t:t'} \overset{\circ}{=} s$. E.g., for a string $\mathbf{c}$ and $\mathcal{S} = \Sigma \cup \{\mathbf{c}_{0:2}, \mathbf{c}_{1:t}\}$, we build a graph:

$$\begin{tikzcd}
0 \arrow[r] \arrow[rr, bend right] & 1 \arrow[r] \arrow[rrr, bend right] & 2 \arrow[r] & \cdots \arrow[r] & t \arrow[r] & \cdots \arrow[r] & {|\mathbf{c}|}
\end{tikzcd}$$

The shortest path from node 0 to $|\mathbf{c}|$ in this graph then gives us $\texttt{tok}_{\Rightarrow}[\mathcal{S}](\mathbf{c})$. As the shortest path of a directed acyclic graph can be computed in $O(N + V)$ time, the time complexity of finding the shortest path in this directed acyclic graph is thus bounded by $O(|\mathbf{c}|^2)$. □

5.2. Direct Tokenisation is NP-hard

We now use a reduction from `max-2-SAT` to prove the NP-hardness of direct tokenisation.

Reduction 1. *Let us have an instance of the* `max-2-SAT` *decision problem as in Definition 1. To reduce this instance to an instance of the direct tokenisation decision problem, as in Definition 3, we first define an alphabet $\Sigma = \{\circledcirc\} \cup \{x_j^{\texttt{T}}, x_j^{\texttt{F}}\}_{j=1}^{J}$. We then construct three sets of strings:*

$$\mathcal{D}_1 = \{\circledcirc x_j^{\texttt{T}} \circledcirc\}_{j=1}^{J} \cup \{\circledcirc x_j^{\texttt{F}} \circledcirc\}_{j=1}^{J} \qquad (11a)$$
$$\mathcal{D}_2 = \{\circledcirc x_j^{\texttt{T}} \circledcirc x_j^{\texttt{F}} \circledcirc\}_{j=1}^{J} \qquad (11b)$$
$$\mathcal{D}_3 = \{\circledcirc L_i^1 \circledcirc L_i^2 \circledcirc\}_{i=1}^{I} \qquad (11c)$$

In these strings L_i is replaced by either character $x_j^{\texttt{T}}$ or $x_j^{\texttt{F}}$, depending on whether it represents X_j or $\neg X_j$, respectively. We then construct our dataset $\mathcal{D}$, and choose K and δ as:

$$\mathcal{D} = \left(\bigcup_{\_=1}^{f} \mathcal{D}_1\right) \cup \left(\bigcup_{\_=1}^{f'} \mathcal{D}_2\right) \cup \mathcal{D}_3 \qquad (12a)$$
$$K = J, \quad \delta = (4f + 3f')\,J + 5\,I - 2\gamma \qquad (12b)$$

where we set $f' \overset{\text{def}}{=} 2I + 1$ and $f \overset{\text{def}}{=} 4f'J + 4I + 1$.

We write $\text{R1}(\mathcal{X}, \mathcal{C}, \gamma)$ to represent a function which, given an instance of `max-2-SAT`, returns an instance of the tokenisation problem given by our reduction (i.e., $\mathcal{D}, K, \delta$).

For our reduction to be correct, we must have that:

$$\mathrm{M2S}(\mathcal{X}, \mathcal{C}, \gamma) \iff \mathrm{Tok}_{\diamond}(\mathrm{R1}(\mathcal{X}, \mathcal{C}, \gamma)) \tag{13}$$

meaning that a `max-2-SAT` instance is satisfiable if and only if its reduced direct tokenisation instance is as well. We now set out to prove this. We start by proving the forward direction of this iff clause.

Lemma 2. *If a `max-2-SAT` instance is satisfiable, then the direct tokenisation instance output by Reduction 1 is also satisfiable. Formally:*

$$\mathrm{M2S}(\mathcal{X}, \mathcal{C}, \gamma) \implies \mathrm{Tok}_{\diamond}(\mathrm{R1}(\mathcal{X}, \mathcal{C}, \gamma)) \tag{14}$$

Proof sketch. See a formal proof in App. A. Our proof works by first fixing a satisfying solution to `max-2-SAT` with values $x_j^\star$. Given this solution, for each variable, we add to our vocabulary $\mathcal{S}$ a subword $\circledcirc x_j^{\mathtt{T}} \circledcirc$ if $x_j^\star$ is true, or $\circledcirc x_j^{\mathtt{F}} \circledcirc$ if $x_j^\star$ is false. Given these subwords, strings in $\mathcal{D}_1$ and $\mathcal{D}_2$ occupy a total length of $(4f + 3f')\,J$. Further, since at least γ of the `max-2-SAT` clauses are satisfied by $x_j^\star$, the strings in $\mathcal{D}_3$ will occupy a total length smaller or equal to $5\,I - 2\gamma$. This solution to the tokenisation problem thus gives us a total length which is smaller or equal to $\delta = (4f + 3f')\,J + 5\,I - 2\gamma$. □

Now, we are left with proving the backward direction of the iff clause in Eq. (13). We do so in the following lemma.

Lemma 3. *If the direct tokenisation instance output by Reduction 1 is satisfiable, the `max-2-SAT` instance which generated it is as well. Formally:*

$$\mathrm{Tok}_{\diamond}(\mathrm{R1}(\mathcal{X}, \mathcal{C}, \gamma)) \implies \mathrm{M2S}(\mathcal{X}, \mathcal{C}, \gamma) \tag{15}$$

Proof sketch. See a formal proof in App. B. Our proof works in three steps. First, we show that any satisfying solution must only have subwords of the form $\circledcirc x_j^{\mathtt{T}} \circledcirc$ or $\circledcirc x_j^{\mathtt{F}} \circledcirc$, since this is required to compress strings in $\mathcal{D}_1$ to at most $4fJ$ symbols. Second, we show that any satisfying solution must only have either subword $\circledcirc x_j^{\mathtt{T}} \circledcirc$ or $\circledcirc x_j^{\mathtt{F}} \circledcirc$ for any variable X_j; this is required to compress strings in $\mathcal{D}_2$ to at most $3f'J$ symbols. Finally, we show that if a tokeniser compresses strings in $\mathcal{D}_3$ to $5I - 2\gamma$, then there is an assignment χ which satisfies at least γ of the original `max-2-SAT` problem. □

Given both lemmas above, we can now trivially prove that direct tokenisation is NP-hard.

Lemma 4. *The direct tokenisation decision problem, as in Definition 3, is NP-hard.*

Proof. First, it is easy to see that Reduction 1 runs in polynomial time. Second, `max-2-SAT` is an NP-hard problem (Garey et al., 1974). This lemma then follows trivially from Lemmas 2 and 3, which together show that an instance of the tokenisation problem generated through Reduction 1 is satisfiable if and only if the `max-2-SAT` instance used to produce it is also satisfiable. □

6. Finding Optimal Bottom-up Tokenisers

We now shift our attention to bottom-up tokenisation. We define both its optimisation and decision problems, and then prove its NP-completeness. We start with defining the optimisation problem.

Definition 4. *Given a dataset $\mathcal{D}$ and a vocabulary size K, the **bottom-up tokenisation optimisation problem** is to find a merge sequence $\mathbf{m}_{\mathtt{opt}} \in \mathcal{M}^*$ which maximally compresses $\mathcal{D}$:*

$$\mathbf{m}_{\mathtt{opt}} = \operatorname*{arg\,min}_{\mathbf{m} \in \mathcal{M}^*} \sum_{\mathbf{c} \in \mathcal{D}} |\mathtt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})| \qquad \text{s.t. } |\mathbf{m}| = K \tag{16}$$

As can be seen, this optimisation problem is similar to the direct tokenisation problem, albeit its target is to find a merge sequence instead of a vocabulary. We similarly define a decision problem.

Definition 5. *Given a dataset $\mathcal{D}$ and a vocabulary size K, the **bottom-up tokenisation decision problem** requires deciding whether there exists a merge sequence $\mathbf{m} \in \mathcal{M}^*$ which compresses $\mathcal{D}$ to at most δ symbols:*

$$\delta \geq \min_{\mathbf{m} \in \mathcal{M}^*} \sum_{\mathbf{c} \in \mathcal{D}} |\mathtt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})| \qquad \text{s.t. } |\mathbf{m}| = K \tag{17}$$

We write $\mathrm{Tok}_{\uparrow}(\mathcal{D}, K, \delta)$ for a function which returns `T` if a bottom-up tokenisation decision problem with those inputs is satisfiable, and `F` otherwise. We spend the rest of this section showing that bottom-up tokenisers are NP-complete.

Theorem 2. *The bottom-up tokenisation decision problem, as in Definition 5, is NP-complete.*

Proof. We prove this in two steps below. We first prove that this problem is in NP, in §6.1. We then prove that this problem is NP-hard, in §6.2. □

6.1. Bottom-up Tokenisation is in NP

We can verify this using a solution, the merge sequence $\mathbf{m} \in \mathcal{M}^*$, as a certificate. By showing that this certificate has polynomial length and that it can be verified in polynomial time, we prove this problem is in NP. To verify this certificate, we simply need to compute the sum in Eq. (17), i.e.:

$$\sum_{\mathbf{c} \in \mathcal{D}} |\mathtt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})| \tag{18}$$

which we show now can be done efficiently.

Lemma 5. *The bottom-up tokenisation decision problem, as in Definition 5, is in NP.*

Proof. First, if K is larger than the total number of characters in $\mathcal{D}$, i.e., $\sum_{\mathbf{c}\in\mathcal{D}} |\mathbf{c}|$, then this dataset can be compressed to $|\mathcal{D}|$ by merging each string down to a single symbol; further, compressing $\mathcal{D}$ more than that is not possible independently of K. Verifying the satisfiability of such an instance of the tokenisation problem is thus trivial, only requiring checking if $\delta \geq |\mathcal{D}|$. Second, if K is bounded by $\sum_{\mathbf{c}\in\mathcal{D}} |\mathbf{c}|$—and therefore polynomial in the input—the certificate $\mathbf{m}$ has polynomial length. Given such a certificate $\mathbf{m}$, verifying it then simply requires computing the sum in Eq. (18). In turn, computing this sum requires $|\mathcal{D}|$ calls to function $\mathtt{tok}_{\uparrow}$. It follows that, if function $\mathtt{tok}_{\uparrow}$ runs in polynomial time, then bottom-up tokenisation is in NP. The computation of $\mathtt{tok}_{\uparrow}$, can be done in polynomial time following the structure described in §3.2. For each $m = \langle s_1, s_2\rangle$ in $\mathbf{m}$, scan the current $\mathbf{c}$ and replace each occurrence of s_1, s_2 by s'. This takes time $\mathcal{O}(|\mathbf{c}|)$ for each merge. Afterwards, the resulting string can be compared against the desired size. We obtain a total runtime of $O(|\mathcal{D}||\mathbf{c}||\mathbf{m}|)$. □

6.2. Bottom-up Tokenisation is NP-hard

As before, we use a reduction from `max-2-SAT` to prove bottom-up tokenisation's NP-hardness.

Reduction 2. *Let us have an instance of the* `max-2-SAT` *decision problem as in Definition 1. To reduce this instance to an instance of the bottom-up tokenisation decision problem, as in Definition 5, we first define an alphabet* $\Sigma = \{\circledcirc, \otimes\} \cup \{x_j^{\mathtt{T}}, x_j^{\mathtt{F}}\}_{j=1}^{J}$. *We then construct five sets of strings:*

$$\mathcal{D}_1 = \{\circledcirc x_j^{\mathtt{T}}\}_{j=1}^{J} \cup \{x_j^{\mathtt{F}}\circledcirc\}_{j=1}^{J} \cup \{x_j^{\mathtt{T}}\circledcirc\}_{j=1}^{J} \cup \{\circledcirc x_j^{\mathtt{F}}\}_{j=1}^{J} \cup \{x_j^{\mathtt{T}}\otimes\}_{j=1}^{J} \cup \{\otimes x_j^{\mathtt{F}}\}_{j=1}^{J} \tag{19}$$

$$\mathcal{D}_2 = \{\circledcirc x_j^{\mathtt{T}}\circledcirc\}_{j=1}^{J} \cup \{\circledcirc x_j^{\mathtt{F}}\circledcirc\}_{j=1}^{J} \cup \{\circledcirc x_j^{\mathtt{T}}\otimes\}_{j=1}^{J} \cup \{\otimes x_j^{\mathtt{F}}\circledcirc\}_{j=1}^{J}$$

$$\mathcal{D}_3 = \{\circledcirc x_j^{\mathtt{T}} \circledcirc x_j^{\mathtt{F}}\circledcirc\}_{j=1}^{J} \cup \{\otimes x_j^{\mathtt{F}} \circledcirc x_j^{\mathtt{T}}\otimes\}_{j=1}^{J}$$

$$\mathcal{D}_4 = \{\circledcirc x_j^{\mathtt{F}} \circledcirc x_j^{\mathtt{T}}\otimes\}_{j=1}^{J} \cup \{\otimes x_j^{\mathtt{F}} \circledcirc x_j^{\mathtt{T}}\circledcirc\}_{j=1}^{J}$$

$$\mathcal{D}_5 = \left\{\begin{matrix} \circledcirc x_j^{\mathtt{T}} \circledcirc x_{j'}^{\mathtt{F}}\circledcirc & \mathtt{if}\ L_i^1 = X_j & \mathtt{and}\ L_i^2 = \neg X_{j'} \\ \circledcirc x_{j'}^{\mathtt{T}} \circledcirc x_j^{\mathtt{F}}\circledcirc & \mathtt{if}\ L_i^1 = \neg X_j & \mathtt{and}\ L_i^2 = X_{j'} \\ \otimes x_j^{\mathtt{F}} \circledcirc x_{j'}^{\mathtt{F}}\circledcirc & \mathtt{if}\ L_i^1 = \neg X_j & \mathtt{and}\ L_i^2 = \neg X_{j'} \\ \circledcirc x_j^{\mathtt{T}} \circledcirc x_{j'}^{\mathtt{T}}\otimes & \mathtt{if}\ L_i^1 = X_j & \mathtt{and}\ L_i^2 = X_{j'} \end{matrix}\right\}_{i=1}^{I}$$

We then construct our dataset $\mathcal{D}$, *and choose* K *and* δ *as:*

$$\mathcal{D} = \bigcup_{\_=1}^{f} \mathcal{D}_1 \cup \bigcup_{\_=1}^{f'} \mathcal{D}_2 \cup \bigcup_{\_=1}^{f''} \mathcal{D}_3 \cup \bigcup_{\_=1}^{f'''} \mathcal{D}_4 \cup \mathcal{D}_5 \tag{20}$$

$$K = 8J, \;\; \delta = (6f + 6f' + 4f'' + 4f''')\, J + 3\,I - \gamma$$

where we set:

$$f''' \overset{\text{def}}{=} 5I, \quad f'' \overset{\text{def}}{=} 10 f''' J + 5I \tag{21a}$$

$$f' \overset{\text{def}}{=} (10f'' + 10f''')\, J + 5I \tag{21b}$$

$$f \overset{\text{def}}{=} (12f' + 10f'' + 10f''')\, J + 5I \tag{21c}$$

As before, we write $\mathrm{R2}(\mathcal{X}, \mathcal{C}, \gamma)$ for a function which, given an instance of the `max-2-SAT` problem, returns an instance of the bottom-up tokenisation problem. For our reduction to be correct, we must have that:

$$\mathrm{M2S}(\mathcal{X}, \mathcal{C}, \gamma) \iff \mathrm{Tok}_{\uparrow}(\mathrm{R2}(\mathcal{X}, \mathcal{C}, \gamma)) \tag{22}$$

We follow the same proof strategies as before, starting by proving the forward direction of this iff statement.

