

More Than Words: Compositional Tokenization for Efficient Language Models

Yuval Reif Guy Kaplan Roy Schwartz

The Hebrew University of Jerusalem

{yuval.reif, guy.kaplan3, roy.schwartz1}@mail.huji.ac.il

Abstract

Language models process and generate text sequentially in *token* units, and the tokenizer determines how much text each inference step covers. Under standard tokenization, a short English phrase such as “On the table.” is usually produced as four separate predictions for the preposition (*On*), article (*the*), noun (*table*), and punctuation (*.*), where each consumes a sequence position and adds inference cost. We introduce COBPE, a compositional tokenization approach that represents such phrases as a lexical base token (*table*) attached with a small set of reusable surface modifiers, composed in embedding space at input and predicted jointly at output. In controlled pretraining from scratch at 780M and 1.3B scales, COBPE shortens sequences by 30% and improves average downstream performance by 1.2 points relative to standard BPE under matched training compute. Our results suggest that part of what is now expressed through token sequences can instead be modeled through structured representations, opening a broad design space for more token-efficient and capable language models.¹

1 Introduction

Large language models (LLMs) do not operate directly on raw text, but instead consume *token* sequences, most commonly produced by Byte-Pair Encoding (BPE; [Sennrich et al., 2016](#)). To build the model’s token vocabulary, such methods first split text into whitespace- and punctuation-bounded chunks in *pre-tokenization*, and then allocate tokens to frequent byte sequences within chunks ([OpenAI, 2024](#)).

While effective, this process results in units that inherit pre-tokenization boundaries and follow surface string structure, creating two related inefficiencies. First, very common sequences such as “on the” cannot become a single token ([Liu et al., 2025](#)). Instead, the model must process them using multiple sequential steps, with items that carry different amounts of content such as “the”, “table”, and “,” all requiring the same amount of compute. Second, the same interface treats near-duplicate forms such as *table*, capitalized *Table*, and space-prefixed *_table* as separate vocabulary entries. As a result, related tokens are represented using inconsistent symbols and require the model to learn their relation from data ([Schäfer et al., 2024](#); [Kaplan et al., 2025](#)), while also consuming token slots in the size-limited vocabulary that could otherwise support other languages or domains ([Reif et al., 2025](#)). These choices can materially affect model efficiency, as training and inference costs depend directly on the number of tokens used to represent text ([Ahia et al., 2023](#); [Ali et al., 2024](#)). Overall, this suggests there is still substantial room to improve compression beyond standard sub-word tokenization.

In this paper, we introduce a **Compositional BPE** tokenization approach (COBPE). COBPE segments text into *compound tokens*, each composed of a *base token* and a small set of *surface modifiers*. The base token serves as the lexical anchor, while the modifiers encode leading space, capitalization, prepositions, determiners, and surrounding punctuation (see Figure 1, top). On the input side, the model composes the unit in embedding space by adding the

¹Code and model weights are available at <https://co-bpe.github.io/>.

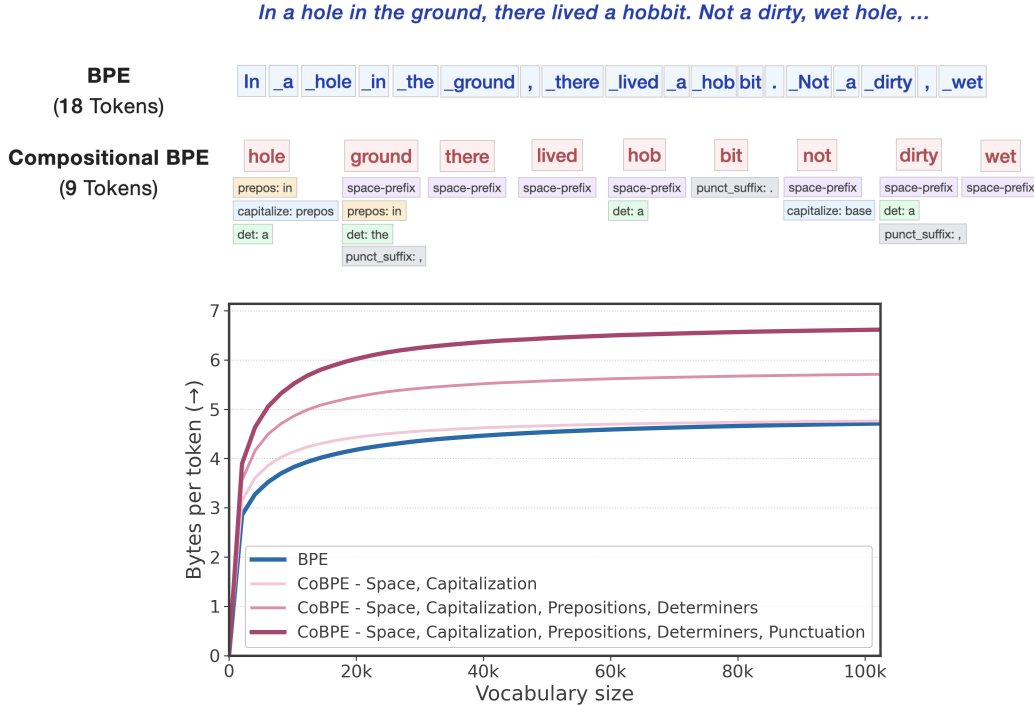


Figure 1: **Compositional tokenization shortens sequences compared to BPE by factoring recurring surface markers around a lexical base.** **Top:** standard BPE represents the excerpt as a sequence of 18 tokens, while Compositional BPE (CoBPE) represents the same text using only 9 *compound tokens* built from a base token together with attached modifiers for leading space, capitalization, determiners, prepositions, and surrounding punctuation. **Bottom:** Encoding efficiency (bytes-per-token) of English text for BPE and CoBPE as a function of vocabulary size. Adding modifier families progressively improves compression over BPE, and the full CoBPE configuration covers roughly $1.4\times$ more bytes per token across vocabulary sizes.

active modifier embeddings to the base embedding; on the output side, it predicts the base token and modifier labels as a structured output, with modifier predictions conditioned on the base token. Thus, phrases like “On the table?” can be generated as a single compound token centered on table, rather than four separate token predictions.

We first quantify how much CoBPE shortens sequences relative to standard BPE by folding frequent markers such as determiners, prepositions, and punctuation into the same structured unit as a base token (Figure 1, bottom). We then describe how a standard decoder-only transformer can be adapted to consume and generate compound tokens through compositional input embeddings and factorized output prediction.

Next, we study CoBPE through controlled English pretraining from scratch on the Climb-Mix corpus (Diao et al., 2025) at 780M and 1.3B scales. Across the main experiments, we hold the model family, architecture, training recipe, vocabulary budget, and evaluation suite fixed and vary only the tokenization scheme.

In experiments, CoBPE improves average downstream performance over standard BPE under matched training-compute budgets while also shortening sequences substantially, allowing more text to be processed under the same budget. At both model scales, it yields stronger downstream performance on the same 30-task evaluation suite, improving the average score by 1.2 points while reducing sequence length by 30%. To test whether these gains translate into end-to-end efficiency, we evaluate CoBPE on the NANOCHAT speedrun benchmark (Karpathy, 2025), which measures how quickly a model reaches a fixed capability

target. In that setting, COBPE reduces training time by 18.6%, showing that its compression gains translate into faster end-to-end training.

Taken together, our results reveal a broader design space for language modeling, in which some of the structure now carried by sequence length can instead be modeled directly within the token representation itself. In COBPE, this takes the form of a shared lexical base combined with a small set of reusable modifiers, yielding a tokenizer that is both more compact and more effective under standard language-model training. More broadly, it points to structured token representations as a promising alternative to standard tokenization.

2 Tokenization beyond flat sequences

In modern LLMs, tokenization determines how much text each model step represents. Because every sequence position incurs computation, this choice directly affects training and inference costs (Ahia et al., 2023; Petrov et al., 2023). The standard approach is BPE tokenization with word-boundary pre-tokenization (Sennrich et al., 2016; Dagan et al., 2024), which greedily builds a token vocabulary of a target size before pretraining. Given a large text sample, the text is first split into word-bounded chunks in pre-tokenization, and the vocabulary is then built by repeatedly merging the most frequent adjacent character or sub-word sequences within those chunks until the target size is reached, allocating tokens to the results of each merge.

