METADATA-ALIGNED DECODING FOR ACCURACY GAINS

Anonymous authors
Paper under double-blind review

ABSTRACT

While metadata is commonly used to curate datasets, its potential as a direct decoding signal remains under-explored. We propose LIME (**Li**nguistic **M**etadata **E**mbeddings), a method that enriches token embeddings with linguistic metadata capturing syntax, semantics, and context. LIME substantially improves decoding accuracy across model sizes from 500M to 2B, while introducing only 0.01% additional parameters at negligible compute overhead. Beyond accuracy, LIME improves tokenization and generative task performance. These benefits persist across model scales, and the decoding accuracy gains extend to five typologically diverse languages, where language modeling improvements scale predictably with tokenizer fertility. The gain stems from metadata that anticipates the token to be generated. Supplied directly, as in our guided variant LIME-G, it raises decoding accuracy by over 40% and outperforms constrained decoding in reasoning and arithmetic. Overall, we establish metadata as a powerful interface for controllable generation, demonstrating that inference-time steering can unlock dormant capabilities in reasoning and arithmetic.

1 INTRODUCTION

Training large language models (LLMs) is computationally expensive and highly data-intensive. Smaller LLMs are particularly attractive because of their reduced resource requirements and accessibility. Nonetheless, compute-optimal scaling prescribes that training data grows proportionally with model size (Hoffmann et al., 2022), imposing substantial data requirements even on smaller models.

At the same time, the availability of novel human-generated high-quality training data is decreasing (Xue et al., 2023; Villalobos et al., 2024), emphasizing the need to improve the utility of existing datasets. To compensate for this shortcoming, methods of accumulating LLM pre-training datasets shift from mere quality filtering to synthesis through earlier model generations and staging of increasing data quality buckets (Su et al., 2025). In order to determine the stage and quality bucket, existing document-level metadata is used, along with more complex—and even model-based—scores that reflect attributes such as educational value, or factual reliability (Schuhmann et al., 2022; Li et al., 2024; Penedo et al., 2024; Wettig et al., 2025).

However, neither pre-existing nor created metadata are typically propagated downstream into the model during training. Modern LLM tokenizers are typically trained on yet another blended dataset with solely text compression as objective, neglecting linguistic research entirely. As such, tokenization can fragment meaningful content, distort sequence relationships, and ultimately degrade the efficiency and quality of model learning. Recent work indicates that linguistically motivated segmentation can improve model training (Hou et al., 2023; Schmidt et al., 2024). Moreover, early work suggests that linguistic token annotation can improve certain modeling capabilities such as in machine translation (Sennrich & Haddow, 2016).

To this end, we introduce a method which rigorously integrates token-grained linguistic metadata into LLM pre-training at negligible complexity and computational overhead: LIME (Linguistic Metadata Embeddings for LLMs). Our method augments the pre-trained subword tokenizer with linguistically informed annotations, namely part-of-speech (POS) and named entity recognition (NER) tags, extending its standard output of raw tokens with token-aligned metadata. We propagate metadata downstream by incorporating it as additional input signals to shift the token embedding space. As a guidance variant, LIME-G shifts the metadata embeddings by one, so that generation

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

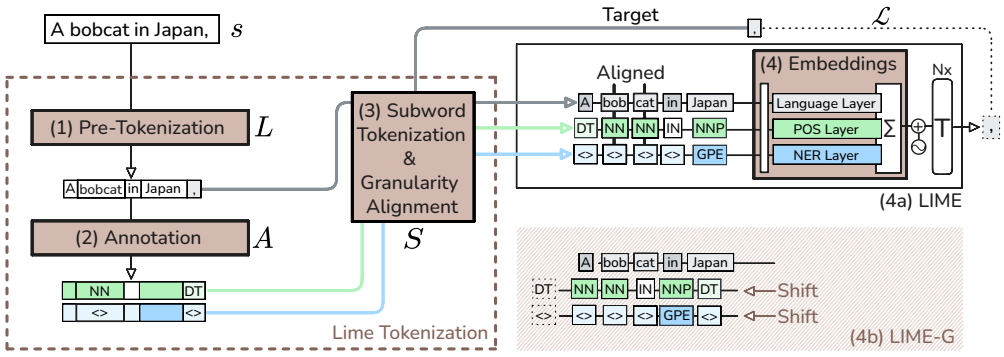


Figure 1: LIME and LIME-G architecture. **(1)** Input text s is split by the linguistic tokenizer (L). **(2)** Linguistic splits are annotated, e.g. with POS and NER tags. **(3)** Subword tokenization (S) is applied to the linguistic tokens and annotations are aligned to the new splits. **(4)** Tokens and metadata are embedded, fused together and passed into consecutive transformer blocks.

is guided by the metadata of the token to be generated. We call decoding with such token-aligned metadata, in both variants, metadata-aligned decoding.

In our experiments, we demonstrate that LIME improves decoding accuracy at every model size and reaches the final accuracy of its baseline with up to 56% fewer training tokens. Additionally, LIME mitigates issues caused by artificial tokenization splits, keeping the meaning of subword tokens together, as supported by both qualitative and quantitative analyses. LIME's accuracy gain stems from metadata that anticipates the token to be generated. Given that metadata directly, LIME-G achieves over 40% higher accuracy (about 20pp), and in reasoning and arithmetic, where the required class is naturally available, guided decoding outperforms constrained decoding. These benefits persist across 500M, 1B and 2B parameter models, and the accuracy gain holds at 7B. Together, these results establish linguistic metadata as a decoding signal that both sharpens prediction and steers generation.

Our main contributions and findings are summarized as follows:

1. We introduce LIME, per-token linguistic metadata embeddings that add about 0.01% parameters, and its guided variant LIME-G (Sec. 3).
2. LIME improves decoding accuracy at every scale from 500M to 7B, excels in generative tasks, and keeps fragmented words and entities together (Sec. 4.2 to 4.4).
3. The gain comes from metadata that anticipates the token to be generated. Given that metadata, LIME-G raises decoding accuracy by over 40% and outperforms constrained decoding in reasoning and arithmetic (Sec. 4.4, 4.5).
4. Models learn to annotate their own output and keep the gains (Sec. 4.6). LIME's gains grow with tokenizer fertility across five languages (Sec. 4.7).

2 Related Work

Pre-Tokenization and Tokenizers. Pre-tokenization defines segment boundaries for subword tokenization by normalizing and splitting text (e.g., on whitespace or punctuation) into coherent units. Subword tokenizers, trained with compression-based methods like BPE (Sennrich et al., 2016), inherit biases from their training data: Ahia et al. (2023) report large cross-lingual disparities, with some languages requiring up to five times more tokens for the same content. Tokenizers optimized for one distribution may become inefficient under distribution shifts (Ahia et al., 2023; Deiseroth et al., 2024; Neitemeier et al., 2025). Thus, fragmentation, or more tokens per word, correlates with poorer model performance. Linguistically informed segmentation can improve results: Hou et al. (2023) find morphological splits reduce perplexity and maintain or improve downstream accuracy, while Schmidt et al. (2024) show ignoring morphology in pre-tokenization can harm performance, as do Fujii et al. (2023) for Japanese. Recent work also explores byte-level or tokenizer-free alternatives (Xue et al., 2023; Yu et al., 2023; Deiseroth et al., 2024).

Metadata in LLM Pre-training. LLM downstream performance is strongly influenced by the quality of pre-training data (Longpre et al., 2024; Wettig et al., 2024). To improve quality, pre-training data is filtered and deduplicated using metadata typically derived from heuristic approaches (Raffel et al., 2020; Rae et al., 2021) or model-based classifiers (Brown et al., 2020; Xie et al., 2023; Penedo et al., 2024; Li et al., 2024; Su et al., 2025). Further, several approaches have been proposed that, instead of leveraging metadata solely to improve data quality, propagate metadata directly into model training. For example, prior work prepends source-domain, temporal, or language metadata as control codes (Keskar et al., 2019; Dhingra et al., 2022; Liu et al., 2020; Khalifa et al., 2024), and recent studies show that even a single prepended token can substantially improve capacity and pre-training efficiency (Allen-Zhu & Li, 2025; Gao et al., 2025). While effective, these methods rely on the existing vocabulary to encode metadata, which consumes valuable input token space, and they typically operate at the document level, limiting annotation granularity.

Auxiliary Supervision with Linguistic Annotations. Incorporating fine-grained linguistic token annotations into neural NLP models has been widely studied. For example, Sennrich & Haddow (2016) showed that features such as POS tags and dependency relations can improve neural machine translation. These ideas have since been extended to pre-training via entity-aware masking (Sun et al., 2021) and syntax-aware attention (Zhang et al., 2022). Knowledge-enhanced pre-trained language models (KEPLMs) (Hu et al., 2024) integrate structured information, such as knowledge graphs. Yet, these approaches substantially increase architectural complexity and computational cost.

Guided and Constrained Decoding. Constrained decoding masks the output distribution to the tokens a regex, grammar or schema admits (Willard & Louf, 2023; Geng et al., 2023; Scholak et al., 2021; Poesia et al., 2022), and Grammar-Aligned Decoding (Park et al., 2024) reweights sampling so that the constrained outputs follow the model's own distribution. Both act on the output distribution after the forward pass. `LIME-G` instead conditions the input embedding on the class of the token to be generated (Sec. 3).

Embedding-Level Metadata Integration. Embedding layers serve as an effective mechanism for metadata injection, enabling structured information to be incorporated directly into the model's representational space. For instance, retrieval-augmented and knowledge-enriched encoders have been explored (Guu et al., 2020; Sun et al., 2020). CUE (Novotney et al., 2022) incorporates metadata, such as author and date, into LLMs through a separate context encoder. Furthermore, McLeish et al. (2024) showed that embedding positional cues into the representations of numerical tokens can enhance performance on arithmetic tasks. Levine et al. (2020) and Armengol-Estapé et al. (2021) investigate the extension of token embeddings with linguistic information in bidirectional transformer models. Nevertheless, incorporating fine-grained linguistic metadata into the token embeddings of autoregressive LLMs remains an open research direction. We address this gap with `LIME`, which adds per-token metadata to a decoder's input embeddings and, as `LIME-G`, conditions decoding inside the model on the metadata of the token to be generated.

3 LIME: Linguistic Metadata for LLMs

`LIME` consists of four stages: (1) linguistic pre-tokenization, (2) metadata annotation, (3) subword tokenization & granularity alignment, and (4) metadata embedding, as shown in Fig. 1. These stages are further illustrated in Tab. 1.

(1) Linguistic Pre-Tokenization. We define a tokenizer on the alphabet Σ as the tuple $T = (f, V)$ consisting of the function $f : \Sigma^* \to V^*$ that splits a given text $s \in \Sigma^*$ into a sequence of tokens of the token vocabulary V (Minixhofer et al., 2024). A token sequence is denoted as $T(s) = t_1, t_2, ..., t_n$ of length n. Unlike the conventional approach of using only a statistically learned subword tokenizer S, we introduce a rule-based linguistic tokenizer L for pre-tokenization. Introducing L allows for effective and linguistically informed segmentation into minimal meaningful text units, enabling the assignment of fine-grained metadata labels.

(2) Metadata Annotation. The metadata annotation process is defined by an annotator $A = (g, C)$ with the annotation function $g : V^n \to C^n$ that, for a given token sequence $T(s)$ of length n, produces an annotation sequence $A(T(s)) = a_1, a_2, ..., a_n$ with $a_i \in C$ and C being the set of pre-defined annotation symbols. The annotation function g can consist of rule-based methods, heuristics

Table 1: Illustrating LIME and LIME-G with the example of Fig. 1. {} denote embeddings.