Lemma 6. *If a* `max-2-SAT` *instance is satisfiable, then the bottom-up tokenisation instance output by Reduction 2 is also satisfiable. Formally:*

$$\mathrm{M2S}(\mathcal{X}, \mathcal{C}, \gamma) \implies \mathrm{Tok}_{\uparrow}(\mathrm{R2}(\mathcal{X}, \mathcal{C}, \gamma)) \tag{23}$$

Proof sketch. See a formal proof in App. C. Without loss of generality, let a satisfying solution to `max-2-SAT` have values $x_j^{\star}$. Our proof works by first defining the three following lists of merges, which must be included in any satisfying solution to this tokenisation problem:

$$\mathbf{m}_1 = \bigcirc_{j=1}^{J} [\langle \otimes, x_j^{\mathtt{F}}\rangle, \langle x_j^{\mathtt{T}}, \otimes\rangle] \tag{24a}$$

$$\mathbf{m}_3 = \bigcirc_{j=1}^{J} [\langle x_j^{\mathtt{F}}, \circledcirc\rangle, \langle \circledcirc, x_j^{\mathtt{T}}\rangle] \tag{24b}$$

$$\mathbf{m}_5 = \bigcirc_{j=1}^{J} [\langle \circledcirc, x_j^{\mathtt{F}}\rangle, \langle x_j^{\mathtt{T}}, \circledcirc\rangle] \tag{24c}$$

We then construct two other lists of merges, which depend on the satisfying assignments to `max-2-SAT`:

$$\mathbf{m}_2 = \bigcirc_{j=1}^{J} \begin{bmatrix} \langle \circledcirc, x_j^{\mathtt{T}}\otimes\rangle & \mathtt{if}\ x_j^{\star} = \mathtt{T} \\ \langle \otimes x_j^{\mathtt{F}}, \circledcirc\rangle & \mathtt{else} \end{bmatrix} \tag{25a}$$

$$\mathbf{m}_4 = \bigcirc_{j=1}^{J} \begin{bmatrix} \langle \circledcirc x_j^{\mathtt{T}}, \circledcirc\rangle & \mathtt{if}\ x_j^{\star} = \mathtt{T} \\ \langle \circledcirc, x_j^{\mathtt{F}}\circledcirc\rangle & \mathtt{else} \end{bmatrix} \tag{25b}$$

Finally, we create a merge sequence by concatenating these lists in order:

$$\mathbf{m} = \mathbf{m}_1 \circ \mathbf{m}_2 \circ \mathbf{m}_3 \circ \mathbf{m}_4 \circ \mathbf{m}_5 \tag{26}$$

Note that we have exactly $K = 8J$ merges in this list. Given this merge sequence, it is easy to verify that strings in $\mathcal{D}_1$ to $\mathcal{D}_4$ will use exactly $(6f + 6f' + 4f'' + 4f''')\, J$ symbols after being tokenised. Further, since at least γ of the `max-2-SAT`'s clauses are satisfied by $x_j^{\star}$, the strings in $\mathcal{D}_5$ will occupy a total length smaller or equal to $3\,I - \gamma$. This solution to the tokenisation problem thus gives us a tokeniser which will compress $\mathcal{D}$ to at most $\delta = (6f + 6f' + 4f'' + 4f''')\, J + 3\,I - \gamma$. □

We now prove the backward direction of the iff clause in Eq. (22).

Lemma 7. *If the bottom-up tokenisation instance output by Reduction 2 is satisfiable, the* `max-2-SAT` *instance which generated it is as well. Formally:*

$$\mathrm{Tok}_{\uparrow}(\mathrm{R2}(\mathcal{X}, \mathcal{C}, \gamma)) \implies \mathrm{M2S}(\mathcal{X}, \mathcal{C}, \gamma) \tag{27}$$

Proof sketch. See a formal proof in App. D. Our proof works in five steps. First, we show that all satisfying solutions must include merges $\mathbf{m}_1$, $\mathbf{m}_3$, and $\mathbf{m}_5$ from Eq. (24), since this is required to compress strings in $\mathcal{D}_1$ to at most $6fJ$ symbols. Second, we show the other merges of any satisfying solution must be of the form:

$$\mathbf{m}_j^{\circledcirc} = \left\{ \begin{matrix} \langle \circledcirc x_j^{\mathtt{T}}, \circledcirc \rangle, \langle \circledcirc, x_j^{\mathtt{F}} \circledcirc \rangle \\ \langle \circledcirc, x_j^{\mathtt{T}} \circledcirc \rangle, \langle \circledcirc x_j^{\mathtt{F}}, \circledcirc \rangle \end{matrix} \right\} \tag{28a}$$

$$\mathbf{m}_j^{\otimes} = \left\{ \begin{matrix} \langle \circledcirc, x_j^{\mathtt{T}} \otimes \rangle, \langle \otimes x_j^{\mathtt{F}}, \circledcirc \rangle \\ \langle \circledcirc x_j^{\mathtt{T}}, \otimes \rangle, \langle \otimes, x_j^{\mathtt{F}} \circledcirc \rangle \end{matrix} \right\} \tag{28b}$$

this is required to compress strings in $\mathcal{D}_2$ to at most $6f'J$ symbols. Third, we show that any satisfying solution will have at least one merge of each set $\mathbf{m}_j^{\circledcirc}$ and one of each set $\mathbf{m}_j^{\otimes}$; this is required to compress strings in $\mathcal{D}_3$ to at most $4f''J$ symbols. Fourth, we show that any satisfying solution will have—for each $j \in \{1, ..., J\}$—both its merges in sets $\mathbf{m}_j^{\circledcirc}$ and $\mathbf{m}_j^{\otimes}$ containing character $x_j^{\mathtt{T}}$ or character $x_j^{\mathtt{F}}$; this is required to compress strings in $\mathcal{D}_4$ to at most $4f'''J$ symbols. Finally, we show that if a tokeniser compresses strings in $\mathcal{D}_5$ to $3I - \gamma$, then there is an assignment χ which satisfies at least γ of the original `max-2-SAT` problem. □

Finally, given both lemmas above, we can now prove that bottom-up tokenisation is NP-hard.

Lemma 8. *The bottom-up tokenisation decision problem, as in Definition 5, is NP-hard.*

Proof. First, it is easy to see that Reduction 2 runs in polynomial time. Second, `max-2-SAT` is an NP-hard problem (Garey et al., 1974). This lemma then follows trivially from Lemmas 6 and 7. □

6.3. Other Definitions of Tokenisation

We now expand our discussion to consider variations of the above tokenisation problems.

Deduped Datasets. Our definitions of both direct and bottom-up tokenisation allow datasets $\mathcal{D}$ to include repeated entries. It is common, however, to deduplicate datasets in NLP—thus removing repeated entries. A small change to both our reductions is enough to adapt it to this deduplicated dataset case: simply append each string in the repeated datasets (either $\mathcal{D}_1$ and $\mathcal{D}_2$ in Reduction 1 or $\mathcal{D}_1$ to $\mathcal{D}_4$ in Reduction 2) with a unique character $\{a_y\}_{y=1}^{\infty}$ and increase the target compression size δ accordingly (by $f + f'$ or $f + f' + f'' + f'''$, respectively). These new characters will never be included in optimal tokenisers' solutions, and thus the previous proofs hold, with the difference that each dataset will require extra symbols once compressed.

A Single Long String. In the previous sections, we considered tokenisers trained on a dataset $\mathcal{D}$. Work on compression, however, usually considers a single long string $\mathbf{c}$ as its input. It is easy to see that direct tokenisation is not an NP-complete problem if its input is a single long string; including this string in vocabulary $\mathcal{S}$ already achieves optimal compression. Bottom-up tokenisation, however, is still NP-complete even when given a single string as input. As before, this can be shown with a similar strategy to Reduction 2, but where we first append each string in dataset $\mathcal{D}$ with a unique character $\{a_y\}_{y=1}^{\infty}$ and then concatenate all these strings. As in the deduped case above, characters a_y will never be merged by any optimal tokeniser; they will thus serve as virtual string delimiters and will not affect our proofs beyond an increase to the target compression size δ.

A Hybrid Approach. Finally, the last variant we consider is a hybrid between direct and bottom-up tokenisation, where we find a merge sequence $\mathbf{m}$ which—when we extract a vocabulary from it as $\mathcal{S} = \Sigma \cup \{s_1 \circ s_2 \mid \langle s_1, s_2 \rangle \in \mathbf{m}\}$—optimally compresses a dataset $\mathcal{D}$ using the direct tokenisation function in Eq. (4). We can easily prove the NP-hardness of this tokenisation variant by relying on Reduction 2; as our proof in Lemma 8 did not make use of the order of merges in $\mathbf{m}$, only of the subwords composed by it, this lemma's proof strategy can be similarly applied to this hybrid variant.

7. Tokenisation and Compression

The variants of tokenisation that we consider here—with compression as their objective function—are closely related to the field of dictionary compression. In both fields, we wish to reduce the size of an input ($\mathbf{c}$ or $\mathcal{D}$) by exploiting repetitive elements. In fact, the most popular tokenisation algorithm to date, BPE, was originally proposed as a compression algorithm (Gage, 1994) and has only somewhat recently been ported into NLP to find tokenisers (by Sennrich et al., 2016).

Not surprisingly, prior work has also considered, from a theoretical perspective, the compression tokenisers achieve. Zouhar et al. (2023b), for instance, analyse bottom-up tokenisation and prove an approximation bound on the compression achieved by the tokenisers found using BPE. More recently, Kozma & Voderholzer (2024) also analyses bottom-up tokenisation, proving a tighter bound on this compression achieved by BPE.

A popular dictionary compression method, the **straight-line program** (SLP; Kieffer & Yang, 2000; Charikar et al., 2005), can be used to illustrate the similarities and differences between tokenisers and compressors.[11] Given a string $\mathbf{c}$, an SLP describes a context-free grammar from which $\mathbf{c}$ can be uniquely derived. Formally, an SLP in Chomsky normal form (CNF) is a set of rules of form $X \rightarrow a$ or $X \rightarrow AB$, where X, A, B are called nonterminals and a is a terminal.[12] Starting from a special nonterminal S, applying these rules exhaustively—until only terminals are left—produces exactly the desired string $\mathbf{c}$. Notably, given a string $\mathbf{c}$, it is NP-complete to find the smallest SLP which generates it (Charikar et al., 2005).

On the one hand, SLPs in CNF are closely linked to bottom-up tokenisation; each of its rules expands to two nonterminals, and thus corresponds to a merge. However, while SLPs must find the minimum number of merges (or rules) to fully compress a string into a single symbol, bottom-up tokenisers must maximally compress the string given a fixed number of merges. On the other hand, SLPs which are not in CNF (that is, for which other context-free production rules are allowed, as long as the decoding stays unique) are closely linked to direct tokenisation. In this case, a direct tokeniser could be converted into an SLP with depth two; this grammar has a start rule $S \rightarrow \mathbf{s}$, and a rule from each subword to its characters $s \rightarrow \mathbf{c}$. Again, while SLPs must find a minimal grammar representing the string, direct tokenisers must minimise the size of rule $S \rightarrow \mathbf{s}$ given a fixed number of rules $s \rightarrow \mathbf{c}$.

The paragraphs above highlights two important differences between tokenisers and compressors. First, tokenisers aim to reduce only the size of the resulting tokenised text (i.e., $|\mathbf{s}|$), whereas compressors also consider the size of the compression information (e.g., considering the size required to store $\mathcal{S}$, which would be $\sum_{s \in \mathcal{S}} |\texttt{detok}(s)|$). This is because tokenisers must create shorter inputs for NLP algorithms, while compressors must make information compact. Second, tokenisers and compressors have different optimisation parameters. Compression algorithms always compress a string to the best extent possible (e.g., for SLPs, until a single nonterminal is reached), whereas tokenisation algorithms are given a maximum vocabulary size (i.e., K) and find tokenisers which only compress their input as much as possible until this limit is reached.

[11]See Lohrey (2012) for an overview of straight-line programs, and Kempa & Prezza (2018); Kociumaka et al. (2023) for a more detailed overview of compression in general.

[12]Although not originally defined that way, SLP's grammars are typically assumed to be in CNF, for simplicity. This does not make a big difference for compression, but will be important for our purposes.

8. Conclusion

In this work, we proved the NP-completeness of two variants of tokenisation. These results underline that finding optimal tokenisers most likely will remain a difficult quest and that research should focus on approximate algorithms instead. Regarding those, there is potential both in improving the analysis of currently used algorithms, such as BPE, as well as in designing new ones.