At sufficiently large vocabulary sizes, BPE is highly effective for compression: in English, modern tokenizers use roughly 1.2 tokens per word on average (Ali et al., 2024), with most words represented by a single token, each receiving its own inference step. However, this design allocates similar compute to process *function words*—frequent words expressing grammatical relations (*the, of, on*)—as it does to *content words*, which carry most of the lexical meaning (such as nouns, main verbs and adjectives). Punctuation is treated similarly: with pre-tokenization, punctuation marks are typically separate tokens rather than part of surrounding words. Importantly, this suggests that even after BPE’s strong compression, considerable compute is spent on processing standalone grammatical tokens.

Recent work relaxes word-boundary constraints in pre-tokenization and allows multi-word tokens to form (e.g., “*in the ground.*”), substantially improving compression and downstream efficiency (Liu et al., 2025; Schmidt et al., 2025). This compression, however, is phrase-specific: for example, instead of representing “*the*” as a reusable unit across contexts, the vocabulary must allocate separate entries to combinations such as “*the ground*”, “*the earth*”, and similar phrases. Pushing this compression strategy further requires adding increasingly specific entries such as “*in the ground?*” or “*On the ground!*”, making it impractical to capture the richness of language even under large vocabulary sizes.

These observations suggest that attaching such markers to neighboring words could substantially improve compression. To measure this opportunity, we quantify the fraction of token positions occupied by grammatical markers in English FineWeb documents (Penedo et al., 2024). We analyze text tokenized with GPT-4’s tokenizer (OpenAI, 2024), which several modern LLMs adopt as a starting point for their own tokenizers (Yang et al., 2024; Grattafiori et al., 2024). We measure the share of positions taken by standalone determiners (e.g., *the, a, this*), prepositions (*by, in, etc.*), and punctuation. In Figure 2, we show that these categories account for 29.8% of all token positions,² despite carrying relatively little lexical content (Jiang et al., 2023; Bitton et al., 2021). This indicates that much of the remaining compression potential for natural language comes not from encoding rarer words, but from explicitly modeling frequent grammatical and orthographic structure surrounding words.

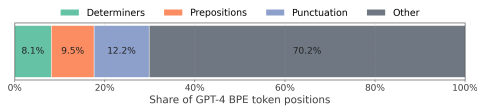


Figure 2: **A substantial share of token positions under a modern BPE tokenizer are grammatical markers.** Breakdown of token positions occupied by determiners, prepositions, and punctuation in FineWeb text tokenized with GPT-4.

²See Figure 4 in Appendix A for a detailed view of the most frequent markers within each category.

3 COBPE: compositional BPE tokenization

The potential to compress frequent grammatical markers (§2) motivates a simple reformulation of standard BPE: rather than dedicating separate prediction steps or vocabulary entries to frequent grammatical material or surface variation, we can represent these as *modifiers* attached to lexical *base tokens* and process each composition in a single inference step.

We call this approach Compositional BPE tokenization (COBPE). We next define the tokenizer and its modifier inventory (§3.1), show how it improves compression relative to BPE (§3.2), and describe the adaptation needed for a language model to consume and generate compound tokens (§3.3).

3.1 Definition

Standard BPE segments a phrase like “*in the ground*,” into four tokens: preposition, determiner, noun, and punctuation. Under COBPE, such spans are represented as a single structured unit we call a *compound token*. Each compound token is built around a lexical *base token*, along with a small set of attached *modifiers* encoding frequent markers such as determiners, prepositions, and punctuation. Thus, the span above is represented as the base token `ground` along with the modifiers `preposition=in`, `determiner=the`, and `punct-suffix=,`.

BPE also treats common surface variants as distinct token strings. For example, `ground`, `Ground`, and `_ground` (with a leading space, as in mid-sentence occurrences) are separate tokens despite sharing the same lexical core. COBPE factors out this variation by keeping the base token fixed and representing leading space and capitalization as dedicated modifiers.

Table 1 shows the COBPE decomposition for several related spans. We focus on English and define eight modifier groups: four categorical groups for determiners, prepositions, prefix punctuation, and suffix punctuation; three binary indicators for whether the base token, determiner, and preposition are capitalized; and one binary indicator for the presence of a leading space before the span. We focus on markers that can be identified and reconstructed using local surface rules, with exact values and exclusions listed in Appendix B.³

Formally, COBPE maps a text t into a sequence of compound tokens $c_1, c_2, \dots, c_N$, where

$$c_i = \left(b_i, m_i^{(1)}, \dots, m_i^{(G)} \right)$$

Here, b_i is a lexical base token, and each $m_i^{(g)}$ is the value assigned to one modifier group. Together, the base token and modifier values specify the surface form realized at position i . If a word spans multiple tokens, COBPE distributes modifiers across it deterministically (see [hob, bi t] in Figure 1): leading-space, capitalization, determiner, preposition, and prefix-punctuation markers are attached to the first token in the span, while suffix punctuation is attached to the last token.

Having defined the representation, we next describe how the tokenizer constructs it. To construct a COBPE tokenizer, we first build a BPE tokenizer with a pre-tokenizer that separates punctuation and leading spaces from word chunks. To avoid allocating separate vocabulary entries to capitalized words that COBPE later represents through lowercase bases, we train this tokenizer on text in which standard titlecased words are lowercased.⁴ Function words and punctuation markers are still included in this underlying BPE vocabulary; they may later be attached as modifiers when composition is possible, but they also remain available as ordinary tokens when no valid attachment applies.

At inference time, COBPE processes text in two stages. The text is first tokenized with the underlying BPE tokenizer, which produces an ordinary BPE token stream containing

³Supporting other types of modifiers (e.g., morphological modifiers, Reif et al., 2025) requires external resources and is thus deferred to future work.

⁴For simplicity, we restrict capitalization modifiers to standard titlecase. ALL CAPS and camelCase are left unchanged and handled by the underlying BPE tokenizer, and broader capitalization handling is deferred to future work.

Text	BPE encoding	# tokens	COBPE encoding (1 compound token; base = ground)
ground	ground	1	—
the ground	the _ground	2	determiner=the
The ground	_The _ground	2	leading-space, determiner=the, capitalize-determiner
The Ground	_The _Ground	2	leading-space, capitalize, determiner=the, capitalize-determiner
In the ground	In _the _ground	3	preposition=in, capitalize-preposition, determiner=the
in the ground,	in _the _ground ,	4	preposition=in, determiner=the, punct-suffix=,
(in the ground)	(in the ground)	5	punct-prefix=(, preposition=in, determiner=the, punct-suffix=)

Table 1: Examples of COBPE decompositions. A leading underscore in the BPE column denotes a space-prefixed token. Standard BPE assigns separate token identities to surface variants such as “*ground*” and “*_Ground*”. It also represents the examples above using 1–5 sequential tokens. COBPE instead keeps the lexical base token identity fixed and expresses variation through attached modifiers within a single structured *compound token*.

lexical items, determiners, prepositions, and punctuation as separate tokens. A deterministic left-to-right pass then identifies eligible base tokens.⁵ For each base, we attach neighboring determiner, preposition, and punctuation markers from a closed inventory of modifiers, along with capitalization and leading-space cues.⁶ Tokens that cannot be validly attached remain as ordinary positions in the final sequence. The resulting sequence is the COBPE representation used throughout training and evaluation. See Appendix B for tokenizer construction details, modifier inventories and usage statistics, and pseudocode for the inference-time composition pass.

As we show next, COBPE allows the tokenizer to represent text in fewer sequence positions while preserving the surface information needed to reconstruct the original text.

3.2 Compression relative to BPE

COBPE improves compression through two complementary mechanisms. First, function-word and punctuation modifiers shorten sequences by absorbing frequent grammatical material into a single prediction. Second, modifiers for leading space and capitalization let related surface forms share a single lexical token (Reif et al., 2025), freeing vocabulary capacity from near-duplicate tokens and allowing additional unique tokens to be allocated.

We measure compression with *bytes per token* (BPT), the average number of UTF-8 bytes covered by each token on a large corpus; higher BPT means that the same text is represented using fewer tokens. We construct BPE and COBPE tokenizers on ClimbMix (Diao et al., 2025) and evaluate BPT on FineWeb (Penedo et al., 2024). For each tokenizer, we plot the BPT as a function of vocabulary size. To isolate the contribution of different modifier types, we also build COBPE variants incrementally: first adding leading-space and capitalization modifiers, then determiners and prepositions, and finally punctuation.

We report the results in Figure 1 (bottom). Sharing entries across capitalization and leading-space variants already improves BPT at smaller vocabulary sizes, indicating that BPE spends much of its vocabulary on near-duplicate forms. For example, in a 32k BPE vocabulary, 15% of entries correspond to capitalized variants and 12% to space-prefixed forms; under COBPE, these are represented through modifiers, allowing new, unique tokens to be allocated at the same vocabulary budget. Adding determiners, prepositions, and punctuation leads to large, consistent improvements across all vocabulary sizes, as these markers occur frequently enough to collapse many multi-token spans into a single compound token. Overall, COBPE improves BPT by roughly 40% across vocabulary sizes. This suggests that much of the remaining compression potential beyond standard BPE lies in modeling local structure around words, not only in better segmenting the words themselves.