(0)	Input Sequence	A␣bobcat␣in␣Japan				
(1)	Linguistic Pre-Tokenization	A	␣bobcat		␣in	␣Japan
(2)	Metadata Annotation	DT	NN		IN	NNP
(3)	Subword Tokenization & Granularity Alignment	A DT	␣bob NN	cat NN	␣in IN	␣Japan NNP
(4a)	Metadata Embeddings LIME	{ A } +{ DT }	{ ␣bob } +{ NN }	{ cat } +{ NN }	{ ␣in } +{ IN }	{ ␣Japan } +{ NNP }
(4b)	Metadata Embeddings LIME-G	{ A } +{ NN }	{ ␣bob } +{ NN }	{ cat } +{ IN }	{ ␣in } +{ NNP }	{ ␣Japan } +{ X }

or classification models. As illustrated in Fig. 1, our method allows defining and integrating multiple annotators, e.g., POS and NER annotations.

(3) Subword Tokenization & Granularity Alignment. Given the subsequent subword tokenizer $S = (f_S, V_S)$, its vocabulary and tokenization function will naturally differ from those of the linguistic tokenizer L used in the first stage, as shown in Fig. 1. In general $n = |L(s)| \neq |S(s)| = m$ for an input text s, so the two sequences must be aligned. In the case of $n < m$, we define the annotation function $g' : V^n \to C^m$ that resolves this granularity mismatch as follows: g' tracks for every token in $S(s)$ its word context defined by $L(s)$ and repeats the relevant annotation until a granularity match is achieved. In the case of $n > m$, where L splits an input sequence into more tokens than S, we keep the finer granularity of L by expressing every token in $L(s)$ with tokens from the vocabulary V_S. For example, the word "don't" could linguistically be split as $L(s) =$ `do n ' t` while the subword tokenizer may produce $S(s) =$ `don ' t`. In that case we keep the slightly less compressed word boundaries of $L(s)$ and their corresponding annotations as the subsequent tokenization of S, to not lose annotation precision. Now, for a set of metadata domains D, and $|D|$ metadata annotators, the final output of this stage is a granularity aligned sequence of tuples $(t_i, (a_{i,d})_{d \in D})$ for token index i. We refer to these first three stages as `LIME Tokenization`.

(4) Metadata Embeddings. In our output tuple $(t_i, (a_{i,d})_{d \in D})$ at index i, token t_i is mapped to its token embedding $E_X(t_i)$ via an index lookup into the embedding matrix $W^X \in \mathbb{R}^{|V_S| \times h}$, where h is the hidden size. From here we introduce two variants to blend in token annotations. The first, termed LIME (see Fig. 1 top), uses metadata embeddings E_M^d originating from the same token:

$$E(t_i) = E_X(t_i) + \sum\nolimits_{d \in D} w_d E_M^d(a_{i,d}) \; . \tag{1}$$

Note that the annotation embedding process is applied to individual respective matrices $W^{(d)} \in \mathbb{R}^{|C_d| \times h}$ producing $E_M^d(a_{i,d})$. E_X combined with the weighted sum of $|D|$ metadata embeddings creates the final embedding and LLM input $E(t_i)$.

The second implementation (see Fig. 1 bottom), termed LIME-G (guided), is tailored for scenarios where the metadata annotation of the token to be generated (a_{i+1}) is available, as in Sec. 4.5. During training and inference (inference in Fig. 4b), the base embedding E_X is augmented using the metadata embeddings of the *next* token, rather than the current one. This guided embedding is defined as:

$$E(t_i) = E_X(t_i) + \sum\nolimits_{d \in D} w_d E_M^d(a_{i+1,d}) \; . \tag{2}$$

The metadata a_{i+1} can come from the task (a number follows an equation sign) or from a user. The model can also annotate its own output (LIME-A, Sec. 4.6). We refer to decoding with token-aligned metadata, as in LIME and LIME-G, as metadata-aligned decoding. Note that we modify only the token embeddings; all other transformer components, including the LLM head and the cross-entropy loss applied during training, remain unaltered and fully agnostic to the metadata.

4 Experiments

We evaluate LIME and LIME-G on decoding accuracy (Sec. 4.2), downstream benchmarks (Sec. 4.3), token-level analysis (Sec. 4.4), reasoning and arithmetic (Sec. 4.5), self-annotation (Sec. 4.6), and five languages (Sec. 4.7).

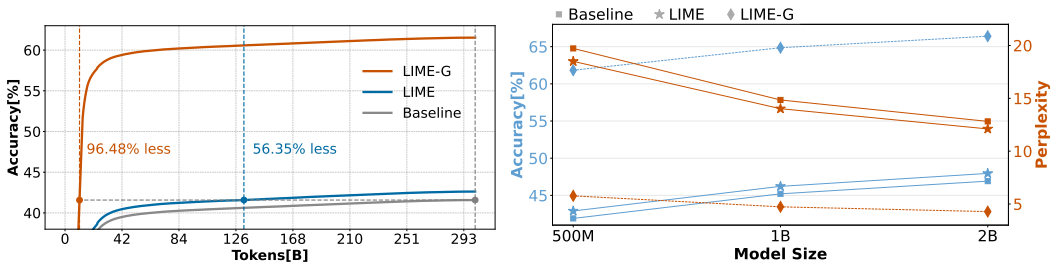


Figure 2: **Left:** Decoding accuracy improves with metadata embedding layers. $\texttt{LIME}_{\texttt{500M}}$ reaches the final accuracy of `Baseline` with 56% fewer training tokens and maintains a stable advantage. **Right:** Accuracy and perplexity improvements translate consistently across model sizes. See Tab. 7 for exact values.

4.1 Experimental Setup

In our experiments, we followed the Gemma architecture (Mesnard et al., 2024) and pre-trained `LIME` models in three sizes, 500M, 1B, 2B, on 302 billion tokens of the `DCLM-BASELINE` (Li et al., 2024) dataset (cf. App. A.1). We applied the three-stage `LIME Tokenization` process described in Sec. 3, with L being the English spaCy tokenizer[1] and S the SentencePiece (Kudo & Richardson, 2018) Gemma tokenizer, and otherwise optimized the standard cross-entropy loss $\mathcal{L}$ on the language head (disregarding metadata). spaCy annotates the full input, in training and in evaluation. For generative tasks, the input is the prompt and the text generated so far. Applying `LIME Tokenization` to 10,000 `DCLM-BASELINE` samples increases the number of tokens needed to encode the same data by 1.19%, because S is applied independently to each linguistic unit, preventing certain cross-boundary merges. As a side effect, the set of distinct tokens observed in this sample is reduced by 1.03%. Further details are found in App. A.3. Models trained with `LIME Tokenization` but no additional metadata embedding layers are referred to as `Baseline` (Base in tables) in the following.

We extended the `Baseline` models with two metadata domains available in spaCy: Syntactic Part of Speech (POS), with $|C_{\text{pos}}| = 51$ (App. A.14), and semantic Named Entity Recognition (NER), with $|C_{\text{ner}}| = 20$ (App. A.15). Both annotation embedding layers are weighted equally with $w_{\text{pos}} = w_{\text{ner}} = 1$ (Eq. 1). For both `LIME` and `LIME-G`, the additional embedding layers, having 71 entries in total, add about 0.01% to the total parameters. Since they require only one vectorized lookup and addition in the forward pass, they add negligible runtime overhead. For details, see App. A.3. Decoding accuracy denotes the share of tokens the model predicts correctly, given the preceding text and its metadata. `LIME` and `LIME-G` inference is illustrated in App. A.2 (Fig. 4a, 4b).

4.2 Linguistic Metadata Improves Decoding Accuracy

Metadata embeddings raise decoding accuracy at every model size and throughout training. Fig. 2 (left) shows the 500M model. `Baseline` and `LIME` follow the usual pattern of rapid early gains, which plateau after about 42B tokens; from there, `LIME` keeps a stable advantage and reaches the final accuracy of its `Baseline` (41.90%) using 56.35% fewer tokens, as extended pre-training of the baseline cannot replace the information the metadata carries. `LIME` keeps its advantage at all sizes and reaches the final `Baseline` accuracy with 44.6% and 34.8% fewer tokens at 1B and 2B (App. A.4). At 7B, with 50B FineWeb-Edu training tokens, the gain is $+1.24$pp (App. A.11). Sec. 4.4 shows that this gain stems from annotations anticipating the token to be generated.

Further, we evaluate model quality across sizes on a `DCLM-BASELINE` test set of 10,000 samples, reporting decoding accuracy and perplexity on the same batches of data (Fig. 2 right). We observe that all `LIME` models have improved accuracy and reduced perplexity relative to their `Baseline` models. `Baseline` outperforms models trained with the standard Gemma tokenizer by 0.75pp accuracy and 0.6 perplexity on average, each on its own token sequence (App. A.4), so LIME Tokenization alone already helps. `LIME`'s gain over `Baseline` is roughly constant at $+1$pp across model sizes.

[1] `https://spacy.io/api/tokenizer`

Table 2: LIME excels at generative tasks. Improvements to Base are indicated by ↑, to LIME by ⇑. We highlight (yellow) generative-format tasks. Exemplified on 500M, other model sizes in App. A.5. LIME−G is guided by the metadata of the reference continuation.

	Base_{500M}	LIME_{500M}	LIME-G_{500M}
ARC-Easy	57.00±1.57	↑57.20±1.57	⇑58.10±1.56
BoolQ	49.50±1.58	47.90±1.58	⇑59.40±1.55
COPA	62.00±4.88	↑64.00±4.82	61.00±4.90
HellaSwag	36.20±1.52	↑37.00±1.53	⇑43.10±1.57
LAMBADA	26.50±1.40	↑29.80±1.45	⇑49.00±1.58
OpenBookQA	31.00±2.07	↑32.60±2.10	⇑34.20±2.12
PIQA	69.90±1.45	69.40±1.46	68.20±1.47
TriviaQA	8.20±0.87	↑ 9.80±0.94	⇑19.50±1.25
WinoGrande	51.70±1.58	↑52.60±1.58	⇑54.60±1.58
Mean	43.56±1.88	↑44.48±1.89	⇑49.68±1.95

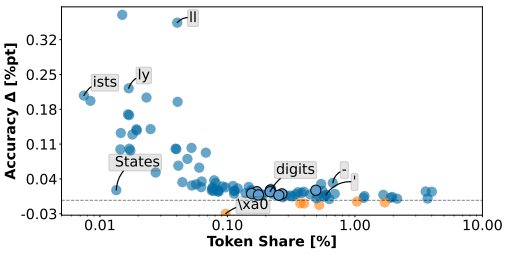


Figure 3: LIME_{500M} decoding accuracy increases within semantic and syntactic metadata class boundaries: Among the 100 most impactful (share×Δ) tokens, our model exhibits improved token coupling for suffix-, digit-tokens and an exemplary entity group token ␣States.

Perplexity reduction is, as expected in this range, more pronounced in smaller model sizes. LIME−G shows a substantial improvement in accuracy and perplexity when guided by the metadata of the token to be generated. Relative to Baseline models, accuracy increases by over 40% across model sizes, while perplexity is reduced by more than 65%, enabling our LIME-G_{2B} model to achieve a perplexity of 4.28. Learning curves are in App. A.4.