Limitations

While we prove the NP-completeness of multiple variants of the tokenisation problem—which is an important part of modern language modelling pipelines—we must note a few limitations in our work. First, we only prove NP-completeness of tokenisation with compression as its objective. This is a popular objective function, frequently used to judge the quality of tokenisers (e.g., Liu et al., 2025); however, it is not perfectly correlated with downstream language modelling performance, as discussed in §2.[13] Investigating the complexity of tokenisation under other objective functions is important. Second, our proofs do not assume a fixed alphabet size, so for fixed alphabets tokenisation might not be NP-complete. Tokenisers are frequently run at the byte level, for which specialised, more efficient algorithms might exist. Finally, while we investigated the complexity of the tokenisation problem for two types of tokenisation functions, similar results for other variants (with other tokenisation functions) remain open; we believe this would be exciting future work.

References

Ahia, O., Kumar, S., Gonen, H., Kasai, J., Mortensen, D., Smith, N., and Tsvetkov, Y. Do all languages cost the same? Tokenization in the era of commercial language models. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 9904–9923, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.614. URL `https://aclanthology.org/2023.emnlp-main.614`.

Ali, M., Fromm, M., Thellmann, K., Rutmann, R., Lübbering, M., Leveling, J., Klug, K., Ebert, J., Doll, N., Buschhoff, J., Jain, C., Weber, A., Jurkschat, L.,

[13]Relatedly, in practice, state-of-the-art language models still achieve very good performance (at least in English), despite sub-optimal tokenisers. These tokenisers thus do not prevent good performance; rather they seem to make the training process less efficient. Studying the relationship between tokenisation choices and training efficiency, as in Schäfer et al. (2024) and Tao et al. (2024), is important future work.

Abdelwahab, H., John, C., Ortiz Suarez, P., Ostendorff, M., Weinbach, S., Sifa, R., Kesselheim, S., and Flores-Herr, N. Tokenizer choice for LLM training: Negligible or crucial? In Duh, K., Gomez, H., and Bethard, S. (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 3907–3924, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.247. URL https://aclanthology.org/2024.findings-naacl.247.

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, 2023. URL https://proceedings.mlr.press/v202/biderman23a/biderman23a.pdf.

Charikar, M., Lehman, E., Liu, D., Panigrahy, R., Prabhakaran, M., Sahai, A., and Shelat, A. The smallest grammar problem. *IEEE Transactions on Information Theory*, 51(7):2554–2576, 2005. doi: 10.1109/TIT.2005.850116.

Cognetta, M., Zouhar, V., Moon, S., and Okazaki, N. Two counterexamples to tokenization and the noiseless channel. In Calzolari, N., Kan, M.-Y., Hoste, V., Lenci, A., Sakti, S., and Xue, N. (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 16897–16906, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.1469.

Gage, P. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, feb 1994. ISSN 0898-9788. URL https://dl.acm.org/doi/10.5555/177910.177914.

Gallé, M. Investigating the effectiveness of BPE: The power of shorter sequences. In Inui, K., Jiang, J., Ng, V., and Wan, X. (eds.), *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, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1141. URL https://aclanthology.org/D19-1141.

Garey, M. R., Johnson, D. S., and Stockmeyer, L. Some simplified NP-complete problems. In *Proceedings of the Sixth Annual ACM Symposium on Theory of Computing*, STOC '74, pp. 47–63, New York, NY, USA, 1974. Association for Computing Machinery. ISBN 9781450374231. doi: 10.1145/800119.803884. URL https://doi.org/10.1145/800119.803884.

Geh, R., Zhang, H., Ahmed, K., Wang, B., and Van Den Broeck, G. Where is the signal in tokenization space? In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3966–3979, Miami, Florida, USA, November 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.emnlp-main.230.

Giulianelli, M., Malagutti, L., Gastaldi, J. L., DuSell, B., Vieira, T., and Cotterell, R. On the proper treatment of tokenization in psycholinguistics. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Miami, Florida, USA, 2024. Association for Computational Linguistics. URL https://arxiv.org/abs/2410.02691.

Goldman, O., Caciularu, A., Eyal, M., Cao, K., Szpektor, I., and Tsarfaty, R. Unpacking tokenization: Evaluating text compression and its correlation with model performance. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 2274–2286, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.134. URL https://aclanthology.org/2024.findings-acl.134.

Gowda, T. and May, J. Finding the optimal vocabulary size for neural machine translation. In Cohn, T., He, Y., and Liu, Y. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 3955–3964, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.352. URL https://aclanthology.org/2020.findings-emnlp.352.

Kempa, D. and Prezza, N. At the roots of dictionary compression: string attractors. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pp. 827–840, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450355599. doi: 10.1145/3188745.3188814.

Kieffer, J. and Yang, E.-H. Grammar-based codes: a new class of universal lossless source codes. *IEEE Transactions on Information Theory*, 46(3):737–754, 2000. doi: 10.1109/18.841160.

Kociumaka, T., Navarro, G., and Prezza, N. Toward a definitive compressibility measure for repetitive sequences. *IEEE Transactions on Information Theory*, 69(4):2074–2092, 2023. doi: 10.1109/TIT.2022.3224382.

Kozma, L. and Voderholzer, J. Theoretical analysis of byte-pair encoding, 2024. URL https://arxiv.org/abs/2411.08671.

Kudo, T. Subword regularization: Improving neural network translation models with multiple subword candidates. In Gurevych, I. and Miyao, Y. (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1007. URL https://aclanthology.org/P18-1007.

Lim, J. P., Choo, D., and Lauw, H. W. A partition cover approach to tokenization, 2025. URL https://arxiv.org/abs/2501.06246.

Liu, A., Hayase, J., Hofmann, V., Oh, S., Smith, N. A., and Choi, Y. SuperBPE: Space travel for language models, 2025. URL https://arxiv.org/abs/2503.13423.

Lohrey, M. Algorithmics on SLP-compressed strings: A survey. *Groups - Complexity - Cryptology*, 4(2):241–299, 2012. doi: doi:10.1515/gcc-2012-0016. URL https://doi.org/10.1515/gcc-2012-0016.

Muffo, M., Cocco, A., and Bertino, E. Evaluating transformer language models on arithmetic operations using number decomposition. In Calzolari, N., Béchet, F., Blache, P., Choukri, K., Cieri, C., Declerck, T., Goggi, S., Isahara, H., Maegaard, B., Mariani, J., Mazo, H., Odijk, J., and Piperidis, S. (eds.), *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pp. 291–297, Marseille, France, June 2022. European Language Resources Association. URL https://aclanthology.org/2022.lrec-1.30.

Nogueira, R., Jiang, Z., and Lin, J. Investigating the limitations of transformers with simple arithmetic tasks, 2021. URL https://arxiv.org/abs/2102.13019.

OpenAI. GPT-4 technical report, 2023. URL https://arxiv.org/abs/2303.08774.

Pagnoni, A., Pasunuru, R., Rodriguez, P., Nguyen, J., Muller, B., Li, M., Zhou, C., Yu, L., Weston, J., Zettlemoyer, L., Ghosh, G., Lewis, M., Holtzman, A., and Iyer, S. Byte latent transformer: Patches scale better than tokens, 2024. URL https://arxiv.org/abs/2412.09871.

Petrov, A., Malfa, E. L., Torr, P., and Bibi, A. Language model tokenizers introduce unfairness between languages. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=78yDLKi95p.

Phan, B., Havasi, M., Muckley, M., and Ullrich, K. Understanding and mitigating tokenization bias in language models, 2024. URL https://arxiv.org/abs/2406.16829.

Pimentel, T. and Meister, C. How to compute the probability of a word. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, Miami, Florida, USA, 2024. Association for Computational Linguistics. URL https://arxiv.org/abs/2406.14561.

Rust, P., Pfeiffer, J., Vulić, I., Ruder, S., and Gurevych, I. How good is your tokenizer? On the monolingual performance of multilingual language models. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 3118–3135, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.243. URL https://aclanthology.org/2021.acl-long.243.

Schäfer, A., Hofmann, T., Schlag, I., and Pimentel, T. On the effect of (near) duplicate subwords in language modelling. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 9580–9597, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.571. URL https://aclanthology.org/2024.findings-acl.571/.

Schmidt, C. W., Reddy, V., Zhang, H., Alameddine, A., Uzan, O., Pinter, Y., and Tanner, C. Tokenization is more than compression. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 678–702, Miami, Florida, USA, November 2024. Association for Computational Linguistics. URL https://aclanthology.org/2024.emnlp-main.40.

Sennrich, R., Haddow, B., and Birch, A. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162.

Singh, A. K. and Strouse, D. Tokenization counts: the impact of tokenization on arithmetic in frontier LLMs, 2024. URL https://arxiv.org/abs/2402.14903.

Tao, C., Liu, Q., Dou, L., Muennighoff, N., Wan, Z., Luo, P., Lin, M., and Wong, N. Scaling laws with vocabulary: Larger models deserve larger vocabularies. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=sKCKPr8cRL.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. LLaMA: Open and efficient foundation language models, 2023a. URL https://arxiv.org/abs/2302.13971.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023b. URL https://arxiv.org/abs/2307.09288.

Yu, L., Simig, D., Flaherty, C., Aghajanyan, A., Zettlemoyer, L., and Lewis, M. MEGABYTE: Predicting million-byte sequences with multiscale transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=JTmO2V9Xpz.

Zouhar, V., Meister, C., Gastaldi, J., Du, L., Sachan, M., and Cotterell, R. Tokenization and the noiseless channel. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5184–5207, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.284. URL https://aclanthology.org/2023.acl-long.284.

Zouhar, V., Meister, C., Gastaldi, J., Du, L., Vieira, T., Sachan, M., and Cotterell, R. A formal perspective on byte-pair encoding. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 598–614, Toronto, Canada, July 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.38. URL https://aclanthology.org/2023.findings-acl.38.

A. Proof of Lemma 2

Lemma 2. *If a* `max-2-SAT` *instance is satisfiable, then the direct tokenisation instance output by Reduction 1 is also satisfiable. Formally:*

$$\mathrm{M2S}(\mathcal{X}, \mathcal{C}, \gamma) \implies \mathrm{Tok}_{\Rightarrow}(\mathrm{R1}(\mathcal{X}, \mathcal{C}, \gamma)) \tag{14}$$

Proof. First, note that if $\mathrm{M2S}(\mathcal{X}, \mathcal{C}, \gamma)$, then we have that Eq. (7) holds: $\gamma \leq \max_{\chi\in\{\mathtt{F},\mathtt{T}\}^J} \sum_{i=1}^{I} \mathbb{1}\{L_i^1 \vee L_i^2\}$. Now, without loss of generality, let a satisfying solution have values $x_j^\star$. In this case, for each variable X_j, we construct token $\circledcirc x_j^{\mathtt{T}} \circledcirc$ if $x_j^\star$ is true, or $\circledcirc x_j^{\mathtt{F}} \circledcirc$ if $x_j^\star$ is false. This gives us a total of J new tokens, so satisfies the $|\mathcal{S}| = |\Sigma| + K$ condition. Now we just need to count the symbols output by this solution to see if Eq. (9) is satisfied, since any given tokenisation $\mathtt{tok}(\cdot, \mathcal{S})$ will provide an upper bound on the optimal tokenisation in terms of compression:

$$\sum_{\mathbf{c}\in\mathcal{D}} |\mathtt{tok}_{\Rightarrow}[\mathcal{S}](\mathbf{c})| \geq \min_{\mathcal{S}'\subset\Sigma^+} \sum_{\mathbf{c}\in\mathcal{D}} |\mathtt{tok}_{\Rightarrow}[\mathcal{S}'](\mathbf{c})| \tag{29}$$
$$\text{s.t. } |\mathcal{S}'| = |\Sigma| + K$$

For each pair of strings $\circledcirc x_j^{\mathtt{T}} \circledcirc$ and $\circledcirc x_j^{\mathtt{F}} \circledcirc$ in $\mathcal{D}_1$, one is compressed into a single subword while the other is kept as originally—using 3 symbols. We thus have that the strings in $\mathcal{D}_1$ will occupy a total of $(1+3)J$ characters, and:

$$\sum_{\mathbf{c}\in(\bigcup_{\_=1}^{f} \mathcal{D}_1)} |\mathtt{tok}_{\Rightarrow}[\mathcal{S}](\mathbf{c})| = 4fJ \tag{30}$$

Similarly, for each string in $\mathcal{D}_2$ of form $\circledcirc x_j^{\mathtt{T}} \circledcirc x_j^{\mathtt{F}} \circledcirc$, we have that either token $\circledcirc x_j^{\mathtt{T}} \circledcirc$ or $\circledcirc x_j^{\mathtt{F}} \circledcirc$ exists. So each of these strings is compressed from 5 into 3 symbols. We thus have:

$$\sum_{\mathbf{c}\in(\bigcup_{\_=1}^{f'} \mathcal{D}_2)} |\mathtt{tok}_{\Rightarrow}[\mathcal{S}](\mathbf{c})| = 3f'J \tag{31}$$

Finally, we have strings in $\mathcal{D}_3$ of form $\circledcirc L_i^1 \circledcirc L_i^2 \circledcirc$. These strings will be compressed into 3 symbols if $\circledcirc L_i^1 \circledcirc$ or $\circledcirc L_i^2 \circledcirc$ (or both) exist, and kept with 5 symbols otherwise. We thus have:

$$\begin{aligned}
\sum_{\mathbf{c}\in\mathcal{D}_3} |\mathtt{tok}_{\Rightarrow}[\mathcal{S}](\mathbf{c})| &= \sum_{i=1}^{I} \left(5 - 2\,\mathbb{1}\left\{\begin{matrix} \circledcirc L_i^1 \circledcirc \in \mathcal{S} \\ \text{or} \\ \circledcirc L_i^2 \circledcirc \in \mathcal{S} \end{matrix}\right\}\right) && \text{(32a)} \\
&= 5I - 2\sum_{i=1}^{I} \mathbb{1}\left\{\begin{matrix} \circledcirc x_j^{\mathtt{T}} \circledcirc \in \mathcal{S} \text{ and } L_i^1 = X_j \\ \text{or} \\ \circledcirc x_j^{\mathtt{F}} \circledcirc \in \mathcal{S} \text{ and } L_i^1 = \neg X_j \\ \text{or} \\ \circledcirc x_{j'}^{\mathtt{T}} \circledcirc \in \mathcal{S} \text{ and } L_i^2 = X_{j'} \\ \text{or} \\ \circledcirc x_{j'}^{\mathtt{F}} \circledcirc \in \mathcal{S} \text{ and } L_i^2 = \neg X_{j'} \end{matrix}\right\} && \text{(32b)} \\
&= 5I - 2\sum_{i=1}^{I} \mathbb{1}_{\chi^\star}\{L_i^1 \vee L_i^2\} && \text{(32c)} \\
&\leq 5I - 2\gamma && \text{(32d)}
\end{aligned}$$

where, by construction, we have that a subword $\circledcirc L_i \circledcirc \in \mathcal{S}$ if and only if its associated variable (X_j or $\neg X_j$) is true. Summing together the lengths in Eqs. (30) to (32), we get that

$$\sum_{\mathbf{c}\in\mathcal{D}} |\mathtt{tok}_{\Rightarrow}[\mathcal{S}](\mathbf{c})| \leq \delta = (4f + 3f')\,J + 5\,I - 2\gamma \tag{33}$$

which concludes the proof. □

B. Proof of Lemma 3

Lemma 3. *If the direct tokenisation instance output by Reduction 1 is satisfiable, the* `max-2-SAT` *instance which generated it is as well. Formally:*

$$\mathrm{Tok}_{\Rightarrow}(\mathrm{R1}(\mathcal{X}, \mathcal{C}, \gamma)) \implies \mathrm{M2S}(\mathcal{X}, \mathcal{C}, \gamma) \tag{15}$$