⁵An eligible base token is any non-punctuation/numeric token that does not occur mid-word.

⁶This attachment rule is purely local and does not require syntactic parsing. For example, in “by the large house”, the markers “by the” attach to “large” rather than the syntactic head “house”. Importantly, no information is discarded: every component remains represented in the sequence, allowing the model to learn the relevant relationships from context, as in standard tokenization. We defer syntax-aware attachment to future work.

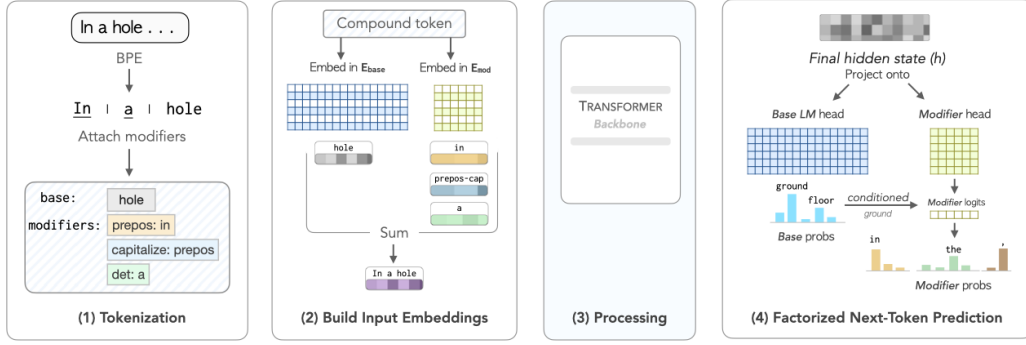


Figure 3: **Adapting language models to COBPE.** (1) A span tokenized under COBPE is represented as a structured *compound token* with a base token and attached modifiers. (2) On the input side, the model embeds this unit compositionally by summing the base embedding and the embeddings of its active modifiers. (3) The resulting sequence is processed by an otherwise standard transformer backbone. (4) On the output side, next-token prediction is factorized: the model first predicts the next base token, and then predicts the modifier values conditioned on that base.

3.3 Adapting language models to COBPE

Given a piece of text tokenized by COBPE, a standard decoder-only transformer can be adapted to operate over compound tokens by modifying the input and output parameterizations. The transformer still processes one sequence position at a time, but each position now corresponds to a structured unit rather than a single surface token.

On the input side, each compound token is represented by composing the embedding of its lexical base with embeddings for its active modifiers. Let $E_{\text{base}} \in \mathbb{R}^{|V| \times d}$ be the embedding matrix over the base-token vocabulary, and for each modifier group g , let $E_g \in \mathbb{R}^{|M_g| \times d}$ be the corresponding modifier embedding matrix. The input embedding of compound token position i is

$$\mathbf{e}(c_i) = E_{\text{base}}[b_i] + \sum_{g=1}^G E_g[m_i^{(g)}]$$

In other words, each input position is the sum of a base-token embedding and a small set of modifier embeddings. Since the modifier vocabularies are small, this adds few new parameters relative to the lexical vocabulary.

On the output side, the model predicts the next compound token in two stages: it first predicts the base token, and then predicts the modifier values conditioned on that base. Let $\mathbf{h}_i$ denote the final normalized hidden state at position i , and $U_{\text{base}} \in \mathbb{R}^{|V| \times d}$ the base-vocabulary unembedding matrix. The model first projects $\mathbf{h}_i$, yielding base-token logits:

$$\ell_i^{\text{base}} = U_{\text{base}} \mathbf{h}_i \quad p(b_i | c_{<i}) = \text{softmax}(\ell_i^{\text{base}})$$

Let $\mathbf{u}_{b_i}$ denote the normalized unembedding vector of the selected base token. The modifier prediction head combines contextual and lexical representations as

$$\ell_i^{\text{mod}} = W_h \mathbf{h}_i + \alpha_i W_b \mathbf{u}_{b_i}, \quad \alpha_i = \sigma(\mathbf{w}_\alpha^\top \mathbf{h}_i)$$

Here W_h and W_b project the hidden state and selected base-token representation into a shared modifier-logit space. The gate $\mathbf{w}_\alpha$ controls how strongly the selected base influences the modifier prediction. A refinement W_r then lets the modifier logits influence one another:

$$\tilde{\ell}_i^{\text{mod}} = \ell_i^{\text{mod}} + W_r \text{SiLU}(\ell_i^{\text{mod}})$$

Finally, the refined logits are partitioned into modifier groups, and a separate softmax is applied within each group, yielding the factorization⁷

$$p(c_i | c_{<i}) = p(b_i | c_{<i}) \prod_{g \in G} p(m_i^{(g)} | b_i, c_{<i})$$

where G denotes the modifier groups.⁸

During training, under teacher forcing, the modifier logits are conditioned on the gold base token. The loss therefore decomposes into a base-prediction term and a sum of modifier-prediction terms:

$$\mathcal{L} = - \sum_{i=1}^N \left[\log p(b_i | c_{<i}) + \sum_{g \in G} \log p(m_i^{(g)} | b_i, c_{<i}) \right]$$

where inactive groups take their *null* label. At inference time, the model first decodes a base token from $p(b_i | c_{<i})$ and then predicts the modifier values conditioned on that choice. The predicted compound token is then deterministically rendered as text directly from the structured base-plus-modifier prediction, so generation remains standard at the level of text even though the model internally operates over a shorter sequence of structured units.

Evaluation. Because COBPE changes both the prediction unit and the factorization of text probability, its token-level likelihoods are not directly comparable to those of a standard autoregressive language model. A single COBPE prediction may jointly determine a base token along with a determiner, preposition, capitalization, and punctuation, whereas a standard BPE model would treat these as separate (and often more easily predictable) tokens. Moreover, each COBPE position scores every modifier group, including inactive labels for unused modifiers. Byte-normalized likelihood therefore remains well defined, but measures this full structured prediction problem rather than sequence compression or compute efficiency alone. We therefore evaluate COBPE primarily through downstream performance, including both likelihood-based benchmarks and open-ended generation tasks, providing a natural basis for comparison across different language modeling setups.

4 Experiments

We test whether replacing BPE with COBPE improves downstream performance under controlled English pretraining from scratch (§4.2), and whether its improved compression translates into better training efficiency (§4.3). The main English comparison holds the architecture, training recipe, token budget, and evaluation suite fixed while varying tokenization. We then analyze inference speed (§4.4) and COBPE compression beyond English (§4.5).

4.1 Setup

We use ClimbMix to construct tokenizers and train models across all settings. Our main experiments use a standard decoder-only transformer architecture with a vocabulary size of 32k and untied input and output embeddings. We evaluate two model scales: a 24-layer model with 780M parameters and a 28-layer model with 1.3B parameters. Following Hoffmann et al. (2022), we train each model on 20 tokens per parameter, yielding approximately 15.6B and 26B tokens for the 24- and 28-layer models, respectively.⁹ At the 24-layer scale, we also include SuperBPE (Liu et al., 2025) as an additional baseline. Its transition point t determines when tokenizer construction stops enforcing word boundaries and begins

⁷This factorization assumes conditional independence among modifier groups given the hidden state and base token. Jointly predicting common modifier bundles or autoregressively predicting modifier groups are natural extensions, which we defer to future work.

⁸The modifier parameter overhead is small: the ~ 100 English modifiers require 0.5M parameters for embeddings and prediction, under 0.1% of the corresponding BPE model’s parameters.

⁹Notably, this comparison approximately matches training compute between BPE and COBPE models, but not raw-text exposure: COBPE encodes more text within the same number of tokens.