4.3 LIME Models Excel in Generative Tasks

LIME's downstream gains concentrate on generative tasks. We evaluate downstream transfer on nine standard benchmarks (Tab. 2), highlighting results for the 500M model; comparable trends hold for all sizes, detailed in App. A.5. Each task was run with up to 1,000 samples using randomly selected 5-shot contexts.

The results show that LIME models on average slightly improve over their respective Baseline models. On tasks that are evaluated using generative greedy sampling (LAMBADA and TriviaQA), our method improves model performance substantially at 500M and 1B.[2] Specifically, for our 500M model, LAMBADA increased from 26.50 to 29.80 and TriviaQA from 8.20 to 9.80. At 2B, LIME improves COPA (77.0 to 81.0), ARC-Easy (71.2 to 73.4), and OpenBookQA (39.8 to 42.0); full results across all scales are reported in Tab. 8. LAMBADA improves at all four sizes up to 7B and TriviaQA at three, and at 7B LIME improves 7 of 9 tasks, including both generative ones (App. A.5, A.11). LIME−G models, guided by the metadata of the reference continuation, improve on 7 of 9 tasks at 500M and 1B, and on 5 at 2B, compared to both Baseline and LIME. The improvement of LIME models on greedily sampled tasks is further amplified by LIME−G models. Relative to Baseline, LIME−G improves TriviaQA by 138%, 89% and 64% and LAMBADA by 85%, 61% and 53% at 500M, 1B and 2B.

4.4 LIME Keeps Tokens Together

Building on the previous finding that LIME boosts performance in generative tasks, we examine where these improvements arise at the token level. We first study accuracy shifts of individual tokens, then accuracy by word length across model sizes, and finally what the annotations encode.

Suffix Tokens, Entity-Groups & Digits. In the following, we take the dataset of Sec. 4.2 and rank all unique tokens by their shift in decoding accuracy weighted by their share in the set. Illustrating accuracy shifts of the 100 most impactful tokens (Fig. 3) reveals: Suffix tokens such as ing and ly, which end multi-token words, exhibit large positive accuracy shifts. This also holds for apostrophe and hyphen tokens, as well as entity-group tokens, i.e. consecutive tokens that share the same entity metadata class, such as ␣States from "United States". Finally, we observe a constant accuracy improvement across all single-digit tokens 0-9. The shifts sum to roughly the total accuracy gain of LIME_{500M} as reported in Sec. 4.2 (more details in App. A.6).

[2]The remaining tasks are multiple-choice, scored by comparing the log-likelihoods of the answer options.

Keeping Natural Language Word Tokens Together. Additionally, in Tab. 3 (left), we count occurring words of certain token lengths and show their averaged accuracies.[3] We observe that LIME models outperform their baselines across all cases, and notably, for word lengths $x \geq 2$.[4] For single-token words ($x = 1$), we still obtain gains of roughly 0.7pp. This highlights that metadata embeddings not only have a strong effect on the cohesion of subword sequences, but also help on single-token words. LIME-G models further amplify the previously described improvements. As expected, LIME-G's metadata of the token to be generated improves decoding accuracy on single-token words by +21pp. LIME-G more than doubles the gain on multi-token words, to +10pp.

What Annotations Encode. Metadata is assigned per word and repeated for every subword (Sec. 3), so the tag of the first piece already describes the word or entity being built. The annotation anticipates the token to be generated, and this is where LIME's gain over Baseline (Sec. 4.2) concentrates, as word and entity continuations gain most (Tab. 3, Fig. 3). We train 500M models on the same 10B tokens, differing only in how their annotations are obtained. Annotations from the full text give +0.84pp, from the preceding text alone no gain (−0.09pp), and with one token of right context +0.90pp. More than half of the gain lies on the 4.6% of positions where the tag without right context differs from spaCy's (App. A.10). Given directly, as in LIME-G, the metadata of the token to be generated raises decoding accuracy by about 20pp.

4.5 Guided Decoding Outperforms Constrained Decoding

While in open-ended generation the class of the token to be generated is unknown, many applications operate under strict output constraints where it is fixed. In tasks such as structured data extraction (Willard & Louf, 2023; Geng et al., 2023) and executable code generation (Scholak et al., 2021; Poesia et al., 2022), the validity of that token is often governed by a known grammar or schema. While hard constraints ensure validity, they can sometimes degrade reasoning performance (Schall & de Melo, 2025). In contrast, we explore embedding guidance (Sec. 3). We evaluate LIME-G guided by the class the task requires on reasoning (FLenQA) and arithmetic (ARI-ADD), and compare it with constrained decoding.

Task Definition. FLenQA (Levy et al., 2024) requires reasoning across multiple, increasingly noisy contexts. We use the FLenQA task "People in Rooms", which requires combining two pieces of information found in a given noisy context. Both facts are required to answer the question, as in the following example: *"John's room is blue walled [...] Ethan is in John's room [...] Is Ethan in a blue walled room?"* To evaluate our models that are trained on DCLM-BASELINE only, we convert the questions into the following generative format: *"John's room is blue walled [...] Ethan is in John's room [...] Ethan is in a room. It has the following properties:"* and greedily sample for the expected property *" blue walled"* to appear in the subsequent words. In this example, we steer $\texttt{LIME-G}_{\texttt{2B}}$ to predict an adjective token (JJ) outside any entity (<>) by giving the last context token the target metadata (App. A.2, Fig. 4b). The benchmark is grouped into five noise levels (250 to 3000 tokens of irrelevant text), each with 400 samples.

For arithmetic, we use the pre-trained models without fine-tuning and the prompt *"The result is: {number}+{number} = "*, which performed best. The greedily sampled completion is then compared with the correct numerical result, thus testing basic symbolic arithmetic capabilities in a generative setting. We randomly sample $5 \leq$ *{number}* ≤ 49 and average the results over two draws of 500 unique number pairs.

Results. In Tab. 3 (right), we report performances across the different FLenQA variants and the ARI-ADD task. For all tasks we show the 2B accuracies (all sizes in App. A.7). At 2B, LIME already improves over Baseline, especially for shorter contexts (e.g., +15.8pp on FLQA-500). Here no annotator sees the token to be generated, since the text is annotated as it is produced. LIME-G delivers double-digit gains on every variant, still reaching 39.3% on FLQA-3000 and gaining +38pp on FLQA-250. The gain holds under long noisy contexts. Guidance outperforms constrained decoding under a class constraint. Restricting Baseline's first generated token to adjectives (constr.) yields 65.5 on FLQA-500, against 73.8 for LIME-G in the same evaluation (Tab. 14), with the same ordering

[3] Details are found in App. A.6.

[4] Note that the category of tokens in words of $x \geq 2$ accounts for 8.4% of the number of single-token words, since we apply an English-optimized tokenizer to a primarily English corpus.

Table 3: **Left:** LIME improves decoding accuracy on tokens in words of $x \geq 2$ tokens and slightly on single-token words ($x = 1$). **Right:** On FLenQA and the addition task ARI-ADD, LIME−G, guided by the metadata of the expected continuation, strongly improves 2B accuracy, and LIME−A improves over Base by annotating its own output. †Base with constrained decoding, same prompts and items. ‡All Base outputs on ARI-ADD are numbers already, so the constraint changes none. Details in App. A.7.

size	x	Base	LIME	LIME−G
500M	≥ 2	69.68	73.95 ↑ 4.27	**80.41** ↑ 10.73
1B	≥ 2	73.32	77.48 ↑ 4.16	**83.43** ↑ 10.11
2B	≥ 2	75.51	79.24 ↑ 3.73	**84.95** ↑ 9.44
500M	1	33.91	34.59 ↑ 0.68	**55.10** ↑ 21.19
1B	1	37.23	37.89 ↑ 0.66	**58.35** ↑ 21.12
2B	1	39.08	39.80 ↑ 0.72	**59.94** ↑ 20.86

Task	Base	constr.†	LIME	LIME−G	LIME−A
FLQA-250	42.0	68.0	52.0 ↑ 10.0	**80.0** ↑ 38.0	58.0 ↑ 16.0
FLQA-500	49.5	65.5	65.3 ↑ 15.8	**73.5** ↑ 24.0	65.8 ↑ 16.3
FLQA-1000	34.8	51.3	47.0 ↑ 12.2	**65.3** ↑ 30.5	49.0 ↑ 14.2
FLQA-2000	40.3	44.5	44.3 ↑ 4.0	**52.5** ↑ 12.2	45.5 ↑ 5.2
FLQA-3000	28.2	37.0	30.0 ↑ 1.8	**39.3** ↑ 11.1	30.8 ↑ 2.6
ARI-ADD	22.6	22.6‡	26.9 ↑ 4.3	**58.7** ↑ 36.1	25.5 ↑ 2.9

at every context length. $\texttt{LIME-G}_{\texttt{500M}}$ even surpasses the larger, unguided $\texttt{Baseline}_{\texttt{2B}}$ at 250 to 1000 tokens of context (70.0 to 73.8 vs. 34.8 to 49.5).

On ARI-ADD, annotating the single digits with number metadata already improves accuracy by $+4.3$pp. Guiding the model to continue with digits at the ␣ token of the prompt raises accuracy by $+36.1$pp. Constraining Baseline to digits leaves every output unchanged, as all outputs are numbers already, whereas LIME−G guided by the digit class alone reaches 51.4 (App. A.7). Guidance changes what the model computes, not only what it emits.

4.6 LIME Models Learn to Self-Annotate

Guidance so far took the class from the task. We now ask whether a model can supply it itself. We introduce LIME with auxiliary heads (LIME−A), which predict the metadata in parallel with the generated token instead of taking it from an annotator or the task. We freeze a trained $\texttt{LIME}_{\texttt{2B}}$ model and, parallel to the language head, deploy two additional metadata heads that take the transformer's last hidden state as input to predict the metadata classes of the generated token. We train these heads on 16B more tokens, about 5% of the pre-training budget. Since a token's metadata enters with its embedding (Eq. 1), the heads must anticipate the class of a token not yet generated, which they do with 64.8% accuracy for POS (majority 13.2%) and 95.5% for NER (majority 88.8%, App. A.12).

Tab. 3 (right) shows that LIME−A achieves up to $+16.3$pp improvement on FLQA-500 and $+2.9$pp on ARI-ADD compared to Baseline. It matches or exceeds LIME at every FLenQA context length and stays within 1.4pp on ARI-ADD. The heads read the metadata from the hidden state (Popovic & Färber, 2024; Ghandeharioun et al., 2024), and $\texttt{LIME-A}_{\texttt{2B}}$ annotates its own output. On multiple-choice benchmarks, where every scored token carries the model's own annotation errors, LIME−A falls below Baseline (App. A.12).

Guidance on Demand. We also explore whether a model can take guidance when it is offered and run unguided otherwise, with a 500M switch model whose last hidden state is conditioned on a class input that may be left empty. Supplying the POS class of the token to be generated gains $+2.27$pp, leaving it empty stays within 0.3pp of Baseline, and the model's own predicted class adds nothing (App. A.13). The switch conditions the last hidden state only, whereas LIME−G's metadata enters with the input embedding.