Proof. First, note that the dataset $\mathcal{D}$ output by Reduction 1 has a total number of characters:

$$\sum_{\mathbf{c} \in \mathcal{D}} |\mathbf{c}| = (6f + 5f')J + 5I \tag{34}$$

Further, let:

$$\texttt{toklen}(\mathcal{D}, \mathcal{S}) \overset{\text{def}}{=} \sum_{\mathbf{c} \in \mathcal{D}} |\texttt{tok}_{\Rightarrow}[\mathcal{S}](\mathbf{c})|, \qquad \mathcal{S}_0 = \Sigma \cup \bigcup_{j=1}^{J} \{\circledcirc x_j^{\mathtt{T}} \circledcirc, \circledcirc x_j^{\mathtt{F}} \circledcirc\} \tag{35}$$

The maximum number of symbols in this dataset after compression is set to $\delta = (4f + 3f')\,J + 5\,I - 2\gamma$. This means that, to satisfy this objective, there must exist a vocabulary whose tokeniser compresses the text by at least $(2f + 2f')\,J + 2\gamma$ symbols. We now prove this lemma in three steps: $\textcircled{1}$ we show that any solution which compresses the text by at least $2fJ$ symbols must only have nontrivial subwords[14] of the form $\circledcirc x_j^{\mathtt{T}} \circledcirc$ or $\circledcirc x_j^{\mathtt{F}} \circledcirc$; $\textcircled{2}$ we show that any solution which compresses the text by at least $(2f + 2f')J$ symbols must only have either subword $\circledcirc x_j^{\mathtt{T}} \circledcirc$ or $\circledcirc x_j^{\mathtt{F}} \circledcirc$ for any variable X_j; $\textcircled{3}$ we show that any solution which compresses the text by at least $(2f + 2f')J + 2\gamma$ symbols must be produced by a `max-2-SAT` instance which has at least γ clauses that are simultaneously satisfiable. □

LemmaProofStep 1. (Step $\textcircled{1}$). *Any solution which compresses the text by at least $2fJ$ symbols must only have nontrivial subwords of the form $\circledcirc x_j^{\mathtt{T}} \circledcirc$ or $\circledcirc x_j^{\mathtt{F}} \circledcirc$, i.e.,:*

$$\left(\texttt{toklen}(\mathcal{D}, \mathcal{S}) \leq \underbrace{(4f + 5f')J + 5I}_{\sum_{\mathbf{c} \in \mathcal{D}} |\mathbf{c}| - 2fJ}\right) \implies \mathcal{S} \subset \mathcal{S}_0 \tag{36}$$

Proof. First, given a solution with $\mathcal{S} \subset \mathcal{S}_0$, each subword $s \in \mathcal{S}_{\setminus\Sigma}$, where $\mathcal{S}_{\setminus\Sigma} \overset{\text{def}}{=} \mathcal{S} \setminus \Sigma$, will replace at least f strings in $\mathcal{D}_1$—i.e., with form $\circledcirc x_j^{\mathtt{T}} \circledcirc$ or $\circledcirc x_j^{\mathtt{F}} \circledcirc$—for a single subword, thus saving $2f$ characters. Since we have $|\mathcal{S}_{\setminus\Sigma}| = K = J$ tokens, we save exactly $2fJ$ symbols in $\mathcal{D}_1$:

$$\mathcal{S} \subset \mathcal{S}_0 \implies \left(\texttt{toklen}(\mathcal{D}_1, \mathcal{S}') = \underbrace{4fJ}_{\sum_{\mathbf{c} \in \mathcal{D}_1} |\mathbf{c}| - 2fJ}\right) \tag{37}$$

Note now that any solution $\mathcal{S}'$ for which $\mathcal{S}' \not\subset \mathcal{S}_0$ has at least one nontrivial subword which is not of the form $\circledcirc x_j^{\mathtt{T}} \circledcirc$ or $\circledcirc x_j^{\mathtt{F}} \circledcirc$; this subword $s \notin \mathcal{S}_0$ will thus not compress strings in $\mathcal{D}_1$ by $2f$ symbols, but by at most f symbols:

$$\mathcal{S}' \not\subset \mathcal{S}_0 \implies \left(\texttt{toklen}(\mathcal{D}_1, \mathcal{S}') \geq \underbrace{4f(J-1) + 5f}_{\sum_{\mathbf{c} \in \mathcal{D}_1} |\mathbf{c}| - 2fJ + f}\right) \tag{38}$$

Even if this new subword were able to fully compress strings in $\mathcal{D}_2$ and $\mathcal{D}_3$ to a single symbol each, it would reach a compression of at most $4f'J + 4I$. Since by design $f = 4f'J + 4I + 1$, we get that:

$$\mathcal{S}' \not\subset \mathcal{S}_0 \implies \left(\texttt{toklen}(\mathcal{D}, \mathcal{S}') \geq 4fJ + f + f'J + I > (4f + 5f')J + 5I\right) \tag{39}$$

which concludes this step of the proof. □

[14] We define nontrivial subwords as subwords with more than one character. Remember that by definition $\Sigma \subseteq \mathcal{S}$, so all characters are always included in tokenisers' vocabularies. Also note that $|\mathcal{S}| = |\Sigma| + K$, so those trivial subwords are not counted towards vocabulary size K.

LemmaProofStep 2. (Step ②). *Any solution which compresses the text by at least $(2f + 2f')J$ symbols must only have either subword $\circledcirc x_j^{\mathtt{T}} \circledcirc$ or $\circledcirc x_j^{\mathtt{F}} \circledcirc$ for any variable X_j, i.e.,:*

$$\left(\mathtt{toklen}(\mathcal{D}, \mathcal{S}) \leq \underbrace{(4f + 3f')J + 5I}_{\sum_{\mathbf{c} \in \mathcal{D}} |\mathbf{c}| - (2f + 2f')J} \right) \implies \forall_{j \in \{1,\dots,J\}} \; |\mathcal{S} \cap \{\circledcirc x_j^{\mathtt{T}} \circledcirc, \circledcirc x_j^{\mathtt{F}} \circledcirc\}| = 1 \tag{40}$$

Proof. In this step of the proof, we show that satisfying solutions must have one and only one of subwords $\circledcirc x_j^{\mathtt{T}} \circledcirc$ and $\circledcirc x_j^{\mathtt{F}} \circledcirc$ for any variable X_j. As before, it's easy to see that a solution of the form described achieves at least $(2f + 2f')J$ symbol compression. Each subword of form $\circledcirc x_j^{\mathtt{T}} \circledcirc$ or $\circledcirc x_j^{\mathtt{F}} \circledcirc$ saves exactly $2f$ characters in the strings in $\mathcal{D}_1$. Further, because we always have either subword $\circledcirc x_j^{\mathtt{T}} \circledcirc$ or $\circledcirc x_j^{\mathtt{F}} \circledcirc$ for each value of j, we also get $2f'$ compression in the strings in $\mathcal{D}_2$:

$$\begin{aligned} &\forall_{j \in \{1,\dots,J\}} \; |\mathcal{S} \cap \{\circledcirc x_j^{\mathtt{T}} \circledcirc, \circledcirc x_j^{\mathtt{F}} \circledcirc\}| = 1 \\ &\qquad \implies \left(\mathtt{toklen}(\mathcal{D}_1, \mathcal{S}) = \underbrace{4fJ}_{\sum_{\mathbf{c} \in \mathcal{D}_1} |\mathbf{c}| - 2fJ} \right) \text{ and } \left(\mathtt{toklen}(\mathcal{D}_2, \mathcal{S}) = \underbrace{3f'J}_{\sum_{\mathbf{c} \in \mathcal{D}_2} |\mathbf{c}| - 2f'J} \right) \end{aligned} \tag{41}$$

Now note that this is not true if both $\circledcirc x_j^{\mathtt{T}} \circledcirc$ and $\circledcirc x_j^{\mathtt{F}} \circledcirc$ exist for a single j; in this case, only one of the tokens can be applied to $\circledcirc x_j^{\mathtt{T}} \circledcirc x_j^{\mathtt{F}} \circledcirc$, and thus both tokens together lead to a benefit of 2 instead of 4. If both $\circledcirc x_j^{\mathtt{T}} \circledcirc$ and $\circledcirc x_j^{\mathtt{F}} \circledcirc$ exist for any token X_j, this implies that neither of $\circledcirc x_{j'}^{\mathtt{T}} \circledcirc$ and $\circledcirc x_{j'}^{\mathtt{F}} \circledcirc$ exists for some other $X_{j'}$, resulting in an uncompressed string. Then, we get at most a compression of $2fJ + 2f'(J-1) + 4I$:

$$\exists_{j \in \{1,\dots,J\}} \; |\mathcal{S}' \cap \{\circledcirc x_j^{\mathtt{T}} \circledcirc, \circledcirc x_j^{\mathtt{F}} \circledcirc\}| \neq 1 \implies \left(\mathtt{toklen}(\mathcal{D}, \mathcal{S}') \geq \underbrace{(4f + 3f')J + 2f' + I}_{\sum_{\mathbf{c} \in \mathcal{D}} |\mathbf{c}| - (2f+2f)'J + 2f' - 4I} \right) \tag{42}$$

By construction $f' = 2I + 1$, which leads to:

$$\exists_{j \in \{1,\dots,J\}} \; |\mathcal{S}' \cap \{\circledcirc x_j^{\mathtt{T}} \circledcirc, \circledcirc x_j^{\mathtt{F}} \circledcirc\}| \neq 1 \implies \left(\mathtt{toklen}(\mathcal{D}, \mathcal{S}') > (4f + 3f')J + 5I \right) \tag{43}$$

This concludes the proof. □

LemmaProofStep 3. (Step ③). *Any instance of the tokenisation problem with a solution which compresses the text by at least $(2f + 2f')J + 2\gamma$ symbols must be produced by a* `max-2-SAT` *problem with at least γ satisfied clauses, i.e.,:*

$$\left(\mathtt{toklen}(\mathcal{D}, \mathcal{S}) \leq \underbrace{(4f + 3f')J + 5I - 2\gamma}_{\sum_{\mathbf{c} \in \mathcal{D}} |\mathbf{c}| - (2f + 2f')J + 2\gamma} \right) \implies \mathrm{M2S}(\mathcal{X}, \mathcal{C}, \gamma) \tag{44}$$

Proof. Finally, we now know any solution with this compression must have—for any variable X_j—either subword $\circledcirc x_j^{\mathtt{T}} \circledcirc$ or $\circledcirc x_j^{\mathtt{F}} \circledcirc$. We can thus create a bijection $\mathrm{Conv}_{\mathcal{S} \to \chi}$ between the set of possible vocabularies respecting this condition, and the set of T/F assignments to SAT variables χ:

$$\mathrm{Conv}_{\mathcal{S} \to \chi}(\mathcal{S}) = \left\{ \begin{array}{ll} \mathtt{T} & \mathtt{if}\ \circledcirc x_j^{\mathtt{T}} \circledcirc \in \mathcal{S} \\ \mathtt{F} & \mathtt{if}\ \circledcirc x_j^{\mathtt{F}} \circledcirc \in \mathcal{S} \end{array} \right\}_{j=1}^{J} \tag{45}$$