Category	Task	24-layer (780M)			28-layer (1.3B)	
		BPE	SuperBPE ($t = 27k$)	CoBPE	BPE	CoBPE
Knowledge	ARC-Easy (MC)	68.3	69.7	74.1	73.8	76.1
	ARC-Challenge (MC)	40.3	41.8	42.7	43.3	47.4
	Jeopardy (EM)	5.9	5.7	12.0	11.8	18.0
	MMLU (MC)	25.2	25.3	25.2	25.0	25.5
	OpenBookQA (MC)	39.8	42.6	42.2	42.8	39.2
	TriviaQA (EM)	9.1	9.2	14.6	14.3	18.7
	WikidataQA (EM)	39.6	36.8	49.8	47.5	51.5
Reasoning \ Symbolic	Arithmetic (EM)	1.7	1.9	1.8	5.9	2.7
	LSAT-AR (MC)	32.6	26.1	29.1	27.0	33.0
	Operators (EM)	15.7	17.1	19.5	22.4	21.9
Reading	BoolQ (MC)	56.0	62.4	61.9	63.5	67.6
	CoQA (EM)	43.0	39.4	43.3	47.3	50.3
	DROP (EM)	17.5	19.9	20.7	18.0	22.9
	SQuAD (EM)	49.9	46.5	50.9	55.9	56.0
Commonsense	CommonsenseQA (MC)	25.2	21.9	23.0	27.2	29.4
	COPA (MC)	63.0	62.0	70.0	71.0	75.0
	PIQA (MC)	74.6	73.8	74.5	76.7	76.7
	Winograd (MC)	67.0	68.9	66.3	71.8	72.9
	Winogrande (MC)	57.8	55.4	55.8	59.4	60.4
Language	HellaSwag (MC)	56.7	56.4	57.5	62.9	63.7
	LAMBADA (EM)	34.4	39.0	29.7	42.7	35.7
	Language ID (EM)	24.4	24.3	24.9	24.3	25.2
String / formal	CS Algorithms (EM)	39.4	39.4	40.7	39.5	45.8
	CUTE (EM)	1.7	2.1	2.7	3.4	3.7
	Dyck Languages (EM)	19.6	19.3	11.6	17.3	14.3
Aggregate	30-task average	30.4	30.4	31.6	33.3	34.6
	Relative sequence length	1.00	0.85	0.70	1.00	0.70

Table 2: **Main downstream results under matched training-compute budgets.** Models are trained for 15.6B tokens at the 24-layer, 780M scale and 26B tokens at the 28-layer, 1.3B scale. The primary comparison is BPE versus CoBPE at each scale. At the 24-layer scale, we also include SuperBPE as a secondary baseline, using the best transition point from our sweep for a 32k vocabulary. Best results per task and model size are in **bold**.

learning multiword tokens; we use $t = 27k$.¹⁰ We run each experiment on 8 L40S GPUs. Evaluation uses SuperBPE’s 30-task suite, spanning knowledge, reasoning, coding, reading, commonsense, language understanding, and string manipulation. See Appendix C for details on training, and Appendix D for data and evaluation details.

We further experiment with the NANOCHAT “GPT-2 speedrun” benchmark (Karpathy, 2025), which measures the time required to train a model under the NANOCHAT harness to GPT-2-level capability (Radford et al., 2019), according to the DCLM CORE downstream evaluation suite (Li et al., 2024). We use the leading NANOCHAT submission at the time of writing—a 24-layer model with a custom architecture and optimization scheme (see Appendix E)—and evaluate BPE and CoBPE under the same recipe, varying only tokenization, the corresponding input and output parameterization, and the training horizon (§3.3).

4.2 CoBPE is more compact and more accurate than BPE

Table 2 reports the main matched training-compute comparison.¹¹ At both the 24-layer (780M) and 28-layer (1.3B) scales, CoBPE outperforms standard BPE on the 30-task downstream suite while also reducing sequence length substantially. At the 24-layer scale, CoBPE improves the 30-task average from 30.4 to 31.6 (+1.2), while SuperBPE matches the BPE baseline at 30.4. At the 28-layer scale, CoBPE improves the average from 33.3 to 34.6 (+1.3). Importantly, CoBPE is also far more efficient than BPE. Across both scales, CoBPE yields a 30% reduction in sequence length, compared with 15% for SuperBPE.

¹⁰We swept $t \in \{20k, 25k, 27k\}$ and selected by the highest 30-task average; the original paper used $t = 180k$ with a 200k vocabulary, and pretrained much larger 8B models.

¹¹For readability, we omit 5 tasks on which all models score near zero, while still including them in the 30-task average; their results are given in Appendix A.2.

Model	CORE $\uparrow$	Tokens to target $\downarrow$	Hours to target $\downarrow$	Speedup $\uparrow$
BPE	25.91	5.84B	7.92h	1.00 $\times$
CoBPE	26.00	4.74B	6.45h	1.23 $\times$

Table 3: **NANOCHAT speedrun results on 8 L40S GPUs.** We report CORE score at the stopping point, along with the training tokens and wall-clock hours used in each run. Times are comparable within this table, but not to the public NANOCHAT H100 leaderboard.

Tokenization-sensitive evaluation tasks. Although COBPE improves the 30-task average, its largest regressions occur on LAMBADA and Dyck Languages. Examining their formats suggests that both are partly driven by evaluation inputs that are atypical under COBPE tokenization. LAMBADA asks the model to predict the final word withheld from a passage. If its prompt ends in an attachable marker, as in “Tom opened the” with target “door”, withholding the target leaves “the” as a standalone base token, although COBPE would normally see it as a modifier. To test this boundary effect, we exclude prompts ending in an attachable determiner or preposition, retaining 2,660 of 5,000 examples. On this subset, the 24-layer COBPE model rises from 29.7 to 35.5 EM, whereas BPE changes from 34.5 to 32.8. Dyck Languages instead consists entirely of bracket strings such as “((())”, where punctuation itself is the content to be modeled. Because COBPE usually composes punctuation with lexical bases, it very rarely encounters such standalone punctuation tokens during pretraining. The original SuperBPE evaluation likewise reports regressions on both tasks (Liu et al., 2025). Together, these cases highlight that some benchmarks are sensitive to tokenizer design, requiring care when comparing tokenization methods.

4.3 Nanochat speedrun

We next evaluate COBPE on the NANOCHAT speedrun benchmark (Karpathy, 2025) which measures wall-clock training time needed to reach GPT-2-level capability, defined by a score of 25.65 on the CORE evaluation suite. The benchmark leaves architecture and training choices open. For a controlled tokenizer comparison, we use the latest NANOCHAT training recipe and configuration available at the time of writing, varying only the tokenization scheme, its corresponding input and output parameterization, and the training horizon.¹² Due to limited resources, we run our experiments on 8 L40S rather than the benchmark’s standard H100. As such, these wall-clock times are not directly comparable to the public NANOCHAT leaderboard, but remain comparable across methods within our setting.

Table 3 shows that COBPE surpasses the BPE reference’s performance with fewer training tokens and shorter wall-clock time.¹³ Overall, this demonstrates that the sequence compression induced by COBPE translates into end-to-end training efficiency.

4.4 Inference speed analysis

To measure inference speed on identical generated text, we prompt the 28-layer vanilla models to copy the same CNN/DailyMail passages (Hermann et al., 2015) and report latency on 250 examples where both models reproduce the target exactly. As shown in Table 4, COBPE uses 32.5% fewer decoding steps and reduces average generation time from 1.94 to 1.51 seconds despite its higher per-token cost. The resulting 1.28 $\times$ speedup shows that sequence compression more than offsets the structured-prediction overhead.

4.5 Compression potential beyond English

Finally, we test whether the same composition principle can shorten sequences beyond English in Spanish, German, Hindi, and Indonesian. For each language, we build 16k-entry BPE and COBPE tokenizers from the same data and vocabulary budget. We derive the COBPE modifier inventories automatically from UD annotations (de Marneffe et al., 2021)

¹²See Appendix E for the exact speedrun configuration and its adaptation to COBPE.

¹³For reference, training BPE for the smaller token budget used by COBPE reaches CORE 23.80.

Metric	BPE	CoBPE
Gen. tokens ↓	120.7	81.5
Gen. time (s) ↓	1.94	1.51
ms / token	16.1	18.6
Speedup ↑	1.00×	1.28×

Table 4: **Inference speed on text copy-ing.** Decoding steps and wall-clock time for 28-layer models prompted to copy CNN/DailyMail passages.

Language	Positions saved
Spanish	31.8%
German	21.2%
Hindi	20.8%
Indonesian	18.0%

Table 5: **Compression beyond English.** Sequence-length reduction from CoBPE relative to BPE with the same 16k vocabulary budget on held-out FineWeb-2 text.

using fixed extraction rules (see Appendix F), alongside shared spacing and punctuation cues, plus capitalization for the Latin-script languages. We measure sequence length directly, leaving multilingual pretraining and downstream evaluation for future work. As shown in Table 5, CoBPE reduces sequence length by 18.0–31.8% across the four languages, suggesting considerable compression opportunities beyond English.