4.7 Metadata Benefits Amplify Across Languages

The token-coupling analysis in Sec. 4.4 showed that metadata embeddings improve prediction on words fragmented into multiple tokens. Languages with higher tokenizer fertility (tokens per word) contain a larger share of such fragmented words, suggesting that LIME should be most beneficial where token fragmentation is most severe. To verify this, we pre-train Baseline and LIME (both 500M) on 10B tokens (Chinchilla-optimal) across five typologically diverse languages (EN, ES, DE, FI, KO) with Universal POS metadata (UPOS). Training data is sourced from FineWeb-Edu and FineWeb-2, using off-the-shelf spaCy models for UPOS annotation. With LIME, decoding accuracy increases by $+2.5$pp on average for all five languages (App. A.9). Gains scale with tokenizer

fertility: Korean (fertility 3.0) benefits most (+4.6pp), while English (1.1) gains least (+1.2pp), with a Pearson correlation of $r = 0.99$ between fertility and perplexity improvement. Bits per byte, which is independent of tokenization, drops by 2.9% for English and up to 9.4% for Korean, in the same order (App. A.9). Higher fertility means more word continuations, whose annotation anticipates the token to be generated (Sec. 4.4).

5 Discussion and Conclusion

In this work, we introduced LIME, a method to enrich token embeddings with linguistic metadata. This signal enhances token and entity cohesion and improves decoding accuracy by about 1pp from 500M to 7B. Given the metadata of the token to be generated, LIME-G outperforms constrained decoding in reasoning and arithmetic.

Decoding Accuracy and Scaling. The accuracy gain of LIME is stable throughout training, persists from 500M to 2B parameters and holds at 7B (App. A.11), since the metadata carries information about the token to be generated that the preceding text does not (Sec. 4.4). However, we find that these decoding accuracy gains do not consistently generalize to standard downstream predictive tasks, particularly those based on logit comparison, consistent with Tay et al. (2022) and Wettig et al. (2024). In generation they do carry over. At 2B, LIME gains on FLenQA and ARI-ADD, where no annotator sees the token to be generated (Sec. 4.5). For LIME-G, the gain over Baseline remains nearly constant across scale (+19.9 to +19.5pp).

Token Coupling and Multilingual Transfer. Linguistic metadata enhances coherence within subword boundaries and beyond single-word tokens (Sec. 4.4). By binding fragmented tokens into coherent units, metadata embeddings resolve subword inconsistencies, such as suffixes and digits. Metadata complements the tokenizer with word structure the token sequence lacks. Our multilingual experiments (Sec. 4.7) confirm that this mechanism transfers across typologically diverse languages, with gains scaling predictably with tokenizer fertility, suggesting LIME is particularly valuable for languages with high tokenizer fertility.

Metadata Steering and Internalization. Metadata-aligned decoding with LIME-G enables controllable generation, suggesting a promising alternative for tasks governed by deterministic schemas, such as structured data extraction and code generation, where traditional hard constraints can degrade reasoning performance. On ARI-ADD, guidance more than doubles accuracy (22.6 to 58.7), while a digit constraint leaves every output unchanged. Guidance acts at the input embedding and does not alter the transformer, but the model must be pre-trained with shifted metadata. The improvements of both LIME and LIME-G models on the addition task suggest that heterogeneity in the Baseline latent space hinders the early emergence of arithmetic capabilities. With LIME-A we demonstrate that metadata prediction can be internalized via lightweight auxiliary heads that predict the metadata of the generated token from the same latent representation as the language head (Sec. 4.6), reducing reliance on external annotation at inference time, while retaining gains over Baseline on FLenQA and ARI-ADD.

Limitations. While LIME-A annotates its own output, the prompt and the training data still require an annotator such as spaCy. Our approach generalizes to any token-level annotation with a discrete label space, but domains lacking a suitable tool would require developing one. LIME-A suits generative tasks, while on multiple-choice tasks its own annotation errors cost accuracy.

Future Work. Open directions include scaling beyond 7B with proportionally scaled data, additional metadata domains, such as the syntax of formal languages in code, mathematics and logic, and closing the gap between self-annotation and guidance with supplied metadata.

Two further directions follow from our results. First, the switch model shows that one model can take guidance when it is offered and run unguided otherwise. Training LIME-G with the class left empty part of the time could build the same switch into LIME-G itself. Second, in structured generation, a grammar or schema could supply the classes for LIME-G where constrained decoding is used today.

Ultimately, we demonstrated that linguistic metadata improves decoding because it anticipates the token to be generated, and that supplying it directly, as in LIME-G, enables guided generation.

AI USE STATEMENT

Generative AI tools were used during manuscript preparation to polish wording and presentation, and to create or edit software code. AI outputs were treated as drafts rather than authoritative sources. All AI-assisted text and code incorporated into the manuscript were reviewed by an author, and AI-assisted code was validated through testing and verification. Beyond the uses described above, generative AI was not used to develop theoretical or conceptual frameworks, formulate or prove mathematical claims, propose or refine research hypotheses, design the reported experiments, implement the research methods, clean or reformat the datasets, or create or modify scientific figures. The authors made the final decisions regarding the methodology, experiments, analyses, and wording, and take responsibility for all content, claims, code, and artifacts in this work.

ETHICS STATEMENT

This paper presents work whose goal is to advance the field of machine learning. `LIME` may inherit biases from its annotation tools: spaCy's accuracy degrades on non-standard text, dialects, and code-mixed input, which may carry over into pre-trained model weights. All models are trained on publicly available corpora (`DCLM-BASELINE`, FineWeb-Edu, FineWeb-2).

REPRODUCIBILITY STATEMENT

The architectures, hyperparameters and training setup are listed in App. A.1. The `LIME Tokenization` pipeline and its alignment rules are described in Sec. 3 and App. A.3. How metadata is obtained in each experiment is listed in App. A.10, and the evaluation protocols for downstream, reasoning and arithmetic tasks are given in App. A.5, A.7 and A.8. All annotations use publicly available spaCy models, and all training data is publicly available. The `LIME Tokenization` code is provided as supplementary material, and the training and evaluation code will be released.

REFERENCES

Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Jungo Kasai, David R. Mortensen, Noah A. Smith, and Yulia Tsvetkov. Do all languages cost the same? tokenization in the era of commercial language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, pp. 9904–9923, 2023.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.3, knowledge capacity scaling laws. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*. OpenReview.net, 2025.

Jordi Armengol-Estapé, Marta R. Costa-jussà, and Carlos Escolano. Enriching the transformer with linguistic factors for low-resource machine translation. In Galia Angelova, Maria Kunilovskaya, Ruslan Mitkov, and Ivelina Nikolova-Koleva (eds.), *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021), Held Online, 1-3 September, 2021*, pp. 73–78. INCOMA Ltd., 2021.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020*, pp. 7432–7439, 2020.

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. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020*, 2020.

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 *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Volume 1 (Long and Short Papers)*, pp. 2924–2936, 2019.

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. *CoRR*, abs/1803.05457, 2018.

Björn Deiseroth, Manuel Brack, Patrick Schramowski, Kristian Kersting, and Samuel Weinbach. T-FREE: subword tokenizer-free generative llms via sparse representations for memory-efficient embeddings. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024*, pp. 21829–21851, 2024.

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. Time-aware language models as temporal knowledge bases. *Trans. Assoc. Comput. Linguistics*, 10:257–273, 2022.

Takuro Fujii, Koki Shibata, Atsuki Yamaguchi, Terufumi Morishita, and Yasuhiro Sogawa. How do different tokenizers perform on downstream tasks in scriptio continua languages?: A case study in japanese. In *ACL (student)*, pp. 39–49. Association for Computational Linguistics, 2023.

Tianyu Gao, Alexander Wettig, Luxi He, Yihe Dong, Sadhika Malladi, and Danqi Chen. Metadata conditioning accelerates language model pre-training. In *Forty-second International Conference on Machine Learning, ICML 2025*, 2025.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, pp. 10932–10952, 2023.

Asma Ghandeharioun, Avi Caciularu, Adam Pearce, Lucas Dixon, and Mor Geva. Patchscopes: A unifying framework for inspecting hidden representations of language models. In *Forty-first International Conference on Machine Learning, ICML 2024*, 2024.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Retrieval augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, pp. 3929–3938, 2020.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *CoRR*, abs/2203.15556, 2022.

Jue Hou, Anisia Katinskaia, Anh-Duc Vu, and Roman Yangarber. Effects of sub-word segmentation on performance of transformer language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023*, pp. 7413–7425, 2023.

Linmei Hu, Zeyi Liu, Ziwang Zhao, Lei Hou, Liqiang Nie, and Juanzi Li. A survey of knowledge enhanced pre-trained language models. *IEEE Trans. Knowl. Data Eng.*, 36(4):1413–1430, 2024.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Volume 1: Long Papers*, pp. 1601–1611, 2017.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. CTRL: A conditional transformer language model for controllable generation. *CoRR*, abs/1909.05858, 2019.

Muhammad Khalifa, David Wadden, Emma Strubell, Honglak Lee, Lu Wang, Iz Beltagy, and Hao Peng. Source-aware training enables knowledge attribution in language models. *CoRR*, abs/2404.01019, 2024.

Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations*, pp. 66–71, 2018.

Yoav Levine, Barak Lenz, Or Dagan, Ori Ram, Dan Padnos, Or Sharir, Shai Shalev-Shwartz, Amnon Shashua, and Yoav Shoham. Sensebert: Driving some sense into BERT. In *ACL*, pp. 4656–4667. Association for Computational Linguistics, 2020.

Mosh Levy, Alon Jacoby, and Yoav Goldberg. Same task, more tokens: the impact of input length on the reasoning performance of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024*, pp. 15339–15353, 2024.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Kumar Guha, Sedrick Scott Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee F. Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah M. Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Raghavi Chandu, Thao Nguyen, Igor Vasiljevic, Sham M. 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, Alex 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 *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*, 2024.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. Multilingual denoising pre-training for neural machine translation. *Trans. Assoc. Comput. Linguistics*, 8:726–742, 2020.

Shayne Longpre, Gregory Yauney, Emily Reif, Katherine Lee, Adam Roberts, Barret Zoph, Denny Zhou, Jason Wei, Kevin Robinson, David Mimno, and Daphne Ippolito. A pretrainer's guide to training data: Measuring the effects of data age, domain coverage, quality, & toxicity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024*, pp. 3245–3276, 2024.

Sean McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, Jonas Geiping, Avi Schwarzschild, and Tom Goldstein. Transformers can do arithmetic with the right embeddings. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*, 2024.

Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, Pouya Tafti, Léonard Hussenot, Aakanksha Chowdhery, Adam Roberts, Aditya Barua, Alex Botev, Alex Castro-Ros, Ambrose Slone, Amélie Héliou, Andrea Tacchetti, Anna Bulanova, Antonia Paterson, Beth Tsai, Bobak Shahriari, Charline Le Lan, Christopher A. Choquette-Choo, Clément Crepy, Daniel Cer, Daphne Ippolito, David Reid, Elena Buchatskaya, Eric Ni, Eric Noland, Geng Yan, George Tucker, George-Cristian Muraru, Grigory Rozhdestvenskiy, Henryk Michalewski, Ian Tenney, Ivan Grishchenko, Jacob Austin, James Keeling, Jane Labanowski, Jean-Baptiste Lespiau, Jeff Stanway, Jenny Brennan, Jeremy Chen, Johan Ferret, Justin Chiu, and et al. Gemma: Open models based on gemini research and technology. *CoRR*, abs/2403.08295, 2024.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, 2018.

Benjamin Minixhofer, Edoardo Maria Ponti, and Ivan Vulic. Zero-shot tokenizer transfer. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*, 2024.

Pit Neitemeier, Björn Deiseroth, Constantin Eichenberg, and Lukas Balles. Hierarchical autoregressive transformers: Combining byte- and word-level processing for robust, adaptable language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*, 2025.

Scott Novotney, Sreeparna Mukherjee, Zeeshan Ahmed, and Andreas Stolcke. CUE vectors: Modular training of language models conditioned on diverse contextual signals. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 3368–3379, 2022.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Volume 1: Long Papers*, 2016.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. Grammar-aligned decoding. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*, 2024.