Further, note that vocabularies of this form (as shown in Eq. (41)) lead to exactly $(2f + 2f')J$ symbols being compressed in $\mathcal{D}_1$ and $\mathcal{D}_2$. To achieve the target compression, a solution must thus compress $\mathcal{D}_3$ by at least 2γ symbols. Now note that for any string $\circledcirc L_i^1 \circledcirc L_i^2 \circledcirc$ in $\mathcal{D}_3$ we have three compression options: $\circledcirc L_i^1 \circledcirc$ will be compressed, saving 2 symbols; $\circledcirc L_i^2 \circledcirc$ will be compressed, also saving 2 symbols; or nothing will be compressed. More specifically, $\circledcirc L_i^1 \circledcirc$ can be compressed if L_i^1 represents X_j and subword $\circledcirc x_j^{\mathtt{T}} \circledcirc$ exists, or if L_i^1 represents $\neg X_j$ and subword $\circledcirc x_j^{\mathtt{F}} \circledcirc$ exists; the same is true for $\circledcirc L_i^2 \circledcirc$. They cannot both be compressed, however, as there is only one symbol $\circledcirc$ between the literals. We thus get a compression of 2 symbols for each of these strings if at least one of its literals has an associated subword in $\mathcal{S}$. Note thus that whenever a string $\circledcirc L_i^1 \circledcirc L_i^2 \circledcirc$ is compressed by 2 symbols using vocabulary $\mathcal{S}$, the `max-2-SAT` disjunction $L_i^1 \vee L_i^2$ will also be satisfied by assignment $\chi = \mathrm{Conv}_{\mathcal{S} \to \chi}(\mathcal{S})$; similarly, whenever this string suffers no compression (i.e., having a compression of zero), the max-2-SAT disjunction will not be satisfied. As our condition assumes a compression of at least 2γ symbols, we know that we have at least γ strings for which a literal has an associated subword. We can thus write:

$$
\begin{aligned}
2\gamma &\leq \max_{\mathcal{S}\subset\Sigma^+} \sum_{\mathbf{c}\in\mathcal{D}_3} |\mathbf{c}| - |\mathtt{tok}_{\Rightarrow}[\mathcal{S}](\mathbf{c})| && \text{(46a)}\\
&\quad \text{s.t. } |\mathcal{S}| = |\Sigma| + J \ \mathtt{and}\ \forall_{j\in\{1,\dots,J\}}\ |\mathcal{S} \cap \{\circledcirc x_j^{\mathtt{T}}\circledcirc, \circledcirc x_j^{\mathtt{F}}\circledcirc\}| = 1\\
&= \max_{\mathcal{S}\subset\Sigma^+} \sum_{\circledcirc L_i^1 \circledcirc L_i^2 \circledcirc\, \in \mathcal{D}_3} 2\,\mathbb{1}\left\{\begin{array}{c} \circledcirc L_i^1 \circledcirc \in \mathcal{S} \\ \text{or} \\ \circledcirc L_i^2 \circledcirc \in \mathcal{S} \end{array}\right\} && \text{(46b)}\\
&\quad \text{s.t. } |\mathcal{S}| = |\Sigma| + J \ \mathtt{and}\ \forall_{j\in\{1,\dots,J\}}\ |\mathcal{S} \cap \{\circledcirc x_j^{\mathtt{T}}\circledcirc, \circledcirc x_j^{\mathtt{F}}\circledcirc\}| = 1\\
&= \max_{\chi\in\{0,1\}^J} \sum_{i=1}^{I} 2\mathbb{1}_\chi\{L_i^1 \vee L_i^2\} && \text{(46c)}\\
&\implies \mathrm{M2S}(\mathcal{X},\mathcal{C},\gamma) && \text{(46d)}
\end{aligned}
$$

We thus know that, if a satisfying tokenisation solution exists, then the associated max-2-SAT problem will also be satisfiable. This concludes the proof. □

C. Proof of Lemma 6

Lemma 6. *If a max-2-SAT instance is satisfiable, then the bottom-up tokenisation instance output by Reduction 2 is also satisfiable. Formally:*

$$
\mathrm{M2S}(\mathcal{X},\mathcal{C},\gamma) \implies \mathrm{Tok}_{\uparrow}(\mathrm{R2}(\mathcal{X},\mathcal{C},\gamma)) \tag{23}
$$

Proof. Our proof starts by first defining the three following lists of merges, which will be included in any satisfying solution to the tokenisation problem:

$$
\mathbf{m}_1 = \bigcirc_{j=1}^{J} \left[\langle \otimes, x_j^{\mathtt{F}}\rangle, \langle x_j^{\mathtt{T}}, \otimes\rangle\right], \quad \mathbf{m}_3 = \bigcirc_{j=1}^{J} \left[\langle x_j^{\mathtt{F}}, \circledcirc\rangle, \langle \circledcirc, x_j^{\mathtt{T}}\rangle\right], \quad \mathbf{m}_5 = \bigcirc_{j=1}^{J} \left[\langle \circledcirc, x_j^{\mathtt{F}}\rangle, \langle x_j^{\mathtt{T}}, \circledcirc\rangle\right] \tag{47}
$$

Now, without loss of generality, let a satisfying solution to max-2-SAT have values $x_j^\star$. We then construct two other lists of merges, which depend on this max-2-SAT solution:

$$
\mathbf{m}_2 = \bigcirc_{j=1}^{J} \left[\begin{array}{lr} \langle \circledcirc, x_j^{\mathtt{T}}\otimes\rangle & \mathtt{if}\ x_j^\star = \mathtt{T} \\ \langle \otimes x_j^{\mathtt{F}}, \circledcirc\rangle & \mathtt{else} \end{array}\right], \qquad \mathbf{m}_4 = \bigcirc_{j=1}^{J} \left[\begin{array}{lr} \langle \circledcirc x_j^{\mathtt{T}}, \circledcirc\rangle & \mathtt{if}\ x_j^\star = \mathtt{T} \\ \langle \circledcirc, x_j^{\mathtt{F}}\circledcirc\rangle & \mathtt{else} \end{array}\right] \tag{48}
$$

In words, we create merges $\langle \circledcirc, x_j^{\mathtt{T}}\otimes\rangle$ and $\langle \circledcirc x_j^{\mathtt{T}}, \circledcirc\rangle$ if $x_j^\star$ is true, or $\langle \otimes x_j^{\mathtt{F}}, \circledcirc\rangle$ and $\langle \circledcirc, x_j^{\mathtt{F}}\circledcirc\rangle$ if $x_j^\star$ is false. We then create a merge sequence by concatenating these lists in order:

$$
\mathbf{m} = \mathbf{m}_1 \circ \mathbf{m}_2 \circ \mathbf{m}_3 \circ \mathbf{m}_4 \circ \mathbf{m}_5 \tag{49}
$$

This gives us a total of $|\mathbf{m}| = K = 8J$ merges. Now we just need to count the symbols output by this solution to see if Eq. (17) is satisfied, since any given tokenisation $\mathtt{tok}_{\uparrow}[\mathbf{m}]$ will provide an upper bound on the optimal tokenisation in terms of compression:

$$
\sum_{\mathbf{c}\in\mathcal{D}} |\mathtt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})| \geq \min_{\mathbf{m}'\in\mathcal{M}^*} \sum_{\mathbf{c}\in\mathcal{D}} |\mathtt{tok}_{\uparrow}[\mathbf{m}'](\mathbf{c})| \quad \text{s.t. } |\mathbf{m}'| = K \tag{50}
$$

By applying the merges $\mathbf{m}$, each string in $\mathcal{D}_1$ will be compressed into a single subword; note that $\mathbf{m}_2$ and $\mathbf{m}_4$ will have no effect on these strings. This is easy to see by applying merges sequentially to these strings, as displayed in Tab. 1. leading to:

$$
\sum_{\mathbf{c}\in(\bigcup_{=1}^{f}\mathcal{D}_1)} |\mathtt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})| = 6fJ \tag{51}
$$

$\mathbf{c}$	$\texttt{tok}_\uparrow[\mathbf{m}_1](\mathbf{c})$	$\texttt{tok}_\uparrow[\mathbf{m}_1 \circ \mathbf{m}_2 \circ \mathbf{m}_3](\mathbf{c})$	$\texttt{tok}_\uparrow[\mathbf{m}_1 \circ \mathbf{m}_2 \circ \mathbf{m}_3 \circ \mathbf{m}_4 \circ \mathbf{m}_5](\mathbf{c})$	$\lvert\texttt{tok}_\uparrow[\mathbf{m}](\mathbf{c})\rvert$
$\langle \circledcirc, x_j^{\texttt{T}} \rangle$	$\cdot$	$\langle \circledcirc x_j^{\texttt{T}} \rangle$	$\cdot$	1
$\langle x_j^{\texttt{F}}, \circledcirc \rangle$	$\cdot$	$\langle x_j^{\texttt{F}} \circledcirc \rangle$	$\cdot$	1
$\langle x_j^{\texttt{T}}, \circledcirc \rangle$	$\cdot$	$\cdot$	$\langle x_j^{\texttt{T}} \circledcirc \rangle$	1
$\langle \circledcirc, x_j^{\texttt{F}} \rangle$	$\cdot$	$\cdot$	$\langle \circledcirc x_j^{\texttt{F}} \rangle$	1
$\langle x_j^{\texttt{T}}, \otimes \rangle$	$\langle x_j^{\texttt{T}} \otimes \rangle$	$\cdot$	$\cdot$	1
$\langle \otimes, x_j^{\texttt{F}} \rangle$	$\langle \otimes x_j^{\texttt{F}} \rangle$	$\cdot$	$\cdot$	1

Table 1: Example of applying $\mathbf{m}$ in $\mathcal{D}_1$ of bottom-up tokenisation problem obtained from Reduction 2. The dot symbol $\cdot$ denotes the string not changing under the given merge.

$\mathbf{c}$	$\texttt{tok}_\uparrow[\mathbf{m}_1](\mathbf{c})$	$\texttt{tok}_\uparrow[\mathbf{m}_1 \circ \mathbf{m}_2](\mathbf{c})$		$\texttt{tok}_\uparrow[\mathbf{m}_1 \circ \mathbf{m}_2 \circ \mathbf{m}_3](\mathbf{c})$	$\texttt{tok}_\uparrow[\mathbf{m}_1 \circ \mathbf{m}_2 \circ \mathbf{m}_3 \circ \mathbf{m}_4](\mathbf{c})$		$\lvert\texttt{tok}_\uparrow[\mathbf{m}](\mathbf{c})\rvert$	
		$x_j^\star = \texttt{T}$	$x_j^\star = \texttt{F}$		$x_j^\star = \texttt{T}$	$x_j^\star = \texttt{F}$	$x_j^\star = \texttt{T}$	$x_j^\star = \texttt{F}$
$\langle \circledcirc, x_j^{\texttt{T}}, \circledcirc \rangle$	$\cdot$	$\cdot$	$\cdot$	$\langle \circledcirc x_j^{\texttt{T}}, \circledcirc \rangle$	$\langle \circledcirc x_j^{\texttt{T}} \circledcirc \rangle$	$\langle \circledcirc x_j^{\texttt{T}}, \circledcirc \rangle$	1	2
$\langle \circledcirc, x_j^{\texttt{F}}, \circledcirc \rangle$	$\cdot$	$\cdot$	$\cdot$	$\langle \circledcirc, x_j^{\texttt{F}} \circledcirc \rangle$	$\langle \circledcirc, x_j^{\texttt{F}} \circledcirc \rangle$	$\langle \circledcirc x_j^{\texttt{F}} \circledcirc \rangle$	2	1
$\langle \circledcirc, x_j^{\texttt{T}}, \otimes \rangle$	$\langle \circledcirc, x_j^{\texttt{T}} \otimes \rangle$	$\langle \circledcirc x_j^{\texttt{T}} \otimes \rangle$	$\langle \circledcirc, x_j^{\texttt{T}} \otimes \rangle$	$\cdot$	$\cdot$	$\cdot$	1	2
$\langle \otimes, x_j^{\texttt{F}}, \circledcirc \rangle$	$\langle \otimes x_j^{\texttt{F}}, \circledcirc \rangle$	$\langle \otimes x_j^{\texttt{F}}, \circledcirc \rangle$	$\langle \otimes x_j^{\texttt{F}} \circledcirc \rangle$	$\cdot$	$\cdot$	$\cdot$	2	1

Table 2: Example of applying $\mathbf{m}$ in $\mathcal{D}_2$ of bottom-up tokenisation problem obtained from Reduction 2. The dot symbol $\cdot$ denotes the string not changing under the given merge. $\mathbf{m}_5$ is not depicted as it does not affect this dataset.

For each pair of strings $\circledcirc x_j^{\texttt{T}} \circledcirc$ and $\circledcirc x_j^{\texttt{F}} \circledcirc$ in $\mathcal{D}_2$, one is compressed into a single subword while the other is only compressed to two subwords—the one with $x_j^{\texttt{T}}$ is compressed to a single symbol if $x_j^\star = \texttt{T}$ and the one with $x_j^{\texttt{F}}$ otherwise. The same is true for each pair of strings $\circledcirc x_j^{\texttt{T}} \otimes$ and $\otimes x_j^{\texttt{F}} \circledcirc$, also in $\mathcal{D}_2$. This is displayed in Tab. 2. We thus have that, for each variable X_j, the strings in $\mathcal{D}_2$ will occupy a total of $(1+2+1+2)J$ characters, and:

$$\sum_{\mathbf{c} \in (\bigcup_{=1}^{f} \mathcal{D}_2)} \lvert\texttt{tok}_\uparrow[\mathbf{m}](\mathbf{c})\rvert = 6f'J \tag{52}$$

Similarly, each string in $\mathcal{D}_3$ and $\mathcal{D}_4$ will be compressed into only 2 symbols after this tokeniser is applied to it. We thus have:

$$\sum_{\mathbf{c} \in (\bigcup_{=1}^{f''} \mathcal{D}_3)} \lvert\texttt{tok}_\uparrow[\mathbf{m}](\mathbf{c})\rvert = 4f''J, \qquad \sum_{\mathbf{c} \in (\bigcup_{=1}^{f'''} \mathcal{D}_4)} \lvert\texttt{tok}_\uparrow[\mathbf{m}](\mathbf{c})\rvert = 4f'''J \tag{53}$$

Finally, we have the strings in $\mathcal{D}_5$. These strings are constructed such that they will be compressed into 2 symbols if either L_i^1 or L_i^2 evaluates to $\texttt{T}$, and kept with 3 symbols otherwise; see Tab. 4 for a detailed simulation of why this is the case. We thus have:

$$\sum_{\mathbf{c} \in \mathcal{D}_5} \lvert\texttt{tok}_\uparrow[\mathbf{m}](\mathbf{c})\rvert = \sum_{i=1}^{I} \left(3 - 1\mathbb{1} \left\{ \begin{array}{c} L_i^1 = X_j \quad \texttt{and}\ \langle \circledcirc, x_j^{\texttt{T}} \otimes \rangle, \langle \circledcirc x_j^{\texttt{T}}, \circledcirc \rangle \in \mathbf{m} \\ \text{or} \\ L_i^1 = \neg X_j \ \texttt{and}\ \langle \otimes x_j^{\texttt{F}}, \circledcirc \rangle, \langle \circledcirc, x_j^{\texttt{F}} \circledcirc \rangle \in \mathbf{m} \\ \text{or} \\ L_i^2 = X_{j'} \quad \texttt{and}\ \langle \circledcirc, x_{j'}^{\texttt{T}} \otimes \rangle, \langle \circledcirc x_{j'}^{\texttt{T}}, \circledcirc \rangle \in \mathbf{m} \\ \text{or} \\ L_i^2 = \neg X_{j'} \ \texttt{and}\ \langle \otimes x_{j'}^{\texttt{F}}, \circledcirc \rangle, \langle \circledcirc, x_{j'}^{\texttt{F}} \circledcirc \rangle \in \mathbf{m} \end{array} \right\} \right) \tag{54a}$$

$$= 3I - \sum_{i=1}^{I} \mathbb{1}_{\chi^\star}\{L_i^1 \vee L_i^2\} \tag{54b}$$

$$\leq 3I - \gamma \tag{54c}$$

where, by construction, we have a merge in our sequence (e.g., $\langle \circledcirc, x_j^{\texttt{T}} \otimes \rangle$ or $\langle \otimes x_j^{\texttt{F}}, \circledcirc \rangle$) if and only if its value is in a satisfying assignment (e.g., $x_j^\star = \texttt{T}$ or $x_j^\star = \texttt{F}$ respectively). Summing together the lengths in Eqs. (51) to (54), we get that:

$$\sum_{\mathbf{c} \in \mathcal{D}} \lvert\texttt{tok}_\uparrow[\mathbf{m}](\mathbf{c})\rvert \leq \delta = (6f + 6f' + 4f'' + 4f''')\,J + 3\,I - \gamma \tag{55}$$