5 Related work

CoBPE connects two lines of work: tokenization for compression, and compositional lexical representations. Standard BPE remains the dominant tokenizer in LMs (Sennrich et al., 2016), with WordPiece (Schuster & Nakajima, 2012; Devlin et al., 2019) and Unigram LM (Kudo, 2018) serving as established alternatives. Recent work shows that tokenizer design affects both compression and downstream quality (Bostrom & Durrett, 2020; Ali et al., 2024; Chizhov et al., 2024), while better compression alone does not guarantee better models (Dagan et al., 2024; Schmidt et al., 2024). Our closest comparison is SuperBPE (Liu et al., 2025), which learns atomic multi-word surface tokens beyond pretokenization boundaries. CoBPE instead represents frequent local markers as reusable modifiers, making the approaches potentially complementary: SuperBPE could learn phrases not already captured by modifier composition. Like tokenization-free methods such as MEGABYTE and BLT, CoBPE aims to cover more text per global modeling step (YU et al., 2023; Pagnoni et al., 2025), but through explicit lexical composition rather than byte patches.

A second connection is to structured lexical representations. Prior work has explored morphology-aware embeddings, tokenization, or pretraining objectives (Matthews et al., 2018; Blevins & Zettlemoyer, 2019; Gan et al., 2022; Cui et al., 2022; Asgari et al., 2025; Lerner & Yvon, 2025). Our input-output parameterization follows the modeling scheme introduced by Reif et al. (2025), but targets grammatical markers rather than morphology. Related NMT systems likewise separate lexical content from surface or linguistic attributes, using inline casing tags or input features (Berard et al., 2019; Etchegoyhen & Gete, 2020) and factored prediction of lemmas with POS, gender, and number (García-Martínez et al., 2016). CoBPE extends this separation to the tokenization and language-modeling interface.

6 Conclusion

We introduced CoBPE, a compositional tokenization scheme that represents text as lexical base tokens with attached surface modifiers. Across two model scales, CoBPE shortens sequences by 30% and improves average downstream performance and end-to-end training efficiency. More broadly, these results show that structure conventionally spread across token positions can instead be modeled within each prediction, making token representation part of the language-modeling interface rather than merely a preprocessing choice. This opens a broad and underexplored design space between surface-form tokenization and fully learned latent interfaces. By allowing each sequence position to carry more information, structured tokenization offers a promising foundation for future language models.

Acknowledgments

This work was supported in part by the Israel Science Foundation (grant no. 2045/21) and by NSF-BSF grant 2020793. We also thank NVIDIA Corporation for support through the NVIDIA Academic Grant Program, which provided access to A100 GPUs used for preliminary experiments.

References

- Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David Mortensen, Noah Smith, and Yulia Tsvetkov. Do all languages cost the same? tokenization in the era of commercial language models. In Houda Bouamor, Juan Pino, and Kalika Bali (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/>.
- Mehdi Ali, Michael Fromm, Klaudia Thellmann, Richard Rutmann, Max Lübbering, Johannes Leveling, Katrin Klug, Jan Ebert, Niclas Doll, Jasper Buschhoff, Charvi Jain, Alexander Weber, Lena Jurkschat, Hammam Abdelwahab, Chelsea John, Pedro Ortiz Suarez, Malte Ostendorff, Samuel Weinbach, Rafet Sifa, Stefan Kesselheim, and Nicolas Flores-Herr. Tokenizer choice for LLM training: Negligible or crucial? In Kevin Duh, Helena Gomez, and Steven Bethard (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/>.
- Zachary Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. Hydra: Sequentially-dependent draft heads for medusa decoding. In *First Conference on Language Modeling*, 2024. URL <https://openreview.net/forum?id=FbhjirzvJG>.
- Ehsaneddin Asgari, Yassine El Kheir, and Mohammad Ali Sadraei Javaheri. MorphBPE: A morpho-aware tokenizer bridging linguistic complexity for efficient llm training across morphologies, 2025. URL <https://arxiv.org/abs/2502.00894>.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021. URL <https://arxiv.org/abs/2108.07732>.
- Amit Ben-Artzy and Roy Schwartz. Spellm: Character-level multi-head decoding. *arXiv preprint arXiv:2507.16323*, 2025.
- Alexandre Berard, Ioan Calapodescu, and Claude Roux. Naver labs Europe’s systems for the WMT19 machine translation robustness task. In Ondřej Bojar, Rajen Chatterjee, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, André Martins, Christof Monz, Matteo Negri, Aurélie Nèveol, Mariana Neves, Matt Post, Marco Turchi, and Karin Verspoor (eds.), *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pp. 526–532, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5361. URL <https://aclanthology.org/W19-5361/>.
- Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiusi Du, Zhe Fu, et al. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.
- BIG-bench. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL <https://openreview.net/forum?id=uyTL5Bvosj>.
- Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

- Yonatan Bitton, Gabriel Stanovsky, Michael Elhadad, and Roy Schwartz. Data efficient masked language modeling for vision and language. In *Findings of EMNLP*, 2021.
- Terra Blevins and Luke Zettlemoyer. Better character language modeling through morphology. *arXiv preprint arXiv:1906.01037*, 2019.
- Kaj Bostrom and Greg Durrett. Byte pair encoding is suboptimal for language model pre-training. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4617–4624, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.414. URL <https://aclanthology.org/2020.findings-emnlp.414/>.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. 2020.
- Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *Forty-first International Conference on Machine Learning*, 2024. URL <https://openreview.net/forum?id=PEpbUobfJv>.
- Anthony Chen, Pallavi Gudipati, Shayne Longpre, Xiao Ling, and Sameer Singh. Evaluating entity disambiguation and the role of popularity in retrieval-based NLP. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli (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. 4472–4485, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.345. URL <https://aclanthology.org/2021.acl-long.345>.
- Pavel Chizhov, Catherine Arnett, Elizaveta Korotkova, and Ivan P. Yamshchikov. BPE gets picky: Efficient vocabulary refinement during tokenizer training. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16587–16604, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.925. URL <https://aclanthology.org/2024.emnlp-main.925/>.
- Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Tamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL <https://aclanthology.org/N19-1300>.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *ArXiv*, abs/1803.05457, 2018. URL <https://api.semanticscholar.org/CorpusID:3922816>.
- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL <https://arxiv.org/abs/2110.14168>.
- Yiming Cui, Wanxiang Che, Shijin Wang, and Ting Liu. Lert: A linguistically-motivated pre-trained language model. *arXiv preprint arXiv:2211.05344*, 2022.
- Gautier Dagan, Gabriele Synnaeve, and Baptiste Rozière. Getting the most out of your tokenizer for pre-training and domain adaptation. *arXiv preprint arXiv:2402.01035*, 2024.

- Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. Universal Dependencies. *Computational Linguistics*, 47(2):255–308, June 2021. doi: 10.1162/coli_a.00402. URL <https://aclanthology.org/2021.cl-2.11/>.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423/>.
- Shizhe Diao, Yu Yang, Yonggan Fu, Xin Dong, Dan Su, Markus Kliegl, Zijia Chen, Peter Belcak, Yoshi Suhara, Hongxu Yin, Mostofa Patwary, Yingyan Lin, Jan Kautz, and Pavlo Molchanov. Nemotron-climb: Clustering-based iterative data mixture bootstrapping for language model pre-training. *arXiv preprint arXiv:2504.13161*, 2025.
- Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL <https://aclanthology.org/N19-1246>.
- Lukas Edman, Helmut Schmid, and Alexander Fraser. CUTE: Measuring LLMs’ understanding of their tokens. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 3017–3026, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.177. URL <https://aclanthology.org/2024.emnlp-main.177>.
- Thierry Etchegoyhen and Harritxu Gete. To case or not to case: Evaluating casing methods for neural machine translation. In Nicoletta Calzolari, Frédéric Béchet, Philippe Blache, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, and Stelios Piperidis (eds.), *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pp. 3752–3760, Marseille, France, May 2020. European Language Resources Association. ISBN 979-10-95546-34-4. URL <https://aclanthology.org/2020.lrec-1.463/>.
- Guobing Gan, Peng Zhang, Sunzhu Li, Xiuqing Lu, and Benyou Wang. MorphTE: Injecting morphology in tensorized embeddings. *Advances in Neural Information Processing Systems*, 35:33186–33200, 2022.
- Mercedes García-Martínez, Loïc Barrault, and Fethi Bougares. Factored neural machine translation architectures. In Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, Rolando Cattoni, and Marcello Federico (eds.), *Proceedings of the 13th International Conference on Spoken Language Translation*, Seattle, Washington D.C, December 8-9 2016. International Workshop on Spoken Language Translation. URL <https://aclanthology.org/2016.iwslt-1.3/>.
- Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In *Forty-first International Conference on Machine Learning*, 2024. URL <https://openreview.net/forum?id=pEWAcejiU2>.
- Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- Akshat Gupta, Jay Yeung, Gopala Anumanchipalli, and Anna A Ivanova. How do LLMs use their depth?, 2026. URL <https://openreview.net/forum?id=Xpd6ZYm4js>.

- Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=d7KBjmI3GmQ>.
- Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/afdec7005cc9f14302cd0474fd0f3c96-Paper.pdf.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pp. 30016–30030, 2022.
- Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. LLMingua: Compressing prompts for accelerated inference of large language models. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 13358–13376, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.825. URL <https://aclanthology.org/2023.emnlp-main.825/>.
- Keller Jordan, Yuchen Jin, Vlado Boza, You Jiacheng, Franz Cesista, Laker Newhouse, and Jeremy Bernstein. Muon: An optimizer for hidden layers in neural networks, 2024. URL <https://kellerjordan.github.io/posts/muon/>.
- Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL <https://aclanthology.org/P17-1147>.
- Guy Kaplan, Matanel Oren, Yuval Reif, and Roy Schwartz. From tokens to words: On the inner lexicon of LLMs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=328vch6tRs>.
- Andrej Karpathy. nanochat: The best chatgpt that \$100 can buy, 2025. URL <https://github.com/karpathy/nanochat>.
- Pride Kavumba, Naoya Inoue, Benjamin Heinzerling, Keshav Singh, Paul Reiser, and Kentaro Inui. When choosing plausible alternatives, clever hans can be clever. In Simon Ostermann, Sheng Zhang, Michael Roth, and Peter Clark (eds.), *Proceedings of the First Workshop on Commonsense Inference in Natural Language Processing*, pp. 33–42, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-6004. URL <https://aclanthology.org/D19-6004>.
- Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In *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/>.
- Paul Lerner and François Yvon. Unlike “likely”, “unlikely” is unlikely: BPE-based segmentation hurts morphological derivations in LLMs. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 5181–5190, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL <https://aclanthology.org/2025.coling-main.348/>.

- Hector J. Levesque, Ernest Davis, and Leora Morgenstern. The winograd schema challenge. In *Proceedings of the Thirteenth International Conference on Principles of Knowledge Representation and Reasoning*, pp. 552–561. AAAI Press, 2012.
- Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruva Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-lm: In search of the next generation of training sets for language models. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 14200–14282. Curran Associates, Inc., 2024. doi: 10.52202/079017-0455. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/19e4ea30dded58259665db375885e412-Paper-Datasets_and_Benchmarks_Track.pdf.
- Alisa Liu, Jonathan Hayase, Valentin Hofmann, Sewoong Oh, Noah A. Smith, and Yejin Choi. SuperBPE: Space travel for language models. *ArXiv*, abs/2503.13423, 2025. URL <https://api.semanticscholar.org/CorpusID:277104160>.
- Austin Matthews, Graham Neubig, and Chris Dyer. Using morphological knowledge in open-vocabulary neural language models. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 1435–1445, 2018.
- Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun’ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL <https://aclanthology.org/D18-1260>.
- OpenAI. tiktoken: A fast BPE tokeniser for use with openai’s models, 2024. URL <https://github.com/openai/tiktoken>.
- Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srini Iyer. Byte latent transformer: Patches scale better than tokens. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9238–9258, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.453. URL <https://aclanthology.org/2025.acl-long.453/>.
- Koyena Pal, Jiuding Sun, Andrew Yuan, Byron Wallace, and David Bau. Future lens: Anticipating subsequent tokens from a single hidden state. In Jing Jiang, David Reitter, and Shumin Deng (eds.), *Proceedings of the 27th Conference on Computational Natural Language Learning (CoNLL)*, pp. 548–560, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.conll-1.37. URL <https://aclanthology.org/2023.conll-1.37/>.
- Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany,

- August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL <https://aclanthology.org/P16-1144>.
- Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849, 2024.
- Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro Von Werra, and Thomas Wolf. Fineweb2: One pipeline to scale them all — adapting pre-training data processing to every language. In *Second Conference on Language Modeling*, 2025. URL <https://openreview.net/forum?id=jnRBe6zatP>.
- Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. Language model tokenizers introduce unfairness between languages. *Advances in neural information processing systems*, 36:36963–36990, 2023.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.
- Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL <https://aclanthology.org/D16-1264>.
- Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019. doi: 10.1162/tacl.a.00266. URL <https://aclanthology.org/Q19-1016>.
- Yuval Reif, Guy Kaplan, and Roy Schwartz. Vocab diet: Reshaping the vocabulary of llms with vector arithmetic. *arXiv preprint arXiv:2510.17001*, 2025.
- Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI) Spring Symposium*, 2011.
- Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: an adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. URL <https://doi.org/10.1145/3474381>.
- Mohammad Samragh, Arnav Kundu, David Harrison, Kumari Nishu, Devang Naik, Minsik Cho, and Mehrdad Farajtabar. Your llm knows the future: Uncovering its multi-token prediction potential. *arXiv preprint arXiv:2507.11851*, 2025.
- Anton Schäfer, Thomas Hofmann, Imanol Schlag, and Tiago Pimentel. On the effect of (near) duplicate subwords in language modelling. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (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/>.
- Craig W Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. Tokenization is more than compression. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (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. doi: 10.18653/v1/2024.emnlp-main.40. URL <https://aclanthology.org/2024.emnlp-main.40/>.
- Craig W Schmidt, Varshini Reddy, Chris Tanner, and Yuval Pinter. Boundless byte pair encoding: Breaking the pre-tokenization barrier. In *Second Conference on Language Modeling*, 2025. URL <https://openreview.net/forum?id=oPAjXGV8qQ>.

- Mike Schuster and Kaisuke Nakajima. Japanese and korean voice search. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5149–5152, 2012. doi: 10.1109/ICASSP.2012.6289079.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith (eds.), *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/>.
- Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL <https://aclanthology.org/N19-1421>.
- Wilson Wu, John Xavier Morris, and Lionel Levine. Do language models plan ahead for future tokens? In *First Conference on Language Modeling*, 2024. URL <https://openreview.net/forum?id=Ba0AvPUyB0>.
- Minghao Yan, Saurabh Agarwal, and Shivaram Venkataraman. Decoding speculative decoding. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6460–6473, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-189-6. doi: 10.18653/v1/2025.naacl-long.328. URL <https://aclanthology.org/2025.naacl-long.328/>.
- An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.
- Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun’ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, Brussels, Belgium, October–November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1259. URL <https://aclanthology.org/D18-1259>.
- LILI YU, Daniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. 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>.
- Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL <https://aclanthology.org/P19-1472>.
- Wanjun Zhong, Siyuan Wang, Duyu Tang, Zenan Xu, Daya Guo, Yining Chen, Jiahai Wang, Jian Yin, Ming Zhou, and Nan Duan. Analytical reasoning of text. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (eds.), *Findings of the Association for Computational Linguistics: NAACL 2022*, pp. 2306–2319, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-naacl.177. URL <https://aclanthology.org/2022.findings-naacl.177>.
- Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. AGIEval: A human-centric benchmark for evaluating foundation models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings*

of the Association for Computational Linguistics: NAACL 2024, pp. 2299–2314, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.149. URL <https://aclanthology.org/2024.findings-naacl.149>.

A Additional results

A.1 Grammatical markers as standalone BPE tokens

Figure 4 breaks down the aggregate categories in Figure 2, showing which individual determiners, prepositions, and punctuation marks account for most of the mass discussed in Section 2.

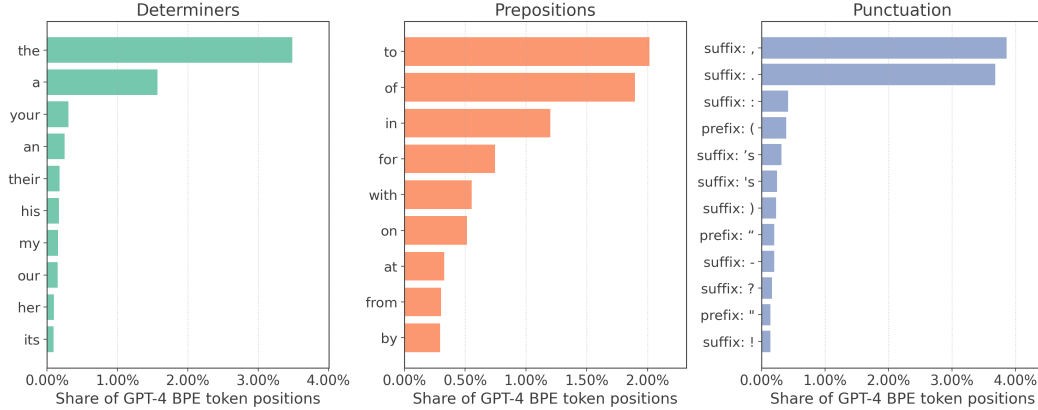


Figure 4: **Detailed breakdown of frequent grammatical markers.** Top marker-level shares within determiners, prepositions, and punctuation under GPT-4 tokenization on FineWeb text.