Guilherme Penedo, Hynek Kydlícek, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin A. Raffel, Leandro von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale. In *Advances in Neural Information Processing Systems 37: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024*, 2024.

Gabriel Poesia, Alex Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models. In *The Tenth International Conference on Learning Representations, ICLR 2022*, 2022.

Nicholas Popovic and Michael Färber. Embedded named entity recognition using probing classifiers. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024*, pp. 17830–17850, 2024.

Jack W. Rae, Sebastian Borgeaud, Trevor Cai, Katie Millican, Jordan Hoffmann, H. Francis Song, John Aslanides, Sarah Henderson, Roman Ring, Susannah Young, Eliza Rutherford, Tom Hennigan, Jacob Menick, Albin Cassirer, Richard Powell, George van den Driessche, Lisa Anne Hendricks, Maribeth Rauh, Po-Sen Huang, Amelia Glaese, Johannes Welbl, Sumanth Dathathri, Saffron Huang, Jonathan Uesato, John Mellor, Irina Higgins, Antonia Creswell, Nat McAleese, Amy Wu, Erich Elsen, Siddhant M. Jayakumar, Elena Buchatskaya, David Budden, Esme Sutherland, Karen Simonyan, Michela Paganini, Laurent Sifre, Lena Martens, Xiang Lorraine Li, Adhiguna Kuncoro, Aida Nematzadeh, Elena Gribovskaya, Domenic Donato, Angeliki Lazaridou, Arthur Mensch, Jean-Baptiste Lespiau, Maria Tsimpoukelli, Nikolai Grigorev, Doug Fritz, Thibault Sottiaux, Mantas Pajarskas, Toby Pohlen, Zhitao Gong, Daniel Toyama, Cyprien de Masson d'Autume, Yujia Li, Tayfun Terzi, Vladimir Mikulik, Igor Babuschkin, Aidan Clark, Diego de Las Casas, Aurelia Guy, Chris Jones, James Bradbury, Matthew J. Johnson, Blake A. Hechtman, Laura Weidinger, Iason Gabriel, William Isaac, Edward Lockhart, Simon Osindero, Laura Rimell, Chris Dyer, Oriol Vinyals, Kareem Ayoub, Jeff Stanway, Lorrayne Bennett, Demis Hassabis, Koray Kavukcuoglu, and Geoffrey Irving. Scaling language models: Methods, analysis & insights from training gopher. *CoRR*, abs/2112.11446, 2021.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020.

Melissa Roemmele, Cosmin Adrian Bejan, and Andrew S. Gordon. Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In *Logical Formalizations of Commonsense Reasoning, Papers from the 2011 AAAI Spring Symposium, Technical Report SS-11-06*, 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, 2021.

Maximilian Schall and Gerard de Melo. The hidden cost of structure: How constrained decoding affects language model performance. In *Proceedings of the 15th International Conference on Recent Advances in Natural Language Processing - Natural Language Processing in the Generative AI Era*, pp. 1074–1084, September 2025.

Craig W. Schmidt, Varshini Reddy, Haoran Zhang, Alec Alameddine, Omri Uzan, Yuval Pinter, and Chris Tanner. Tokenization is more than compression. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024*, pp. 678–702, 2024.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021*, pp. 9895–9901, 2021.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5B: an open large-scale dataset for training next generation image-text models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022*, 2022.

Rico Sennrich and Barry Haddow. Linguistic input features improve neural machine translation. In *Proceedings of the First Conference on Machine Translation, WMT 2016, colocated with ACL 2016*, pp. 83–91, 2016.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Volume 1: Long Papers*, 2016.

Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025*, pp. 2459–2475, 2025.

Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. Colake: Contextualized language and knowledge embedding. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020*, pp. 3660–3670, 2020.

Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, Weixin Liu, Zhihua Wu, Weibao Gong, Jianzhong Liang, Zhizhou Shang, Peng Sun, Wei Liu, Xuan Ouyang, Dianhai Yu, Hao Tian, Hua Wu, and Haifeng Wang. ERNIE 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *CoRR*, abs/2107.02137, 2021.

Yi Tay, Mostafa Dehghani, Jinfeng Rao, William Fedus, Samira Abnar, Hyung Won Chung, Sharan Narang, Dani Yogatama, Ashish Vaswani, and Donald Metzler. Scale efficiently: Insights from pretraining and finetuning transformers. In *The Tenth International Conference on Learning Representations, ICLR 2022*, 2022.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Position: Will we run out of data? limits of LLM scaling based on human-generated data. In *Forty-first International Conference on Machine Learning, ICML 2024*, 2024.

Alexander Wettig, Aatmik Gupta, Saumya Malik, and Danqi Chen. Qurating: Selecting high-quality data for training language models. In *Forty-first International Conference on Machine Learning, ICML 2024*, 2024.

Alexander Wettig, Kyle Lo, Sewon Min, Hannaneh Hajishirzi, Danqi Chen, and Luca Soldaini. Organize the web: Constructing domains enhances pre-training data curation. *CoRR*, abs/2502.10341, 2025.

Brandon T. Willard and Rémi Louf. Efficient guided generation for large language models. *CoRR*, abs/2307.09702, 2023.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. Data selection for language models via importance resampling. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*, 2023.

Fuzhao Xue, Yao Fu, Wangchunshu Zhou, Zangwei Zheng, and Yang You. To repeat or not to repeat: Insights from scaling LLM under token-crisis. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*, 2023.

Lili Yu, Daniel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. MEGABYTE: predicting million-byte sequences with multiscale transformers. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023*, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Volume 1: Long Papers*, pp. 4791–4800, 2019.

Shuai Zhang, Lijie Wang, Xinyan Xiao, and Hua Wu. Syntax-guided contrastive learning for pre-trained language model. In *Findings of the Association for Computational Linguistics: ACL 2022*, pp. 2430–2440, 2022.

A APPENDIX

The appendix provides training details, extended results, and additional analyses supplementing the main paper. App. A.1–A.4 cover the pre-training setup, hyperparameters, inference, tokenization impact, and training dynamics. App. A.5–A.8 present extended downstream benchmark results, token coupling tables, and detailed FLenQA and ARI-ADD results with prompt and completion examples. App. A.9 details multilingual pre-training results across five languages. App. A.10 describes how annotations are obtained in each experiment and reports the annotation-context runs. App. A.11–A.13 report the 7B models, the `LIME-A` details, and guidance as a switch. App. A.14–A.15 list the POS and NER annotator vocabularies.

A.1 LIME PRE-TRAINING SETUP DETAILS

Our 2B model architecture is derived from a Gemma-1 architecture with 2.43B parameters[5]. The 1B model architecture follows a Gemma-3-inspired design with 0.92B parameters[6]. Finally, the 500M architecture is based on a Gemma-3-style architecture with 0.50B trainable parameters. Tab. 4 presents a complete list of hyperparameter values. Exact parameter count can be found in Tab. 7. Training was conducted on 64 NVIDIA A100 GPUs (80GB each) across 8 nodes and required approximately 60 hours per model.

Table 4: Detailed list of hyperparameter values used for models and training.

Category	Hyperparameters	500M	1B	2B
Architecture	num_layers	12	26	18
	d_model	768	1024	2048
	mlp_factor	6	6	8
	num_heads	4	4	8
	num_kv_heads	1	1	1
	norm_type	RMS, $\epsilon = 1e\text{-}06$		
	vocab_size	256,000		
	position_embedding_type	rotary complex		
	rotary_embedding_base	10,000		
	activation_function	GELU		
	mlp_bias	no		
Optimization	loss_fn	Cross-Entropy		
	optimizer	AdamW		
	beta1, beta2, epsilon	[0.9, 0.95, 1.e-8]		
	learning_rate	3.e-3		
	lr_schedule	cosine decay to 3.e-4		
	warmup_steps	3,600 (5%)		
	weight_decay	0.033		
	gradient_clipping	no		
Stabilization	dropout	no		
	attention_dropout	no		
	embedding_dropout	no		
	embedding_grad_scaling	inverse mini_batch freq.		
	precision	bf16		
Training	global_batch_size	2048	2048	512
	sequence_length	2048	2048	8192
	micro_batch_size	4	4	1
	tokens_per_step	4,194,304		
	steps	72,000		
	packing_strategy	concatenation		

FineWeb-Edu Setup. The 7B models (App. A.11), the annotation-context runs (App. A.10), the switch model (App. A.13) and the multilingual models (App. A.9) use a second setup on FineWeb-Edu (Penedo et al., 2024) (Tab. 5). These experiments were added after the main runs, and each compares `Baseline` and `LIME` variants trained on identical data in identical order, which requires paired runs rather than the 302B-token budget; FineWeb-Edu and FineWeb-2 also provide the five

[5] https://huggingface.co/google/gemma-2b/blob/main/config.json

[6] https://huggingface.co/google/gemma-3-1b-pt/blob/main/config.json

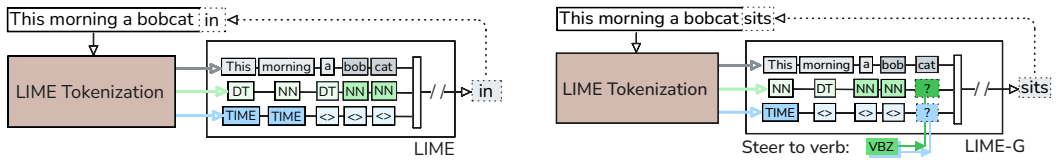
languages of App. A.9. At 7B the peak learning rate is 3e-4. The 7B pair, the annotation-context runs and the seed runs were trained on NVIDIA B200 GPUs.

Table 5: FineWeb-Edu setup. All models use RMSNorm, rotary embeddings (base 10,000), multi-query attention, GELU and the Gemma vocabulary of 256,000 tokens. The multilingual models use UPOS metadata, the switch model a POS class input only.

	7B (App. A.11)	500M (App. A.9, A.10, A.13)
`num_layers`, `d_model`	34, 3072	12, 768
`mlp_factor`, `num_heads`	8, 16	6, 4
parameters	7,388,958,720	495,864,576
metadata	POS and NER	POS and NER (see caption)
`sequence_length`, `global_batch_size`	8192, 512	2048, 2048
steps, tokens	11,920, 50B	2,384, 10B
`learning_rate` (cosine), warmup	3e-4 to 3e-5, 2,000	3e-3 to 3e-4, 300
`gradient_clipping`, precision	1.0, FP8	1.0, FP8

A.2 Inference with LIME and LIME-G

Fig. 4 illustrates inference. The prompt is tokenized and annotated by `LIME Tokenization`, and `LIME-G` attaches the metadata of the token to be generated to the last context token.



(a) `LIME` models receive metadata augmented input signal via `LIME Tokenization` for auto-regressive token prediction.

(b) `LIME-G` models can steer in metadata directions. At inference time, shifting metadata by one and injecting `VBZ` and `<>` for `␣cat` steers the model to generate a verb.

Figure 4: Inference with `LIME` and `LIME-G` models.

A.3 The impact of LIME Tokenization

We quantify the case where $n > m$, meaning that L produces more tokens from an input sequence than S (the Gemma tokenizer). By preserving the finer-grained pre-tokenization boundaries of L, `LIME Tokenization` may produce slightly more tokens. However, this tokenization maintains linguistically meaningful boundaries while reducing the effective vocabulary size by 1.03%. For instance, encoding 10,000 randomly selected `DCLM-BASELINE` samples results in a 1.19% increase. Tab. 6 lists the most frequent sequences in this dataset where `LIME Tokenization` introduces additional granularity.