$\mathcal{D}$	$\mathbf{c}$	$\mathtt{tok}_{\uparrow}[\mathbf{m}_1](\mathbf{c})$	$\mathtt{tok}_{\uparrow}[\mathbf{m}_1 \circ \mathbf{m}_2](\mathbf{c})$		$\mathtt{tok}_{\uparrow}[\mathbf{m}_1 \circ \mathbf{m}_2 \circ \mathbf{m}_3](\mathbf{c})$		$\mathtt{tok}_{\uparrow}[\mathbf{m}_1 \circ \mathbf{m}_2 \circ \mathbf{m}_3 \circ \mathbf{m}_4](\mathbf{c})$		$\mathtt{tok}_{\uparrow}[\mathbf{m}_1 \circ \mathbf{m}_2 \circ \mathbf{m}_3 \circ \mathbf{m}_4 \circ \mathbf{m}_5](\mathbf{c})$		$\|\mathtt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})\|$
			$x_j^\star = \mathtt{T}$	$x_j^\star = \mathtt{F}$	$x_j^\star = \mathtt{T}$	$x_j^\star = \mathtt{F}$	$x_j^\star = \mathtt{T}$	$x_j^\star = \mathtt{F}$	$x_j^\star = \mathtt{T}$	$x_j^\star = \mathtt{F}$	
$\mathcal{D}_3$	$\langle \circledcirc, x_j^{\mathtt{T}}, \circledcirc, x_j^{\mathtt{F}}, \circledcirc \rangle$	$\cdot$	$\cdot$	$\cdot$	$\langle \circledcirc x_j^{\mathtt{T}}, \circledcirc, x_j^{\mathtt{F}} \circledcirc \rangle$		$\langle \circledcirc x_j^{\mathtt{T}} \circledcirc, x_j^{\mathtt{F}} \circledcirc \rangle$	$\langle \circledcirc x_j^{\mathtt{T}}, \circledcirc x_j^{\mathtt{F}} \circledcirc \rangle$	$\cdot$	$\cdot$	2
$\mathcal{D}_3$	$\langle \otimes, x_j^{\mathtt{F}}, \circledcirc, x_j^{\mathtt{T}}, \otimes \rangle$	$\langle \otimes x_j^{\mathtt{F}}, \circledcirc, x_j^{\mathtt{T}} \otimes \rangle$	$\langle \otimes x_j^{\mathtt{F}}, \circledcirc x_j^{\mathtt{T}} \otimes \rangle$	$\langle \otimes x_j^{\mathtt{F}} \circledcirc, x_j^{\mathtt{T}} \otimes \rangle$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	$\cdot$	2
$\mathcal{D}_4$	$\langle \circledcirc, x_j^{\mathtt{F}}, \circledcirc, x_j^{\mathtt{T}}, \otimes \rangle$	$\langle \circledcirc, x_j^{\mathtt{F}}, \circledcirc, x_j^{\mathtt{T}} \otimes \rangle$	$\langle \circledcirc, x_j^{\mathtt{F}}, \circledcirc x_j^{\mathtt{T}} \otimes \rangle$	$\cdot$	$\cdot$	$\langle \circledcirc, x_j^{\mathtt{F}} \circledcirc, x_j^{\mathtt{T}} \otimes \rangle$	$\cdot$	$\langle \circledcirc x_j^{\mathtt{F}} \circledcirc, x_j^{\mathtt{T}} \otimes \rangle$	$\langle \circledcirc x_j^{\mathtt{F}}, \circledcirc x_j^{\mathtt{T}} \otimes \rangle$	$\cdot$	2
$\mathcal{D}_4$	$\langle \otimes, x_j^{\mathtt{F}}, \circledcirc, x_j^{\mathtt{T}}, \circledcirc \rangle$	$\langle \otimes x_j^{\mathtt{F}}, \circledcirc, x_j^{\mathtt{T}}, \circledcirc \rangle$	$\cdot$	$\langle \otimes x_j^{\mathtt{F}} \circledcirc, x_j^{\mathtt{T}}, \circledcirc \rangle$	$\langle \otimes x_j^{\mathtt{F}}, \circledcirc x_j^{\mathtt{T}}, \circledcirc \rangle$	$\cdot$	$\langle \otimes x_j^{\mathtt{F}}, \circledcirc x_j^{\mathtt{T}} \circledcirc \rangle$	$\cdot$	$\cdot$	$\langle \otimes x_j^{\mathtt{F}} \circledcirc, x_j^{\mathtt{T}} \circledcirc \rangle$	2

Table 3: Example of applying $\mathbf{m}$ in $\mathcal{D}_3$ and $\mathcal{D}_4$ of the bottom-up tokenisation problem obtained from Reduction 2. The dot symbol $\cdot$ denotes the string not changing under the given merge.

Assignment	Condition	$\mathbf{c}$	$\mathtt{tok}_{\uparrow}[\mathbf{m}_1](\mathbf{c})$	$\mathtt{tok}_{\uparrow}[\mathbf{m}_1 \circ \mathbf{m}_2](\mathbf{c})$	$\mathtt{tok}_{\uparrow}[\mathbf{m}_1 \circ \mathbf{m}_2 \circ \mathbf{m}_3](\mathbf{c})$	$\mathtt{tok}_{\uparrow}[\mathbf{m}_1 \circ \mathbf{m}_2 \circ \mathbf{m}_3 \circ \mathbf{m}_4](\mathbf{c})$	$\|\mathtt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})\|$
$L_i^1 = X_j$ and $L_i^2 = \neg X_{j'}$	$x_j^\star = \mathtt{T} \wedge x_{j'}^\star = \mathtt{T}$	$\langle \circledcirc, x_j^{\mathtt{T}}, \circledcirc, x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	$\cdot$	$\cdot$	$\langle \circledcirc x_j^{\mathtt{T}}, \circledcirc, x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	$\langle \circledcirc x_j^{\mathtt{T}} \circledcirc, x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	2
	$x_j^\star = \mathtt{F} \wedge x_{j'}^\star = \mathtt{T}$		$\cdot$	$\cdot$		$\langle \circledcirc x_j^{\mathtt{T}}, \circledcirc, x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	3
	$x_j^\star = \mathtt{T} \wedge x_{j'}^\star = \mathtt{F}$		$\cdot$	$\cdot$		$\langle \circledcirc x_j^{\mathtt{T}} \circledcirc, x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	2
	$x_j^\star = \mathtt{F} \wedge x_{j'}^\star = \mathtt{F}$		$\cdot$	$\cdot$		$\langle \circledcirc x_j^{\mathtt{T}}, \circledcirc x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	2
$L_i^1 = \neg X_j$ and $L_i^2 = X_{j'}$	$x_j^\star = \mathtt{T} \wedge x_{j'}^\star = \mathtt{T}$	$\langle \circledcirc, x_{j'}^{\mathtt{T}}, \circledcirc, x_j^{\mathtt{F}}, \circledcirc \rangle$	$\cdot$	$\cdot$	$\langle \circledcirc x_{j'}^{\mathtt{T}}, \circledcirc, x_j^{\mathtt{F}} \circledcirc \rangle$	$\langle \circledcirc x_{j'}^{\mathtt{T}}, \circledcirc, x_j^{\mathtt{F}} \circledcirc \rangle$	2
	$x_j^\star = \mathtt{F} \wedge x_{j'}^\star = \mathtt{T}$		$\cdot$	$\cdot$		$\langle \circledcirc x_{j'}^{\mathtt{T}}, \circledcirc, x_j^{\mathtt{F}} \circledcirc \rangle$	2
	$x_j^\star = \mathtt{T} \wedge x_{j'}^\star = \mathtt{F}$		$\cdot$	$\cdot$		$\langle \circledcirc x_{j'}^{\mathtt{T}}, \circledcirc, x_j^{\mathtt{F}} \circledcirc \rangle$	3
	$x_j^\star = \mathtt{F} \wedge x_{j'}^\star = \mathtt{F}$		$\cdot$	$\cdot$		$\langle \circledcirc x_{j'}^{\mathtt{T}}, \circledcirc x_j^{\mathtt{F}} \circledcirc \rangle$	2
$L_i^1 = \neg X_j$ and $L_i^2 = \neg X_{j'}$	$x_j^\star = \mathtt{T} \wedge x_{j'}^\star = \mathtt{T}$	$\langle \otimes, x_j^{\mathtt{F}}, \circledcirc, x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	$\langle \otimes x_j^{\mathtt{F}}, \circledcirc, x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	$\cdot$	$\langle \otimes x_j^{\mathtt{F}}, \circledcirc, x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	$\cdot$	3
	$x_j^\star = \mathtt{F} \wedge x_{j'}^\star = \mathtt{T}$			$\langle \otimes x_j^{\mathtt{F}} \circledcirc, x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	$\langle \otimes x_j^{\mathtt{F}} \circledcirc, x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	$\cdot$	2
	$x_j^\star = \mathtt{T} \wedge x_{j'}^\star = \mathtt{F}$			$\cdot$	$\langle \otimes x_j^{\mathtt{F}}, \circledcirc, x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	$\langle \otimes x_j^{\mathtt{F}}, \circledcirc x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	2
	$x_j^\star = \mathtt{F} \wedge x_{j'}^\star = \mathtt{F}$			$\langle \otimes x_j^{\mathtt{F}} \circledcirc, x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	$\langle \otimes x_j^{\mathtt{F}} \circledcirc, x_{j'}^{\mathtt{F}}, \circledcirc \rangle$	$\cdot$	2
$L_i^1 = X_j$ and $L_i^2 = X_{j'}$	$x_j^\star = \mathtt{T} \wedge x_{j'}^\star = \mathtt{T}$	$\langle \circledcirc, x_j^{\mathtt{T}}, \circledcirc, x_{j'}^{\mathtt{T}}, \otimes \rangle$	$\langle \circledcirc, x_j^{\mathtt{T}}, \circledcirc, x_{j'}^{\mathtt{T}} \otimes \rangle$	$\langle \circledcirc, x_j^{\mathtt{T}}, \circledcirc x_{j'}^{\mathtt{T}} \otimes \rangle$	$\langle \circledcirc x_j^{\mathtt{T}}, \circledcirc x_{j'}^{\mathtt{T}} \otimes \rangle$	$\langle \circledcirc x_j^{\mathtt{T}}, \circledcirc x_{j'}^{\mathtt{T}} \otimes \rangle$	2
	$x_j^\star = \mathtt{F} \wedge x_{j'}^\star = \mathtt{T}$						2
	$x_j^\star = \mathtt{T} \wedge x_{j'}^\star = \mathtt{F}$			$\cdot$	$\langle \circledcirc x_j^{\mathtt{T}}, \circledcirc, x_{j'}^{\mathtt{T}} \otimes \rangle$	$\langle \circledcirc x_j^{\mathtt{T}} \circledcirc, x_{j'}^{\mathtt{T}} \otimes \rangle$	2
	$x_j^\star = \mathtt{F} \wedge x_{j'}^\star = \mathtt{F}$					$\cdot$	3

Table 4: Example of applying $\mathbf{m}$ in $\mathcal{D}_5$ of the bottom-up tokenisation problem obtained from Reduction 2. The dot symbol $\cdot$ denotes the string not changing under the given merge. $\mathbf{m}_5$ is not depicted as it does not affect this dataset.

which concludes the proof. $\square$

D. Proof of Lemma 7

Lemma 7. *If the bottom-up tokenisation instance output by Reduction 2 is satisfiable, the* $\mathtt{max\text{-}2\text{-}SAT}$ *instance which generated it is as well. Formally:*

$$\mathrm{Tok}_{\uparrow}(\mathrm{R2}(\mathcal{X}, \mathcal{C}, \gamma)) \implies \mathrm{M2S}(\mathcal{X}, \mathcal{C}, \gamma) \tag{27}$$

Proof. First, note that:

$$\sum_{\mathbf{c} \in \mathcal{D}} |\mathbf{c}| = (12f + 12f' + 10f'' + 10f''')\, J + 5\, I \tag{56}$$