A.2 Additional downstream results

Table 6 reports the five tasks omitted from Table 2 for readability. These tasks are still included in the 30-task average reported in the main text.

B COBPE implementation details

This section records additional details on the COBPE tokenizer construction. We specify the closed modifier inventories used in the reported experiments and the deterministic, second-stage composition pass.

B.1 Modifier inventories

All reported COBPE experiments use the same English modifier inventory: determiner and preposition groups, prefix- and suffix-punctuation groups, a leading-space indicator, and capitalization indicators for the base token, determiner, and preposition. The determiner and preposition groups contain ten and nine active entries, respectively. Tables 7 and 8 list the exact supported values.

The modifier inventory is deliberately small and restricted to markers that can be identified and reconstructed from their surface form and local token order. No syntactic parsing is used. Listed forms are attached when they occur in an eligible local configuration, without determining their grammatical role. We therefore exclude forms such as “before” and “past”, pronouns and auxiliaries that require role disambiguation, and forms written in ALLCAPS or camelCase. When a form is excluded or cannot be attached, the tokenizer leaves it as an ordinary BPE token in the final representation.

Modifier usage. In FineWeb text, 85.9% of COBPE token positions carry at least one modifier, 13.4% are base-only lexical positions, and 0.7% are standalone inventory markers that could not be attached. Modifier-bearing positions are dominated by the leading-space modifier, which factors surface variation out of the vocabulary but does not itself absorb a separate BPE position.

Category	Task	24-layer (780M)			28-layer (1.3B)	
		BPE	SuperBPE ($t = 27k$)	CoBPE	BPE	CoBPE
Reasoning / symbolic	Repeat-Copy-Logic (EM)	0.0	3.1	0.0	0.0	3.1
Reasoning / symbolic	GSM8K (EM)	2.7	2.4	2.0	2.7	1.3
Reading	HotpotQA (EM)	0.0	0.0	0.0	0.0	0.0
Coding	HumanEval (P@10)	0.0	0.0	0.0	0.3	0.0
Coding	MBPP (P@10)	0.0	0.0	0.0	0.0	0.0

Table 6: **Results for the five tasks omitted from Table 2.** These tasks are included in the 30-task average reported in the main text.

Algorithm 1 CoBPE tokenization at inference time

Require: text x , base tokenizer BPE, modifier inventories

Ensure: base-token ids Y and modifier rows M

```

1:  $R \leftarrow \text{BPE.encode}(x)$ 
2:  $Y \leftarrow [], M \leftarrow []$ 
3:  $p \leftarrow \text{EmptyModifier}()$ 
4:  $i \leftarrow 0$ 
5: while  $i < |R|$  do
6:   if  $R_i$  is attachable suffix punctuation for the previous emitted token then
7:     attach it to the last modifier row in  $M$ 
8:      $i \leftarrow i + 1$ ; continue
9:   end if
10:  if  $R_i$  is attachable prefix punctuation, determiner, or preposition then
11:    record it in pending modifier row  $p$ 
12:     $i \leftarrow i + 1$ ; continue
13:  end if
14:  if a lexical base span starts at position  $i$  then
15:    merge  $p$  with the span's own space/capitalization information
16:    place span-initial modifiers on the first token and suffix punctuation on the last
17:    append the span to  $(Y, M)$ 
18:    clear  $p$ 
19:    advance  $i$  past the span
20:    continue
21:  end if
22:  emit  $R_i$  literally, merging in any pending modifiers
23:  clear  $p$ 
24:   $i \leftarrow i + 1$ 
25: end while
26: if pending modifiers remain then
27:   emit them literally
28: end if
29: return  $(Y, M)$ 

```

B.2 CoBPE composition pass

At inference time, we apply a deterministic left-to-right composition pass to the first-stage BPE stream. The pass maintains a pending modifier state, absorbs eligible prefix punctuation, determiners, and prepositions before a lexical base, and attaches suffix punctuation after the lexical span is identified. Leading-space and capitalization cues are attached to the first token of a multi-token base span, while suffix punctuation is attached to its final token. Tokens that do not participate in a valid local composition remain ordinary base tokens. Algorithm 1 gives the exact pseudocode for the current implementation.

B.3 Optimizing CoBPE tokenization speed

The reported tokenizer is a deliberately simple two-stage research implementation rather than a latency-optimized one. In principle, the same composition logic could be fused

Group	Values used in the reported experiments
Leading space	Binary indicator: absent, present.
Preposition	No modifier, by, at, of, to, in, on, with, for, from.
Determiner	No modifier, the, a, an, my, your, his, her, our, their, its.
Preposition capitalization	Binary indicator: lowercase preposition, titlecased preposition.
Determiner capitalization	Binary indicator: lowercase determiner, titlecased determiner.
Base capitalization	Binary indicator: lowercase base, titlecased base.

Table 7: **Non-punctuation modifier inventory for the reported COBPE runs.** Each group includes a no-modifier state used when that modifier is inactive.

Group	Values used in the reported experiments
Prefix punctuation	No modifier, ", ' , ' , (, [, { , - , --- , ' , ' , \ , "
Suffix punctuation	No modifier, " , ' , ' ,) ,] , } , . , ! , ? , , , ; , : , - , --- , ' , possessive endings 's , s' , 's , s'

Table 8: **Exact punctuation strings supported by the COBPE inventory.** These are closed lists; punctuation outside them remains in the base-token stream.

directly into BPE tokenization, and should bring COBPE encoding speed closer to that of regular BPE. For research and experimentation purposes, we chose to use an explicit second pass for the reported experiments, as it makes the modifier inventory and attachment rules transparent and easy to vary.

C Experimental setup

C.1 Data and tokenizer training

All experiments use ClimbMix data for constructing tokenizers and training models. We keep the training corpus, validation split, and tokenizer-training sample fixed across tokenization variants. All tokenizers use a vocabulary size of 32k.

Tokenizer training follows the dedicated tokenization scripts in the nanochat codebase. The baseline BPE tokenizer is trained on 2B characters from the ClimbMix corpus, with each document truncated to at most 10k characters and random seed 42. The BPE tokenizer trained on space- and capitalization-normalized text that underlies COBPE uses the same 2B-character sample with the same document cap and seed.

C.2 Model configurations

All main models are decoder-only transformers with untied input and output embeddings. Table 9 gives the exact architecture settings for the 24-layer and 28-layer runs.

C.3 Modifier prediction head architecture

We use the base-conditioned gated head described in §3.3. Let M be the total number of modifier classes. The head combines a linear projection of the model’s final hidden state with a gated projection of the selected base-token representation:

$$\ell_i^{\text{mod}} = W_h \mathbf{h}_i + \alpha_i W_b \mathbf{u}_{b_i}, \quad \alpha_i = \sigma(\mathbf{w}_\alpha^\top \mathbf{h}_i)$$

The head then applies a refinement layer that adds a learned nonlinear transformation of the full logit vector:

$$\tilde{\ell}_i^{\text{mod}} = \ell_i^{\text{mod}} + W_r \text{SiLU}(\ell_i^{\text{mod}}).$$

	24 layers	28 layers
Parameters	780M	1.3B
Layers	24	28
Hidden size	1536	2048
Intermediate size	6144	6144
Head dimension	128	128
Query heads	12	16
KV heads	12	16
Training tokens	15.6B	26B

Table 9: **Exact model configurations for the main experiments.**

The refined logits are split into modifier groups. In our inventory, $M = 124$. During teacher forcing, the head receives the gold base token, while during generation it receives the selected base token. The gate is initialized at equal weighting, W_r is initialized to zero, and the modifier-group losses are summed with unit weight.

C.4 Optimization

Across the main matched training-compute runs, we keep the architecture preset, context length, optimizer family (Muon; [Jordan et al., 2024](#)), learning-rate schedule, checkpoint cadence, and data source fixed while varying only the tokenization scheme and, for COBPE, the corresponding structured input and output parameterization. The main runs use a vanilla variant of the NANOCHAT codebase and architecture preset, stripped of the speedrun-specific architecture and optimization tricks (e.g., no value embeddings or residual connections across non-contiguous layers), with a maximum sequence length of 2048, a target parameter-to-data ratio of 20, and the DeepSeek-style 80/10/10 learning-rate step-based schedule ([Bi et al., 2024](#)). The reported speedrun comparison does not include midtraining or SFT. It evaluates the pretraining-to-CORE target portion of the NANOCHAT recipe.