Computational Cost. The computational overhead of `LIME Tokenization` is dominated by the inference cost of the spaCy word-classification models we employ. These models are lightweight and run efficiently on CPUs, achieving throughput of up to 10,000 words per second[7]. Assuming pre-tokenization and annotation of 302B pre-training tokens and a conservative estimate of 1.5 tokens per word, this corresponds to roughly 5,583 CPU hours. Assuming preprocessing were performed prior to training, it would require approximately 10.9 hours of wall time on 8 nodes with 64 AMD EPYC 7F52 cores each, ahead of our 60-hour pre-training run. However, our training was not data-pipeline-bound and therefore we executed labeling on-the-fly and fully distributed the workload across data workers, resulting in virtually no additional pre-training wall time.

[7] `https://spacy.io/usage/facts-figures#benchmarks-speed`

Table 6: Top 15 most frequent sequences where LIME Tokenization produces higher granularity when encoding 10,000 DCLM-BASELINE samples. Together, these sequences account for 54.18% of all cases with increased granularity.

Gemma Tokenizer	LIME Tokenization	%
▁▁	▁ ▁	8.67
▁▁▁.	▁ ▁.	6.53
▁don	▁do n	6.32
).	) .	6.07
),	) ,	4.17
.”	. ”	3.71
."	. "	3.64
▁didn	▁did n	2.47
▁doesn	▁does n	2.45
▁can	▁ca n	2.29
,"	, "	1.92
,”	, ”	1.73
▁isn	▁is n	1.59
▁cannot	▁can not	1.33
.)	.)	1.26

A.4 LIME Pre-training Result Details

LIME$_{1B}$ reached the final accuracy of Baseline with 44.63% fewer tokens, while LIME$_{2B}$ did so with 34.76% fewer (Fig. 5). As shown in Tab. 7, LIME-G consistently improves accuracy by 19.5 to 19.9pp across all model sizes, with perplexity reductions more pronounced at 500M (13.95) than at 2B (8.54). Furthermore, by applying LIME Tokenization, we observe accuracy and perplexity improvements (0.75pp accuracy, 0.6 perplexity on average) compared to the same model trained with standard Gemma BPE tokenization, measured on the respective token sequences. As model size increases, the proportion of learnable parameters introduced by LIME decreases, reaching as little as 0.006% for the 2B model.

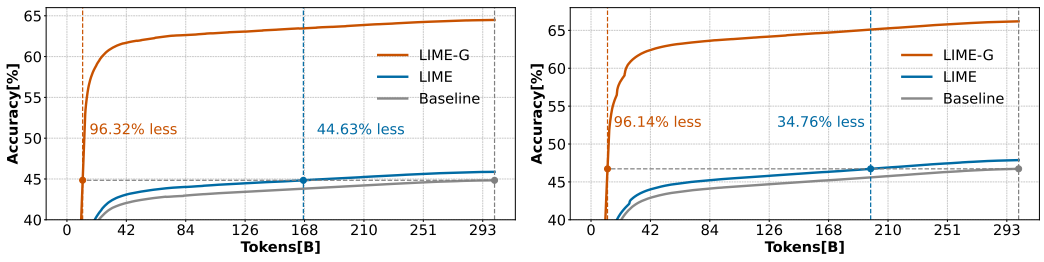


Figure 5: Decoding accuracy during pre-training. **Left**: 1B models. **Right**: 2B models. Dashed lines mark where LIME and LIME-G reach the final accuracy of Baseline.

A.5 Detailed Benchmark Results

Benchmark results across all model sizes are listed in Tab. 8. The nine benchmarks are ARC-Easy (Clark et al., 2018), BoolQ (Clark et al., 2019), COPA (Roemmele et al., 2011), HellaSwag (Zellers et al., 2019), LAMBADA (Paperno et al., 2016), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), TriviaQA (Joshi et al., 2017) and WinoGrande (Sakaguchi et al., 2021). All tasks are evaluated using a randomly selected 5-shot context on up to 1,000 samples, with standard errors reported. For multiple-choice tasks, context and answer choices are annotated jointly (App. A.10).

Table 7: Overview of pre-training metrics and parameter scaling across model sizes. Arrows: `Base` against Gemma BPE, `LIME` and `LIME-G` against `Base`.

Size	Model	Token Accuracy	Perplexity	Total Parameters	`LIME` Parameters
500M	`Gemma BPE`	41.35	20.37	495,864,576	
	`Baseline`	41.90 ↑ 0.55	19.72 ↓ 0.65	495,864,576	
	`LIME`	42.91 ↑ 1.01	18.50 ↓ 1.22	495,919,104	+54,528 +0.0110%
	`LIME-G`	61.83 ↑ 19.93	5.77 ↓ 13.95	495,919,104	+54,528 +0.0110%
1B	`Gemma BPE`	43.94	15.70	919,655,424	
	`Baseline`	45.20 ↑ 1.26	14.84 ↓ 0.86	919,655,424	
	`LIME`	46.21 ↑ 1.01	14.02 ↓ 0.82	919,728,128	+72,704 +0.0079%
	`LIME-G`	64.86 ↑ 19.66	4.73 ↓ 10.11	919,728,128	+72,704 +0.0079%
2B	`Gemma BPE`	46.48	13.11	2,426,480,640	
	`Baseline`	46.92 ↑ 0.44	12.82 ↓ 0.29	2,426,480,640	
	`LIME`	47.95 ↑ 1.03	12.10 ↓ 0.72	2,426,626,048	+145,408 +0.0060%
	`LIME-G`	66.40 ↑ 19.48	4.28 ↓ 8.54	2,426,626,048	+145,408 +0.0060%

Table 8: Detailed benchmark results across all model sizes. Improvements relative to the respective `Baseline` are indicated by ↑, relative to `LIME` by ⇑. Generative-format tasks are highlighted. `LIME-G` results use the metadata of the reference continuation.

	500M			1B			2B		
	`Base`	`LIME`	`LIME-G`	`Base`	`LIME`	`LIME-G`	`Base`	`LIME`	`LIME-G`
ARC-Easy	57.00±1.57	↑57.20±1.57	⇑58.10±1.56	67.30±1.48	↑68.40±1.47	⇑68.80±1.47	71.20±1.43	↑73.40±1.40	↑72.50±1.41
BoolQ	49.50±1.58	47.90±1.58	⇑59.40±1.55	51.60±1.58	51.60±1.58	⇑60.70±1.55	64.10±1.52	63.10±1.53	60.30±1.55
COPA	62.00±4.88	↑64.00±4.82	61.00±4.90	72.00±4.51	67.00±4.73	71.00±4.56	77.00±4.23	↑81.00±3.94	⇑83.00±3.78
HellaSwag	36.20±1.52	↑37.00±1.53	⇑43.10±1.57	52.90±1.58	↑53.20±1.58	⇑55.60±1.57	63.30±1.52	61.50±1.54	⇑64.40±1.51
LAMBADA	26.50±1.40	↑29.80±1.45	⇑49.00±1.58	41.00±1.56	↑46.00±1.58	⇑66.00±1.50	48.00±1.58	↑49.50±1.58	⇑73.40±1.40
OpenBookQA	31.00±2.07	↑32.60±2.10	⇑34.20±2.12	36.40±2.15	↑37.80±2.17	⇑39.20±2.19	39.80±2.19	↑42.00±2.21	⇑43.00±2.22
PIQA	69.90±1.45	69.40±1.46	68.20±1.47	72.60±1.41	↑73.20±1.40	71.20±1.43	74.10±1.39	↑75.10±1.37	73.70±1.39
TriviaQA	8.20±0.87	↑ 9.80±0.94	⇑19.50±1.25	17.90±1.21	↑19.70±1.26	⇑33.90±1.50	25.80±1.38	25.00±1.37	⇑42.20±1.56
WinoGrande	51.70±1.58	↑52.60±1.58	⇑54.60±1.58	57.40±1.56	↑57.60±1.56	⇑59.20±1.55	64.20±1.52	62.60±1.53	62.00±1.54
Mean	43.56±1.88	↑44.48±1.89	⇑49.68±1.95	52.12±1.90	↑52.72±1.90	⇑58.40±1.92	58.61±1.86	↑59.24±1.85	⇑63.83±1.82

A.6 Token Coupling Details

Fig. 6 illustrates that summing all positive and negative decoding accuracy changes from our qualitative analysis in Sec. 4.4 reproduces the overall $\texttt{LIME}_{\texttt{500M}}$ accuracy improvement of 1.01pp reported in Sec. 4.2.

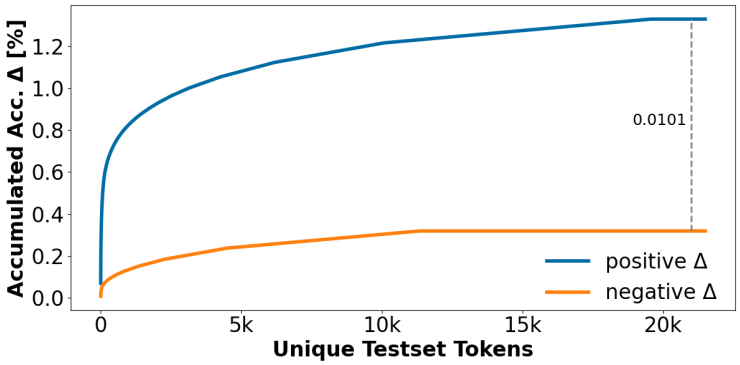


Figure 6: Accuracy shifts resemble the initially observed accuracy improvement of our qualitative analysis of $\texttt{LIME}_{\texttt{500M}}$ and `Baseline`.

We define words as sequences of x alphabetic character tokens ([A-Za-z]) enclosed by whitespaces. Additionally, we include hyphenated compound words, e.g. *long-term* and words bound by the apostrophe ' covering possessions and contractions, e.g., *Murphy's* and *We'll*. Tab. 9 provides a detailed view of how decoding accuracy varies for words of up to $x = 6$ tokens. Additionally, Tab. 10

presents accuracies calculated in the same way but *includes* the first token of each word. While the absolute values and improvements are smaller, the trend across word lengths remains consistent, and no outliers are observed. This suggests that the observed coupling improvements are not notably influenced by the model correctly predicting the first token of a word.

Table 9: Token accuracy within words of length x improves across all model sizes.