Further, let:

$$\mathtt{toklen}(\mathcal{D}, \mathbf{m}) \stackrel{\text{def}}{=} \sum_{\mathbf{c} \in \mathcal{D}} |\mathtt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})| \tag{57}$$

$$\mathbf{m}_1 = \bigcirc_{j=1}^{J} [\langle \otimes, x_j^{\mathtt{F}} \rangle, \langle x_j^{\mathtt{T}}, \otimes \rangle], \qquad \mathbf{m}_3 = \bigcirc_{j=1}^{J} [\langle x_j^{\mathtt{F}}, \circledcirc \rangle, \langle \circledcirc, x_j^{\mathtt{T}} \rangle], \qquad \mathbf{m}_5 = \bigcirc_{j=1}^{J} [\langle \circledcirc, x_j^{\mathtt{F}} \rangle, \langle x_j^{\mathtt{T}}, \circledcirc \rangle]$$

$$\mathbf{m}_j^{\circledcirc} = \left\{ \begin{array}{l} \langle \circledcirc x_j^{\mathtt{T}}, \circledcirc \rangle, \langle \circledcirc, x_j^{\mathtt{F}} \circledcirc \rangle \\ \langle \circledcirc, x_j^{\mathtt{T}} \circledcirc \rangle, \langle \circledcirc x_j^{\mathtt{F}}, \circledcirc \rangle \end{array} \right\}, \qquad \mathbf{m}_j^{\otimes} = \left\{ \begin{array}{l} \langle \circledcirc, x_j^{\mathtt{T}} \otimes \rangle, \langle \otimes x_j^{\mathtt{F}}, \circledcirc \rangle \\ \langle \circledcirc x_j^{\mathtt{T}}, \otimes \rangle, \langle \otimes, x_j^{\mathtt{F}} \circledcirc \rangle \end{array} \right\}$$

$$\mathbf{m}_j^{\mathtt{T}} = \left\{ \begin{array}{l} \langle \circledcirc x_j^{\mathtt{T}}, \circledcirc \rangle, \langle \circledcirc, x_j^{\mathtt{T}} \otimes \rangle, \\ \langle \circledcirc, x_j^{\mathtt{T}} \circledcirc \rangle, \langle \circledcirc x_j^{\mathtt{T}}, \otimes \rangle, \end{array} \right\}, \qquad \mathbf{m}_j^{\mathtt{F}} = \left\{ \begin{array}{l} \langle \circledcirc, x_j^{\mathtt{F}} \circledcirc \rangle, \langle \otimes x_j^{\mathtt{F}}, \circledcirc \rangle, \\ \langle \circledcirc x_j^{\mathtt{F}}, \circledcirc \rangle, \langle \otimes, x_j^{\mathtt{F}} \circledcirc \rangle, \end{array} \right\}$$

The maximum number of symbols in this dataset after compression is set to $\delta = (6f + 6f' + 4f'' + 4f''')\, J + 3\, I - \gamma$. This means that to satisfy this objective, there must exist a vocabulary whose tokeniser compresses the text by at least $(6f + 6f' + 6f'' + 6f''')\, J + 2I + \gamma$ symbols. We now prove this lemma in five steps: ① we show that any solution which compresses the text by at least $6f J$ symbols must include all merges in $\mathbf{m}_1$, $\mathbf{m}_3$, and $\mathbf{m}_5$; ② we show that any solution which compresses the text by at least $(6f + 6f')J$ symbols must only include either merges in $\mathbf{m}_1$, $\mathbf{m}_3$, $\mathbf{m}_5$, or in either $\mathbf{m}_j^{\circledcirc}$ or $\mathbf{m}_j^{\otimes}$; ③ we show that any solution which compresses the text by at least $(6f + 6f' + 6f'')J$ symbols must include, for each $j \in \{1, \dots, J\}$, exactly one merge in set $\mathbf{m}_j^{\circledcirc}$ and one in set $\mathbf{m}_j^{\otimes}$; ④ we show that any solution which compresses the text by at least $(6f + 6f' + 6f'' + 6f''')J$ symbols must include, for each $j \in \{1, \dots, J\}$, exactly two merges in either set $\mathbf{m}_j^{\mathtt{T}}$ or in set $\mathbf{m}_j^{\mathtt{F}}$; ⑤ we show that any solution which compresses the text by at least $(6f + 6f' + 6f'' + 6f''')J + 2I + \gamma$ symbols must be produced by a `max-2-SAT` problem with at least γ satisfied clauses. □

LemmaProofStep 1. (Step ①). *Any solution which compresses the text by at least $6fJ$ symbols must include all merges in $\mathbf{m}_1$, $\mathbf{m}_3$, and $\mathbf{m}_5$, i.e.,:*

$$\left(\texttt{toklen}(\mathcal{D}, \mathbf{m}) \leq \underbrace{6fJ + (12f' + 10f'' + 10f''')\, J + 5\, I}_{\sum_{\mathbf{c} \in \mathcal{D}} |\mathbf{c}| - 6fJ}\right) \tag{58}$$

$$\implies \underbrace{\bigcirc_{j=1}^{J} [\langle \otimes, x_j^{\mathtt{F}} \rangle, \langle x_j^{\mathtt{T}}, \otimes \rangle]}_{\mathbf{m}_1} \subset \mathbf{m}, \quad \underbrace{\bigcirc_{j=1}^{J} [\langle x_j^{\mathtt{F}}, \circledcirc \rangle, \langle \circledcirc, x_j^{\mathtt{T}} \rangle]}_{\mathbf{m}_3} \subset \mathbf{m}, \quad \underbrace{\bigcirc_{j=1}^{J} [\langle \circledcirc, x_j^{\mathtt{F}} \rangle, \langle x_j^{\mathtt{T}}, \circledcirc \rangle]}_{\mathbf{m}_5} \subset \mathbf{m}$$

Proof. We prove this statement by contradiction. Assume that one of the subwords above is not present in the tokenisers' merge sequence $\mathbf{m}$. In that case, the strings in $\mathcal{D}_1$ which contain this character string will not be compressed, and will thus still be represented with 2 symbols. There will thus be at most $6J - 1$ strings in $\mathcal{D}_1$ represented with a single symbol, and at least one represented with two symbols. The minimum length achievable would thus be:

$$\texttt{toklen}(\mathcal{D}, \mathbf{m}) = \underbrace{\sum_{\mathbf{c} \in \bigcup_{\_=1}^{f} \mathcal{D}_1} |\texttt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})|}_{\geq (6J-1)f + 2f} + \underbrace{\sum_{\mathbf{c} \in \mathcal{D} \setminus (\bigcup_{\_=1}^{f} \mathcal{D}_1)} |\texttt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})|}_{>0} \tag{59a}$$

$$> (6J + 1)f \qquad \text{By construction } f = (12f' + 10f'' + 10f''')\, J + 5I \tag{59b}$$

$$= (6f + 12f' + 10f'' + 10f''')\, J + 5I \tag{59c}$$

which contradicts the proof's statement. □

LemmaProofStep 2. (Step ②). *Any solution which compresses the text by at least $(6f + 6f')J$ symbols must only include either merges in $\mathbf{m}_1$, $\mathbf{m}_3$, $\mathbf{m}_5$, or in either $\mathbf{m}_j^{\circledcirc}$ or $\mathbf{m}_j^{\otimes}$, i.e.,:*

$$\left(\texttt{toklen}(\mathcal{D}, \mathcal{S}) \leq \underbrace{(6f + 6f')J + (10f'' + 10f''')\, J + 5\, I}_{\sum_{\mathbf{c} \in \mathcal{D}} |\mathbf{c}| - (6f + 6f')J}\right) \tag{60}$$

$$\implies \mathbf{m} \setminus (\mathbf{m}_1 \circ \mathbf{m}_3 \circ \mathbf{m}_5) \subseteq \underbrace{\left\{ \begin{matrix} \langle \circledcirc, x_j^{\mathtt{T}} \otimes \rangle, \langle \otimes x_j^{\mathtt{F}}, \circledcirc \rangle, \langle \circledcirc x_j^{\mathtt{T}}, \circledcirc \rangle, \langle \circledcirc, x_j^{\mathtt{F}} \circledcirc \rangle \\ \langle \circledcirc x_j^{\mathtt{T}}, \otimes \rangle, \langle \otimes, x_j^{\mathtt{F}} \circledcirc \rangle, \langle \circledcirc, x_j^{\mathtt{T}} \circledcirc \rangle, \langle \circledcirc x_j^{\mathtt{F}}, \circledcirc \rangle \end{matrix} \right\}_{j=1}^{J}}_{\bigcup_{j=1}^{J} (\mathbf{m}_j^{\circledcirc} \cup \mathbf{m}_j^{\otimes})}$$

Proof. We again prove this statement by contradiction. Assume that $\mathbf{m}$ has all merges $\mathbf{m}_1, \mathbf{m}_3, \mathbf{m}_5$, but one of its other merges is in neither of the sets $\mathbf{m}_j^{\circledcirc}$ and $\mathbf{m}_j^{\otimes}$. This means that at least one of the sets $\mathbf{m}_j^{\circledcirc} \cup \mathbf{m}_j^{\otimes}$ will not have at least two merges in the solution; this is because there are J such sets $(\mathbf{m}_j^{\circledcirc} \cup \mathbf{m}_j^{\otimes})$, which—coupled together with the $6J$ already selected merges in $\mathbf{m}_1, \mathbf{m}_3, \mathbf{m}_5$—would amount to the maximum of $8J$ merges. In that case, the strings (e.g., $\circledcirc x_j^{\mathtt{T}} \circledcirc$, $\circledcirc x_j^{\mathtt{F}} \circledcirc$, $\circledcirc x_j^{\mathtt{T}} \otimes$ and $\otimes x_j^{\mathtt{F}} \circledcirc$) in $\mathcal{D}_2$ containing the characters this absent merge represents will not be fully compressed to a single symbol, being represented with 2 symbols instead. Out of the $4J$ strings in $\mathcal{D}_2$ then, there will thus be at most $2J - 1$ represented with a single symbol, and at least $2J + 1$ represented with two symbols—resulting in a total of at least $2J - 1 + 2(2J + 1) = 6J + 1$ symbols. The minimum length achievable would thus be:

$$\texttt{toklen}(\mathcal{D}, \mathbf{m}) = \underbrace{\sum_{\mathbf{c} \in \bigcup_{\_=1}^{f} \mathcal{D}_1} |\texttt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})|}_{=6fJ} + \underbrace{\sum_{\mathbf{c} \in \bigcup_{\_=1}^{f'} \mathcal{D}_2} |\texttt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})|}_{\geq (6J+1)f'} + \underbrace{\sum_{\mathbf{c} \in \mathcal{D} \setminus (\mathcal{D}_1 \cup \mathcal{D}_2)} |\texttt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})|}_{>0} \tag{61a}$$

$$> 6fJ + (6J + 1)f' \qquad \text{By construction } f' = (10f'' + 10f''')\, J + 5I \tag{61b}$$

$$= (6f + 6f' + 10f'' + 10f''')\, J + 5\, I \tag{61c}$$

which contradicts the proof's statement. □

LemmaProofStep 3. (Step ③). *Any solution which compresses the text by at least* $(6f+6f'+6f'')J$ *symbols must include all merges in* $\mathbf{m}_1$, $\mathbf{m}_3$, $\mathbf{m}_5$, *and, for each* $j \in \{1,\dots,J\}$, *exactly one merge in set* $\mathbf{m}_j^{\circledcirc}$ *and one in set* $\mathbf{m}_j^{\otimes}$, *i.e.,:*

$$\left(\texttt{toklen}(\mathcal{D},\mathbf{m}) \le \underbrace{(6f+6f'+4f'')J + 10f'''\,J + 5\,I}_{\sum_{\mathbf{c}\in\mathcal{D}}|\mathbf{c}|-(6f+6f'+6f'')J}\right) \tag{62}$$

$$\implies \forall_{j\in\{1,\dots,J\}} \left|\mathbf{m} \cap \underbrace{\left\{\begin{matrix}\langle\circledcirc x_j^{\mathtt{T}},\circledcirc\rangle,\langle\circledcirc,x_j^{\mathtt{F}}\circledcirc\rangle \\ \langle\circledcirc,x_j^{\mathtt{T}}\circledcirc\rangle,\langle\circledcirc x_j^{\mathtt{F}},\circledcirc\rangle\end{matrix}\right\}}_{\mathbf{m}_j^{\circledcirc}}\right| = 1 \texttt{ and } \left|\mathbf{m} \cap \underbrace{\left\{\begin{matrix}\langle\circledcirc,x_j^{\mathtt{T}}\otimes\rangle,\langle\otimes x_j^{\mathtt{F}},\circledcirc\rangle \\ \langle\circledcirc x_j^{\mathtt{T}},\otimes\rangle,\langle\otimes,x_j^{\mathtt{F}}\circledcirc\rangle\end{matrix}\right\}}_{\mathbf{m}_j^{\otimes}}\right| = 1$$

Proof. We again prove this statement by contradiction. First, assume that $\mathbf{m}$ contains all the merges in $\mathbf{m}_1, \mathbf{m}_3, \mathbf{m}_5$; further, assume all its other merges are contained in sets $\mathbf{m}_j^{\circledcirc}$ and $\mathbf{m}_j^{\otimes}$. Note now that, if any merge in $\mathbf{m}_j^{\otimes}$ is in the selected merges $\mathbf{m}$, the string $\otimes x_j^{\mathtt{F}} \circledcirc x_j^{\mathtt{T}} \otimes$ in $\mathcal{D}_3$ will be compressed to 2 symbols (e.g., $\langle \otimes x_j^{\mathtt{F}}, \circledcirc x_j^{\mathtt{T}} \otimes\rangle$); if none of these merges is present, however, this string will only be compressed to 3 symbols (e.g., $\langle \otimes x_j^{\mathtt{F}}, \circledcirc, x_j^{\mathtt{T}} \otimes\rangle$). The same is true for strings $\circledcirc x_j^{\mathtt{T}} \circledcirc x_j^{\mathtt{F}} \circledcirc$ and merges in $\mathbf{m}_j^{\circledcirc}$. Now, assume the contradictory case: for a value of $j \in \{1,\dots,J\}$, $\mathbf{m}$ does not satisfy the condition above. As, by construction, our solution has $K = 8J$ merges, and because $|\mathbf{m}_1 \circ \mathbf{m}_3 \circ \mathbf{m}_5| = 6J$, we know that we have $2J$ merges in sets $\mathbf{m}_j^{\circledcirc}$ and $\mathbf{m}_j^{\otimes}$. As there are exactly $2J$ such sets, if the condition above does not hold, at least one of these sets must have no merge present in $\mathbf{m}$. In that case, the strings in $\mathcal{D}_3$ which contain the character string represented by these absent merges will be compressed to three symbols, while others will be compressed to two symbols. There will thus be at most $2J-1$ strings in $\mathcal{D}_3$ represented with two symbols, and at least one represented with three symbols. The minimum length achievable would thus be:

$$\texttt{toklen}(\mathcal{D},\mathbf{m}) = \underbrace{\sum_{\mathbf{c}\in \bigcup_{\_=1}^{f}\mathcal{D}_1 \cup \bigcup_{\_=1}^{f'}\mathcal{D}_2} |\texttt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})|}_{=(6f+6f')J} + \underbrace{\sum_{\mathbf{c}\in \bigcup_{\_=1}^{f''}\mathcal{D}_3} |\texttt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})|}_{\ge(2J-1)2f''+3f''} + \underbrace{\sum_{\mathbf{c}\in \bigcup_{\_=1}^{f'''}\mathcal{D}_4\cup\mathcal{D}_5} |\texttt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})|}_{>0} \tag{63a}$$