D Evaluation details

We follow SuperBPE’s prompting and scoring protocol for the evaluation tasks. Unless noted otherwise, each task uses five in-context examples. Multiple-choice tasks are scored by accuracy over the answer choices, open-ended tasks are scored by exact match against the target after task-specific normalization of case, punctuation, and articles, following the SQuAD protocol, and coding tasks are scored with pass@10. We evaluate up to 5,000 examples per task, or the full split when the dataset is smaller.

D.1 Knowledge

The knowledge group contains ARC-Easy and ARC-Challenge, which are four-way science multiple-choice benchmarks drawn from grades 3–9 science exams ([Clark et al., 2018](#)); Jeopardy, an open-ended quiz benchmark built from “Jeopardy!” questions,¹⁴ TriviaQA, an open-ended factual QA task ([Joshi et al., 2017](#)); WikidataQA, a factual continuation task; MMLU, which covers broad academic and professional subject areas ([Hendrycks et al., 2021](#)); and OpenBookQA, a science QA benchmark requiring fact retrieval and light reasoning ([Mihaylov et al., 2018](#)).

D.2 Reasoning and symbolic tasks

This group includes Arithmetic on the 2da, 2dm, and 2ds subsets ([Brown et al., 2020](#)); LSAT-AR for analytical-reasoning multiple choice ([Zhong et al., 2022; 2024](#)); BIG-Bench Operators; Repeat-Copy-Logic ([Austin et al., 2021](#)); and GSM8K ([Cobbe et al., 2021](#)). For GSM8K, the

¹⁴<https://www.kaggle.com/datasets/tunguz/200000-jeopardy-questions>

demonstrations include worked solutions with calculator annotations removed, and the model is prompted to produce the final numeric answer after the delimiter #####.

D.3 Reading

The reading-comprehension group contains BoolQ (Clark et al., 2019), CoQA Reddy et al. (2019), DROP (Dua et al., 2019), HotpotQA (Yang et al., 2018), and SQuAD (Rajpurkar et al., 2016). CoQA is evaluated without additional in-context examples because the conversation history already provides the effective demonstrations; the remaining tasks use the standard five-shot protocol.

D.4 Commonsense

The commonsense group contains CommonsenseQA (Talmor et al., 2019), COPA (Roemmele et al., 2011; Kavumba et al., 2019), PIQA (Bisk et al., 2020), Winograd (Levesque et al., 2012), and Winogrande (Sakaguchi et al., 2021), all scored by exact match over the provided answer options.

D.5 Language understanding

This group contains HellaSwag (Zellers et al., 2019), LAMBADA (Paperno et al., 2016), and BIG-Bench Language Identification. HellaSwag is four-way multiple choice, while LAMBADA and Language Identification are open-ended exact-match tasks.

D.6 String and formal tasks

The string/formal group contains BIG-Bench CS Algorithms (BIG-bench, 2023), CUTE (Edman et al., 2024), and BIG-Bench Dyck Languages, all scored by exact match on the generated answer string.

D.7 Coding tasks

The coding tasks are HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021), both evaluated zero-shot, following prior work. For HumanEval, we use the stop tokens `\nclass`, `\ndef`, `\n#`, and `\nif`; sample 20 continuations with $\text{top-}p = 0.95$ and temperature 0.8; and cap generation at 128 new tokens. MBPP uses the same decoding and functional-correctness setup. We report `pass@10`, estimated from the 20 sampled solutions per problem.

E Speedrun details

We use the latest NANOCHAT GPT-2 training recipe and speedrun configuration available as of July 24, 2026. We run both methods on 8 L40S GPUs rather than the benchmark’s standard H100 hardware and keep the backbone and per-step optimization configuration fixed. We vary the tokenization, the corresponding COBPE input and output parameterization, and the training horizon between methods.

Table 10 gives the exact 24-layer configuration. Relative to the H100 recipe, the L40S runs disable FP8 training and use the full-window attention pattern L rather than the short-window SSSL pattern. The reported COBPE speedrun uses the same base determiner profile and existing preposition profile as the main experiments.

E.1 NANOCHAT-specific residual mechanisms

The speedrun recipe uses smear, which mixes a gated amount of the previous input embedding into the current one, and backout, which subtracts a learned fraction of a middle-layer representation before next-token prediction.

Setting	Value
Backbone scale	24 layers, hidden size 1536, intermediate size 6144, head dimension 128.
Attention heads	12 query heads and 12 KV heads.
Vocabulary / context	32k vocabulary, maximum sequence length 2048.
Hardware	8× L40S GPUs.
Batching	Device batch size 16; total batch size 1.05M tokens.
Target param:data ratio	8.0 for BPE; 6.5 for the COBPE variant
Architecture details	Residual scalars, value embeddings on alternating layers, QK-norm scale 1.15, and MLP input-projection init scale 0.5. Both BPE and COBPE use smear, which mixes the previous input embedding into the current one, and backout, which removes a learned middle-layer component before prediction. For COBPE, these operate only on the base stream.
Optimization details	40 steps of warmup, Muon final momentum of 0.97, and LR linear decay in the last 65% steps
Attention pattern on L40S	Full-window pattern L. Relative to the H100 recipe, the L40S runs do not use FP8 and do not use the short-window SSSL pattern.
Data input	ClimbMix, tokenized on the fly.

Table 10: **Exact configuration for the reported NANOCHAT speedrun comparison.** Both methods share the backbone and per-step optimization configuration. The target parameter-to-data ratio sets the training horizon for each method.

For COBPE, both mechanisms operate only on the lexical base stream. Smear precedes the addition of modifier embeddings, backout affects only the base-token prediction, and modifier prediction uses the pre-backout transformer representation. Modifier embeddings therefore participate in the shared transformer computation without forming a separate residual stream.

F Multilingual tokenization analysis

This appendix details the multilingual tokenizer analysis summarized in §4.5. We compare matched BPE and COBPE tokenizers, describe the automatically extracted modifier inventories, and document the validation procedure. The full inventories reduce positions by 18.0–31.8% across the four languages.

F.1 Data and construction

We analyze Spanish, German, Hindi, and Indonesian on FineWeb-2 (Penedo et al., 2025). For each language, we build matched 16k-entry BPE and COBPE tokenizers from the same training text and vocabulary budget, then measure sequence length on held-out text. All languages use the same whitespace-delimited lexical bases, spacing modifiers, and punctuation modifiers, with capitalization modifiers for the Latin-script languages. The training split is capped at 200,000 documents, with at most 10,000 Unicode characters per document. Evaluation uses 10,000 documents from the disjoint test split, capped at 10,000 characters per document. We report tokenizer compression only.

F.2 Automatic modifier extraction

We derive language-specific determiner and adposition inventories automatically from Universal Dependencies v2.18 annotations. The extraction retains frequent forms with consistent annotations and a consistent position relative to the lexical base, producing closed lists of surface strings that can be applied with exact matching and local attachment rules, without a language-specific parser. Forms that do not meet the filters remain ordinary BPE tokens. Hindi additionally uses a Devanagari-aware split and longest-match rule for compound postpositions, in which a multiword expression such as *ke liye* (“for”) functions

as a single postposition. Capitalization modifiers are used for Spanish, German, and Indonesian.

The inventories are conservative rather than exhaustive. For Indonesian, they contain four pre-base determiners and nine pre-base adpositions. Because the shared modifier interface provides one determiner group for pre-base markers, post-base determiners and multiword or affixed components remain ordinary BPE tokens.

Language	Determiners	Adpositions	Attachment direction
Spanish	17	14	before
German	24	16	before
Hindi	5	19 + 15 compound	before / after
Indonesian	4	9	before

Table 11: **Language-specific modifier inventories.** Counts give the retained surface forms after frequency, annotation-consistency, and direction filtering. Attachment indicates whether the markers normally occur before or after the lexical base. Hindi reports ordinary adpositions plus compound postpositions.

G Additional related work

Other recent work aims to make each autoregressive step carry more future information, but without changing tokenization. Hidden-state analyses show that representations at one position already contain substantial signal about subsequent tokens (Pal et al., 2023; Wu et al., 2024; Samragh et al., 2025). Multi-token prediction exploits this by training models to predict multiple future tokens from a shared trunk (Gloeckle et al., 2024), while related inference-time approaches propose or decode multiple future tokens or characters and then verify them (Cai et al., 2024; Ankner et al., 2024; Ben-Artzy & Schwartz, 2025; Yan et al., 2025). These methods are complementary to ours: they modify the objective or decoding procedure, whereas COBPE changes the representational unit. Analyses of depth also suggest that easy, high-frequency items are resolved earlier in the network, with function words and punctuation predicted earlier on average than nouns and verbs (Gupta et al., 2026), consistent with our motivation for modeling such markers compositionally.