Model Size	x	n	Baseline	LIME	LIME-G
500M	2	164,256	69.63	76.34 ↑ 6.71	79.80 ↑ 10.17
500M	3	174,852	68.48	71.81 ↑ 3.33	80.84 ↑ 12.36
500M	4	61,947	75.36	76.94 ↑ 1.58	82.09 ↑ 6.73
500M	5	16,152	66.98	68.57 ↑ 1.59	80.18 ↑ 13.20
500M	6	2,840	55.95	56.65 ↑ 0.70	67.82 ↑ 11.87
1B	2	164,256	73.94	80.23 ↑ 6.29	83.31 ↑ 9.37
1B	3	174,852	71.87	75.13 ↑ 3.26	83.49 ↑ 11.62
1B	4	61,947	77.88	79.61 ↑ 1.73	84.68 ↑ 6.80
1B	5	16,152	70.38	72.91 ↑ 2.53	83.36 ↑ 12.98
1B	6	2,840	60.39	63.24 ↑ 2.85	71.97 ↑ 11.58
2B	2	164,256	76.46	82.24 ↑ 5.78	85.06 ↑ 8.60
2B	3	174,852	73.77	76.78 ↑ 3.01	84.72 ↑ 10.95
2B	4	61,947	79.92	81.00 ↑ 1.08	86.34 ↑ 6.42
2B	5	16,152	72.46	74.41 ↑ 1.95	84.73 ↑ 12.27
2B	6	2,840	64.61	66.20 ↑ 1.59	74.82 ↑ 10.21
500M	≥ 2	424,960	69.68	73.95 ↑ 4.27	80.41 ↑ 10.73
1B	≥ 2	424,960	73.32	77.48 ↑ 4.16	83.43 ↑ 10.11
2B	≥ 2	424,960	75.51	79.24 ↑ 3.73	84.95 ↑ 9.44
500M	1	5,049,553	33.91	34.59 ↑ 0.68	55.10 ↑ 21.19
1B	1	5,049,553	37.23	37.89 ↑ 0.66	58.35 ↑ 21.12
2B	1	5,049,553	39.08	39.80 ↑ 0.72	59.94 ↑ 20.86

Table 10: The trend of token-level accuracy improvements for words of length x persists even when the first token of each word is included.

Model Size	x	n	Baseline	LIME	LIME-G
500M	1	5,049,553	33.91	34.59 ↑ 0.68	55.10 ↑ 21.19
500M	2	328,512	43.28	46.86 ↑ 3.58	54.15 ↑ 10.87
500M	3	262,278	51.89	54.29 ↑ 2.40	66.23 ↑ 14.34
500M	4	82,596	60.97	62.32 ↑ 1.35	69.80 ↑ 8.83
500M	5	20,190	55.88	57.29 ↑ 1.41	69.36 ↑ 13.48
500M	6	3,408	49.32	50.15 ↑ 0.83	61.30 ↑ 11.98
1B	1	5,049,553	37.23	37.89 ↑ 0.66	58.35 ↑ 21.12
1B	2	328,512	47.35	50.79 ↑ 3.44	58.57 ↑ 11.22
1B	3	262,278	55.27	57.66 ↑ 2.39	69.67 ↑ 14.40
1B	4	82,596	63.67	65.14 ↑ 1.47	72.73 ↑ 9.06
1B	5	20,190	59.26	61.37 ↑ 2.11	72.97 ↑ 13.71
1B	6	3,408	53.58	55.96 ↑ 2.38	65.02 ↑ 11.44
2B	1	5,049,553	39.08	39.80 ↑ 0.72	59.94 ↑ 20.86
2B	2	328,512	49.79	53.00 ↑ 3.21	60.87 ↑ 11.08
2B	3	262,278	57.25	59.42 ↑ 2.17	71.21 ↑ 13.96
2B	4	82,596	65.68	66.63 ↑ 0.95	74.59 ↑ 8.91
2B	5	20,190	61.26	62.98 ↑ 1.72	74.57 ↑ 13.31
2B	6	3,408	57.28	58.74 ↑ 1.46	68.10 ↑ 10.82

A.7 Extensive FLenQA and ARI-ADD Results

Tab. 11 shows that the evaluated arithmetic capabilities begin to emerge at the 1B scale, with LIME providing slight improvements and LIME-G yielding a substantial increase of +12.9pp.

Tab. 12 reports FLenQA results for all model sizes and applicable noise lengths. Variations in spaces and dashes between two words are ignored when matching the expected property to avoid penalizing

Table 11: Complete ARI-ADD results.

Size	Baseline	LIME	LIME-G
2B	22.6	26.9 ↑ 4.3	58.7 ↑ 36.1
1B	4.1	5.2 ↑ 1.1	17.0 ↑ 12.9
500M	0.5	0.1 ↓ 0.4	1.3 ↑ 0.8

correctly predicted but differently formatted completions. LIME-G consistently improves reasoning performance across scales, with the 500M model showing the largest gain of 60.0pp on FLQA-1000.

Table 12: Detailed FLenQA results across model sizes. Accuracy is a match of the expected property within the first eight generated words for Baseline and LIME, and within the first two for LIME-G.

Size	Task	Baseline	LIME	LIME-G
2B	FLQA-250	42.0	52.0 ↑ 10.0	80.0 ↑ 38.0
2B	FLQA-500	49.5	65.3 ↑ 15.8	73.5 ↑ 24.0
2B	FLQA-1000	34.8	47.0 ↑ 12.2	65.3 ↑ 30.5
2B	FLQA-2000	40.3	44.3 ↑ 4.0	52.5 ↑ 12.2
2B	FLQA-3000	28.2	30.0 ↑ 1.8	39.3 ↑ 11.1
1B	FLQA-250	36.0	28.0 ↓ 8.0	80.0 ↑ 44.0
1B	FLQA-500	39.0	32.0 ↓ 7.0	74.0 ↑ 35.0
1B	FLQA-1000	32.4	33.0 ↑ 0.6	78.5 ↑ 46.1
500M	FLQA-250	22.0	40.0 ↑ 18.0	70.0 ↑ 48.0
500M	FLQA-500	22.0	48.8 ↑ 26.8	73.8 ↑ 51.8
500M	FLQA-1000	12.5	30.5 ↑ 18.0	72.5 ↑ 60.0

Additionally we present FLenQA results with a different prompt and stricter matching (Tab. 13). The prompt is shorter and less specific (*Ethan is in a*), and the matching criteria are more stringent (first-two words). Under these conditions, Baseline and LIME perform generally lower. LIME-G improvements across model sizes are also amplified, reaching +75.8pp for $\text{LIME-G}_{\text{500M}}$.

Table 13: Additional FLenQA results evaluated with matching within the first two words using the prompt *'is in a'*.

Size	Task	Baseline	LIME	LIME-G
2B	FLQA-250	12.0	22.0 ↑ 10.0	76.0 ↑ 64.0
2B	FLQA-500	12.5	15.3 ↑ 2.8	62.3 ↑ 49.8
2B	FLQA-1000	7.3	9.5 ↑ 2.2	63.8 ↑ 56.5
2B	FLQA-2000	5.0	7.0 ↑ 2.0	55.8 ↑ 50.8
2B	FLQA-3000	3.8	7.8 ↑ 4.0	52.5 ↑ 48.7
1B	FLQA-250	32.0	58.0 ↑ 26.0	98.0 ↑ 66.0
1B	FLQA-500	6.0	32.5 ↑ 26.5	78.0 ↑ 72.0
1B	FLQA-1000	4.3	24.8 ↑ 20.5	68.8 ↑ 64.5
500M	FLQA-250	10.0	14.0 ↑ 4.0	80.0 ↑ 70.0
500M	FLQA-500	3.5	4.0 ↑ 0.5	79.3 ↑ 75.8
500M	FLQA-1000	2.3	1.5 ↓ 0.8	68.8 ↑ 66.5

Comparison with Constrained Decoding. Tab. 14 compares guidance with constrained decoding under a class constraint, with identical prompts and items. Its Baseline results agree with Tab. 12 and Tab. 11 within two points. For FLenQA, constrained decoding masks the first generated token of Baseline to the tokens tagged as adjectives (JJ) in the training annotations and leaves later tokens free, so that multi-token properties can be completed. For ARI-ADD, every generated token is masked to digit tokens.

LIME-G is guided by the class the task requires (App. A.2). Reading its predictions in a single pass or generating token by token gives the same accuracy (73.8 vs. 73.3 on FLQA-500, 55.8 on ARI-ADD). Annotating the prompt on its own, so that the guidance carries only the digit class, yields 51.4 on ARI-ADD. Annotating it together with the expected result, as in Tab. 14, yields 55.8. Over

Table 14: Guided and constrained decoding at 2B under a class constraint, same setting.

	FLenQA, context length					
	250	500	1000	2000	3000	ARI-ADD
`Baseline`	42.0	47.8	34.5	41.3	29.0	21.8
`Baseline` + constrained decoding	68.0	65.5	51.3	44.5	37.0	21.8
`LIME-G`, class of the expected continuation	**80.0**	**73.8**	**65.5**	**51.3**	**38.8**	**55.8**

six prompt formulations and five context lengths, `LIME-G` improves over `Baseline` in 28 of 30 settings.

A.8 ARI-ADD AND FLENQA COMPLETIONS

We illustrate FLenQA and ARI-ADD evaluations using example prompts and completions from the 2B models (Tab. 15). In the two sampled tokens shown, the completions consist solely of digits, a pattern that holds for all completions across all model variants. The gains are correct results, not digit formatting.

Table 15: ARI-ADD prompt and completion examples generated by our 2B models.

Prompt	Completion	Model
The result is: 12+15 =	27	`Baseline`
	27	`LIME`
	27	`LIME-G`
The result is: 18+26 =	36	`Baseline`
	44	`LIME`
	44	`LIME-G`
The result is: 48+45 =	10	`Baseline`
	91	`LIME`
	93	`LIME-G`

Further, Tab. 16 contains prompt and completion examples from our FLenQA evaluation, specifically from the 2B model with 1,000 noise (FLQA-1000). The expected property is matched within the first eight words, while `LIME-G` is matched within the first two. For comparability, words such as *white walled* are counted as a single word, consistent with counting *marble-floored* as one word.

Table 16: FLQA-1000 prompt and completion examples generated by our 2B models.

Prompt	Completion	Model
Eric George is in a room. It has the following properties:	\n • It is a **white walled**. \n • It is	`Baseline`
	it is **white walled**, it is a room, and	`LIME`
	white walled	`LIME-G`
Kathleen Russo is in a room. It has the following properties:	\n - it is a room \n - it is a room	`Baseline`
	It is **marble-floored**.\n It is a foyer. It	`LIME`
	marble-floored	`LIME-G`
Rachel Hancock is in a room. It has the following properties:	it is a room, it is a room	`Baseline`
	it is a room, it is a room	`LIME`
	wooden-floored	`LIME-G`

A.9 MULTILINGUAL PRE-TRAINING DETAILS

We pre-train `Baseline` and `LIME` ($w_d = 1$, $+0.003\%$ parameters) at 500M parameters on 10B tokens (Chinchilla-optimal) equally split across five languages (English=EN, Spanish=ES, German=DE,

Finnish=FI, Korean=KO). Tab. 17 reports the results as per-language perplexity and decoding accuracy.

Table 17: Multilingual pre-training results for 500M models trained on 10B tokens (2B per language). LIME improves all five languages. Gains in perplexity scale with tokenizer fertility (Pearson $r = 0.99$).

Lang	Fertility	Base PPL	LIME PPL	Base Acc	LIME Acc
EN	1.11	34.1	30.8 ↓ 9.9%	36.5	37.7 ↑ 1.2
ES	1.22	30.1	26.8 ↓ 10.7%	36.4	37.8 ↑ 1.4
DE	1.44	32.1	27.7 ↓ 13.5%	38.6	40.5 ↑ 1.9
FI	2.14	16.6	13.8 ↓ 16.8%	46.1	49.6 ↑ 3.5
KO	3.00	12.8	10.1 ↓ 21.2%	46.6	51.2 ↑ 4.6
Mean	—	25.1	21.8 ↓ **13.1%**	40.8	43.4 ↑ **2.5**

Tab. 19 reports bits per byte (BPB), which does not depend on tokenization. LIME reduces BPB by 5.3% on average, with the largest improvement for Korean (-9.4%), confirming that the gains are not an artifact of increased sequence length.