$$> (6f+6f')J + (4J+1)f'' \qquad \text{By construction } f'' = 10f'''\,J + 5I \tag{63b}$$

$$= (6f+6f'+4f''+10f''')\,J + 5\,I \tag{63c}$$

which contradicts the proof's statement. □

LemmaProofStep 4. (Step ④). *Any solution which compresses the text by at least* $(6f+6f'+6f''+6f''')J$ *symbols must include all merges in* $\mathbf{m}_1$, $\mathbf{m}_3$, $\mathbf{m}_5$, *and, for each* $j \in \{1,\dots,J\}$, *exactly one merge in set* $\mathbf{m}_j^{\circledcirc}$ *and one in set* $\mathbf{m}_j^{\otimes}$, *such that either both these merges are in* $\mathbf{m}_j^{\mathtt{T}}$ *or both are in* $\mathbf{m}_j^{\mathtt{F}}$, *i.e.,:*

$$\left(\texttt{toklen}(\mathcal{D},\mathbf{m}) \le \underbrace{(6f+6f'+4f''+4f''')J + 5\,I}_{\sum_{\mathbf{c}\in\mathcal{D}}|\mathbf{c}|-(6f+6f'+6f''+6f''')J}\right) \tag{64}$$

$$\implies \forall_{j\in\{1,\dots,J\}} \left|\mathbf{m} \cap \underbrace{\left\{\begin{matrix}\langle\circledcirc x_j^{\mathtt{T}},\circledcirc\rangle,\langle\circledcirc,x_j^{\mathtt{T}}\otimes\rangle, \\ \langle\circledcirc,x_j^{\mathtt{T}}\circledcirc\rangle,\langle\circledcirc x_j^{\mathtt{T}},\otimes\rangle,\end{matrix}\right\}}_{\mathbf{m}_j^{\mathtt{T}}}\right| = 2 \texttt{ or } \left|\mathbf{m} \cap \underbrace{\left\{\begin{matrix}\langle\circledcirc,x_j^{\mathtt{F}}\circledcirc\rangle,\langle\otimes x_j^{\mathtt{F}},\circledcirc\rangle, \\ \langle\circledcirc x_j^{\mathtt{F}},\circledcirc\rangle,\langle\otimes,x_j^{\mathtt{F}}\circledcirc\rangle,\end{matrix}\right\}}_{\mathbf{m}_j^{\mathtt{F}}}\right| = 2$$

Proof. First, note that the conditions of the step of our proof are stricter than previous ones, so we assume the conditions of steps ① to ③ hold—i.e., $\mathbf{m}$ contains all merges in $\mathbf{m}_1, \mathbf{m}_3, \mathbf{m}_5$; further, it has one and only one merge from each set $\mathbf{m}_j^{\circledcirc}$ and $\mathbf{m}_j^{\otimes}$. (Note that $\mathbf{m}_j^{\circledcirc} \cup \mathbf{m}_j^{\otimes} = \mathbf{m}_j^{\mathtt{T}} \cup \mathbf{m}_j^{\mathtt{F}}$, and that the just-mentioned condition implies $|\mathbf{m} \cap (\mathbf{m}_j^{\mathtt{T}} \cup \mathbf{m}_j^{\mathtt{F}})| = 2$.) We now again prove this statement by contradiction. Consider now the case:

$$\left|\mathbf{m} \cap \underbrace{\left\{\begin{matrix}\langle\circledcirc x_j^{\mathtt{T}},\circledcirc\rangle,\langle\circledcirc,x_j^{\mathtt{T}}\otimes\rangle, \\ \langle\circledcirc,x_j^{\mathtt{T}}\circledcirc\rangle,\langle\circledcirc x_j^{\mathtt{T}},\otimes\rangle,\end{matrix}\right\}}_{\mathbf{m}_j^{\mathtt{T}}}\right| = 2 \texttt{ or } \left|\mathbf{m} \cap \underbrace{\left\{\begin{matrix}\langle\circledcirc,x_j^{\mathtt{F}}\circledcirc\rangle,\langle\otimes x_j^{\mathtt{F}},\circledcirc\rangle, \\ \langle\circledcirc x_j^{\mathtt{F}},\circledcirc\rangle,\langle\otimes,x_j^{\mathtt{F}}\circledcirc\rangle,\end{matrix}\right\}}_{\mathbf{m}_j^{\mathtt{F}}}\right| = 2 \tag{65}$$

If this is true, then strings $\circledcirc x_j^{\mathtt{F}} \circledcirc x_j^{\mathtt{T}} \otimes$ and $\otimes x_j^{\mathtt{F}} \circledcirc x_j^{\mathtt{T}} \circledcirc$ in $\mathcal{D}_4$ will be compressed to 2 symbols each (e.g., to $\langle \circledcirc x_j^{\mathtt{F}}, \circledcirc x_j^{\mathtt{T}} \otimes \rangle$ and $\langle \otimes x_j^{\mathtt{F}}, \circledcirc x_j^{\mathtt{T}} \circledcirc \rangle$ or $\langle \circledcirc x_j^{\mathtt{F}} \circledcirc, x_j^{\mathtt{T}} \otimes \rangle$ and $\langle \otimes x_j^{\mathtt{F}} \circledcirc, x_j^{\mathtt{T}} \circledcirc \rangle$); if this condition is false, however, one of these strings will only be compressed to 3 symbols (e.g., to $\langle \circledcirc x_j^{\mathtt{F}}, \circledcirc x_j^{\mathtt{T}} \otimes \rangle$ and $\langle \otimes x_j^{\mathtt{F}}, \circledcirc, x_j^{\mathtt{T}} \circledcirc \rangle$). Now, assume the contradictory case: for a value of $j \in \{1, \ldots, J\}$, $\mathbf{m}$ does not satisfy the condition above. In that case, the strings in $\mathcal{D}_4$ for which the condition does not hold will be compressed to $3 + 2$ symbols, while others will be compressed to $2 + 2$ symbols. There will thus be at most $2J - 1$ strings in $\mathcal{D}_4$ represented with two symbols, and at least one represented with three symbols. The minimum length achievable would thus be:

$$\begin{aligned}
\mathtt{toklen}(\mathcal{D}, \mathbf{m}) &= \underbrace{\sum_{\mathbf{c} \in \bigcup_{\_=1}^{f} \mathcal{D}_1 \cup \bigcup_{\_=1}^{f'} \mathcal{D}_2 \cup \bigcup_{\_=1}^{f''} \mathcal{D}_3} |\mathtt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})|}_{=(6f+6f'+4f'')J} + \underbrace{\sum_{\mathbf{c} \in \bigcup_{\_=1}^{f'''} \mathcal{D}_4} |\mathtt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})|}_{\geq (2J-1)2f'''+3f'''} + \underbrace{\sum_{\mathbf{c} \in \mathcal{D}_5} |\mathtt{tok}_{\uparrow}[\mathbf{m}](\mathbf{c})|}_{>0} && \text{(66a)} \\
&> (6f + 6f' + 4f'')J + (4J + 1)f''' \qquad \text{By construction } f''' = 5I && \text{(66b)} \\
&= (6f + 6f' + 4f'' + 4f''')\, J + 5\, I && \text{(66c)}
\end{aligned}$$

which contradicts the proof's statement. □

LemmaProofStep 5. (Step ⑤). *Any instance of the tokenisation problem with a solution which compresses the text by at least $(6f + 6f' + 6f'' + 6f'')J + 2I + \gamma$ symbols must be produced by a* `max-2-SAT` *problem with at least γ satisfied clauses, i.e.,:*

$$\left(\mathtt{toklen}(\mathcal{D}, \mathcal{S}) \leq \underbrace{(6f + 6f' + 4f'' + 4f''')J + 3I - \gamma}_{\sum_{\mathbf{c} \in \mathcal{D}} |\mathbf{c}| - (6f+6f'+6f''+6f''')J - 2I - \gamma} \right) \implies \mathrm{M2S}(\mathcal{X}, \mathcal{C}, \gamma)$$

Proof. Finally, we now know any solution with this compression must have—for any variable X_j—either two merges in $\mathbf{m}_j^{\mathtt{T}}$ or in $\mathbf{m}_j^{\mathtt{F}}$ (and never both). We can thus create a bijection $\mathrm{Conv}_{\mathbf{m} \to \chi}$ between the set of possible merge sequences respecting this condition, and the set of `T`/`F` assignments to SAT variables χ:

$$\mathrm{Conv}_{\mathbf{m} \to \chi}(\mathbf{m}) = \left\{ \begin{array}{ll} \mathtt{T} & \mathtt{if}\, |\mathbf{m} \cap \mathbf{m}_j^{\mathtt{T}}| = 2 \\ \mathtt{F} & \mathtt{if}\, |\mathbf{m} \cap \mathbf{m}_j^{\mathtt{F}}| = 2 \end{array} \right\}_{j=1}^{J} \tag{67}$$

Further, note that merge sequences of this form (as shown in Eq. (41)) lead to exactly $(6f + 6f' + 6f'' + 6f''')J$ symbols being compressed in datasets $\mathcal{D}_1$ to $\mathcal{D}_4$. To achieve the target compression, a solution must thus compress $\mathcal{D}_5$ by at least $2I + \gamma$ symbols. Now note that for any string in $\mathcal{D}_5$, e.g., $\circledcirc x_j^{\mathtt{T}} \circledcirc x_{j'}^{\mathtt{F}} \circledcirc$, we have three compression options: $\circledcirc x_j^{\mathtt{T}} \circledcirc$ and $x_{j'}^{\mathtt{F}} \circledcirc$ will be compressed, saving 3 symbols; $\circledcirc x_j^{\mathtt{T}}$ and $\circledcirc x_{j'}^{\mathtt{F}} \circledcirc$ will be compressed, also saving 3 symbols; or only $\circledcirc x_j^{\mathtt{T}}$ and $x_{j'}^{\mathtt{F}} \circledcirc$ will be compressed saving only 2 symbols. More specifically, $\circledcirc x_j^{\mathtt{T}} \circledcirc$ will be compressed to a single symbol if merge $\langle \circledcirc, x_j^{\mathtt{T}} \circledcirc \rangle$ exists; similarly, $\circledcirc x_{j'}^{\mathtt{F}} \circledcirc$ will be compressed to a single symbol if merge $\langle \circledcirc x_{j'}^{\mathtt{F}}, \circledcirc \rangle$ exists. They cannot both be compressed, however, as there is only one symbol $\circledcirc$ between the literals. We thus get a reduction of 3 symbols for each of these strings if at least one of its literals has an associated merge in $\mathbf{m}$. Note thus that whenever a string $\circledcirc x_j^{\mathtt{T}} \circledcirc x_{j'}^{\mathtt{F}} \circledcirc$ is compressed by 3 symbols using merges $\mathbf{m}$, the `max-2-SAT` disjunction $X_j \vee \neg X_{j'}$ will also be satisfied by assignment $\chi = \mathrm{Conv}_{\mathbf{m} \to \chi}(\mathbf{m})$; similarly, whenever this string is only compressed by two symbols, the `max-2-SAT` disjunction will not be satisfied. A similar logic applies to all potential strings in $\mathcal{D}_5$: $\circledcirc x_j^{\mathtt{T}} \circledcirc x_{j'}^{\mathtt{F}} \circledcirc$, $\circledcirc x_{j'}^{\mathtt{T}} \circledcirc x_j^{\mathtt{F}} \circledcirc$, $\otimes x_j^{\mathtt{F}} \circledcirc x_{j'}^{\mathtt{F}} \circledcirc$, and $\circledcirc x_j^{\mathtt{T}} \circledcirc x_{j'}^{\mathtt{T}} \otimes$. As our condition assumes a compression of at least $2I + \gamma$ symbols, we know that we have at least γ strings for which a literal has an associated merge. Now, let $\mathbf{m}^\star \in \mathcal{M}^*$ be a valid solution to $\mathrm{Tok}_{\uparrow}(\mathrm{R2}(\mathcal{X}, \mathcal{C}, \gamma))$ and

$\chi^\star = \text{Conv}_{\mathbf{m}\to\chi}(\mathbf{m}^\star)$ the equivalent `max-2-SAT` assignment. We can thus write:

$$2I + \gamma \leq \sum_{\mathbf{c}\in\mathcal{D}_5} |\mathbf{c}| - |\texttt{tok}_{\uparrow}[\mathbf{m}^\star](\mathbf{c})| \tag{68a}$$

$$= 2I + \sum_{\mathbf{c}\,\in\,\mathcal{D}_5} \mathbb{1}\left\{\begin{array}{c} \left(x_j^{\texttt{T}} \in \mathbf{c}\right) \texttt{ and } \left(|\mathbf{m}_j^{\texttt{T}} \cap \mathbf{m}^\star| = 2\right) \\ \texttt{or} \\ \left(x_j^{\texttt{F}} \in \mathbf{c}\right) \texttt{ and } \left(|\mathbf{m}_j^{\texttt{F}} \cap \mathbf{m}^\star| = 2\right) \\ \texttt{or} \\ \left(x_{j'}^{\texttt{T}} \in \mathbf{c}\right) \texttt{ and } \left(|\mathbf{m}_{j'}^{\texttt{T}} \cap \mathbf{m}^\star| = 2\right) \\ \texttt{or} \\ \left(x_{j'}^{\texttt{F}} \in \mathbf{c}\right) \texttt{ and } \left(|\mathbf{m}_{j'}^{\texttt{F}} \cap \mathbf{m}^\star| = 2\right) \end{array}\right\} \tag{68b}$$

$$= 2I + \sum_{i=1}^{I} \mathbb{1}_{\chi^\star}\{L_i^1 \vee L_i^2\} \tag{68c}$$

$$\implies \text{M2S}(\mathcal{X}, \mathcal{C}, \gamma) \tag{68d}$$

where $\mathbb{1}_{\chi^\star}\{L_i^1 \vee L_i^2\}$ evaluates $L_i^1 \vee L_i^2$ using assignments $\chi^\star$. We thus know that, if a satisfying tokenisation solution exists, then the associated `max-2-SAT` problem will also be satisfiable. This concludes the proof. $\square$