Seeds. We repeat the multilingual pair with three seeds (Tab. 18). The average gain over all five languages is $+2.36 \pm 0.52$pp, the English gain $+1.09 \pm 0.35$pp, and the ordering of the gains by tokenizer fertility is identical for all three seeds. The Pearson correlation between fertility and perplexity improvement of $r = 0.99$ has a 95% confidence interval of $[0.84, 0.999]$ (five languages).

Table 18: Accuracy gain of LIME over Baseline (pp) at 500M for three seeds.

Seed	EN	ES	DE	FI	KO	Mean
42	+1.16	+1.44	+1.90	+3.49	+4.61	+2.52
1337	+1.39	+1.80	+2.43	+3.52	+4.75	+2.78
2024	+0.71	+0.81	+1.32	+2.51	+3.51	+1.77
Mean	$+1.09 \pm 0.35$					$+2.36 \pm 0.52$

Table 19: Bits per byte (BPB) for multilingual 500M models. BPB is length-invariant and confirms that LIME improvements are not an artifact of tokenization granularity differences.

Lang	Base BPB	LIME BPB
EN	1.091	1.059 ↓ 2.9%
ES	1.091	1.054 ↓ 3.3%
DE	1.152	1.104 ↓ 4.2%
FI	1.240	1.158 ↓ 6.6%
KO	1.110	1.006 ↓ 9.4%
Mean	1.137	1.076 ↓ **5.3%**

A.10 How Annotations Are Obtained

Tab. 20 lists for each experiment how the metadata is obtained.

Annotations computed on the full document reflect what follows a token. To locate the gain, we train 500M models in the FineWeb-Edu setup (App. A.1) on identical text and data order, differing only in how their annotations are obtained, and evaluate each with annotations obtained the same way. Restricted annotations come from a tagger distilled from spaCy that sees only the preceding tokens and the current token, or additionally exactly one following token (95 to 97% agreement with spaCy). Tab. 21 reports the results on a shared validation set of 20M tokens. With annotations from the full document, LIME improves accuracy by $+0.84$pp, close to the $+1.01$pp of Sec. 4.2 at a fraction of the budget. With annotations from the preceding text only, the gain vanishes, for both seeds. A single token of right context recovers it in full.

Table 20: How metadata is obtained in each experiment.

Experiment	Metadata
Pre-training, all models	spaCy on the full document
`LIME`, decoding accuracy (Sec. 4.2), multilingual (Sec. 4.7), 7B (App. A.11)	spaCy on the full document, as in training
`LIME`, the nine benchmarks incl. LAMBADA and TriviaQA (Sec. 4.3)	spaCy on context and candidate, annotated jointly
`LIME`, FLenQA and ARI-ADD (Sec. 4.5)	spaCy on the prompt; the text generated so far is re-annotated after each generated token
`LIME-G`, decoding accuracy and benchmarks	metadata of the token to be generated, obtained from the reference text
`LIME-G`, FLenQA (Sec. 4.5)	class of the expected property
`LIME-G`, ARI-ADD (Sec. 4.5)	class of the expected result, annotated together with the prompt (App. A.7)
`LIME-A` (Sec. 4.6)	spaCy on the prompt; generated tokens annotated by the model's own heads
Switch model (App. A.13)	class of the token to be generated, supplied part of the time

Table 21: Annotation-context runs, 500M, 10B tokens. Δ against the `Baseline` of the same seed; the two `Baseline` seeds differ by 0.07pp. Sampled: tags sampled from the tagger's posterior instead of the most likely tag.

Seed	Annotations	Loss	Accuracy	Δ Acc. (pp)
42	none (`Baseline`)	3.0357	41.48	–
42	full document	2.9851	42.32	+0.84
42	preceding text only	3.0435	41.39	−0.09
42	preceding text only, sampled	3.0478	41.34	−0.14
42	one token of right context	2.9829	42.38	+0.90
1337	none (`Baseline`)	3.0418	41.41	–
1337	preceding text only	3.0406	41.43	+0.01

Tab. 22 breaks the accuracy gain down by position type. Among position types, the gain is largest on entity continuations, and more than half of it lies on the 4.6% of positions where the restricted tagger disagrees with spaCy on the preceding token.

Table 22: Accuracy gain over `Baseline` (pp) by position type, seed 42.

Position type	Share	full document	preceding text only	one token of right context
token with leading space	73.5%	+0.59	−0.13	+0.64
token without leading space	26.4%	+1.55	+0.04	+1.61
entity continuation	9.6%	+2.98	−0.14	+2.88
digit	4.5%	+0.68	−0.14	+0.74
restricted tagger disagrees with spaCy	4.6%	+9.91	−0.95	+9.48
all	100%	+0.84	−0.09	+0.90

Without right context, spaCy agrees with its own full-document annotation on 92% of POS and 96% of NER tags. With one word of right context, agreement is 98% for both (503k positions). One word therefore settles almost all annotations.

A.11 7B MODELS

We train Baseline and LIME with 7B parameters on 50B tokens in the FineWeb-Edu setup (App. A.1). Tab. 23 reports decoding accuracy and perplexity on a held-out FineWeb-Edu set of 20M tokens and on 10,000 DCLM-BASELINE documents, and Tab. 24 the downstream benchmarks (5-shot, as in App. A.5). The accuracy gain is +1.24pp, and LIME improves 7 of 9 downstream tasks. BoolQ decreases, as at 500M and 2B, and both models are below the always-yes rate of 60.1%.

Table 23: 7B models, decoding accuracy and perplexity.

	FineWeb-Edu (held out)		DCLM-BASELINE	
	Accuracy	Perplexity	Accuracy	Perplexity
Baseline	49.39	10.86	41.96	23.72
LIME	**50.63**	**10.12**	**43.21**	**22.19**

Table 24: 7B models, downstream benchmarks (5-shot).

	ARC-E	BoolQ	COPA	HellaSw.	LAMBADA	OBQA	PIQA	TriviaQA	WinoGr.	Mean
Baseline	74.1	57.9	77.0	44.0	28.3	29.2	74.6	19.2	56.6	51.2
LIME	73.6	48.4	**81.0**	**44.3**	**31.1**	**29.6**	**75.1**	**23.7**	**58.4**	**51.7**

A.12 LIME-A DETAILS

The auxiliary heads of LIME-A predict the metadata of the token being generated from the last hidden state, while the model itself remains identical to LIME. Tab. 25 reports their accuracy on held-out text, with the preceding tokens annotated by spaCy on the full document. Tab. 26 reports the downstream benchmarks at 2B, with the context annotated by spaCy and the answer tokens annotated by the model's own heads, next to Baseline and LIME from Tab. 8. LIME-A matches or exceeds LIME on FLenQA and stays within 1.4pp on ARI-ADD (Sec. 4.6). On multiple-choice tasks, where every scored token carries its own annotation errors, it falls below Baseline. On LAMBADA and TriviaQA it is within one point of Baseline.

Table 25: Accuracy of the LIME-A heads for the metadata of the token being generated. The heads are trained on the frozen LIME model of each size.

	500M	1B	2B	majority class
POS	61.2	63.2	64.8	13.2
NER	95.1	95.2	95.5	88.8

Table 26: Downstream benchmarks at 2B for LIME-A, with Baseline and LIME from Tab. 8.

	ARC-E	BoolQ	COPA	HellaSw.	LAMBADA	OBQA	PIQA	TriviaQA	WinoGr.	Mean
Baseline	71.2	64.1	77.0	63.3	48.0	39.8	74.1	25.8	64.2	58.6
LIME	73.4	63.1	81.0	61.5	49.5	42.0	75.1	25.0	62.6	59.2
LIME-A	62.1	63.3	65.0	41.8	47.8	36.6	65.8	25.2	55.9	51.5

A.13 GUIDANCE AS A SWITCH

The switch model is a 500M model in the FineWeb-Edu setup. A linear head on the last hidden state predicts the POS class of the token to be generated, and token prediction is conditioned on a class input through a feature-wise modulation of the last hidden state and a low-rank logit bias. During training, the class input is the true class part of the time, empty part of the time, and the model's own prediction otherwise. Tab. 27 compares class inputs on the same checkpoint and positions. Supplying the class increases accuracy by $+2.27$pp, while the model's own prediction (60.3% accurate) adds nothing. Without the input, the model is within 0.3pp of `Baseline` on the full validation set.

Table 27: Switch model, decoding accuracy for different class inputs on the same positions.

Class input	Accuracy	Δ (pp)
none	41.19	–
the model's own prediction	41.10	-0.09
class of the model's own most likely token	41.19	0.00
class of the token to be generated, supplied	**43.46**	$\mathbf{+2.27}$

A.14 METADATA ANNOTATOR VOCABULARY: POS (ENGLISH)

For POS metadata embeddings, we use the spaCy `en_core_web_sm` model containing a lightweight CPU-based POS tagger that returns 50 tags (Tab. 28). `LIME Tokenization` augments its vocabulary with one additional special token.

Table 28: Glossary of 51 POS tags with their respective descriptions.

.	punctuation mark, sentence closer
,	punctuation mark, comma
-LRB-	left round bracket
-RRB-	right round bracket
“	opening quotation mark
”	closing quotation mark
:	punctuation mark, colon or ellipsis
$	symbol, currency
AFX	affix
CC	conjunction, coordinating
CD	cardinal number
DT	determiner
EX	existential there
FW	foreign word
HYPH	punctuation mark, hyphen
IN	conjunction, subordinating or preposition
JJ	adjective (English), other noun-modifier (Chinese)
JJR	adjective, comparative
JJS	adjective, superlative
LS	list item marker
MD	verb, modal auxiliary
NN	noun, singular or mass
NNP	noun, proper singular
NNPS	noun, proper plural
NNS	noun, plural
PDT	predeterminer
POS	possessive ending
PRP	pronoun, personal
PRP$	pronoun, possessive
RB	adverb
RBR	adverb, comparative
RBS	adverb, superlative
RP	adverb, particle
TO	infinitival “to”
UH	interjection
VB	verb, base form
VBD	verb, past tense
VBG	verb, gerund or present participle
VBN	verb, past participle
VBP	verb, non-3rd person singular present
VBZ	verb, 3rd person singular present
WDT	wh-determiner
WP	wh-pronoun, personal
WP$	wh-pronoun, possessive
WRB	wh-adverb
SP	space (English), sentence-final particle (Chinese)
ADD	email
NFP	superfluous punctuation
XX	unknown
_SP	whitespace
SPECIAL	special token

A.15 METADATA ANNOTATOR VOCABULARY: NER

For NER metadata embeddings, we employ the spaCy `en_core_web_sm` model, using its lightweight CPU-based NER tagger. The tagger distinguishes 18 entity types and the absence of an entity. With the special token added by `LIME Tokenization`, this gives 20 tags (Tab. 29).

Table 29: Glossary of 20 NER tags with their respective descriptions.

PERSON	People, including fictional
NORP	Nationalities or religious or political groups
FAC	Buildings, airports, highways, bridges, etc.
ORG	Companies, agencies, institutions, etc.
GPE	Countries, cities, states
LOC	Non-GPE locations, mountain ranges, bodies of water
PRODUCT	Objects, vehicles, foods, etc. (not services)
EVENT	Named hurricanes, battles, wars, sports events, etc.
WORK_OF_ART	Titles of books, songs, etc.
LAW	Named documents made into laws
LANGUAGE	Any named language
DATE	Absolute or relative dates or periods
TIME	Times smaller than a day
PERCENT	Percentage, including "%"
MONEY	Monetary values, including unit
QUANTITY	Measurements, as of weight or distance
ORDINAL	“first”, “second”, etc.
CARDINAL	Numerals that do not fall under another type
<>	No entity
SPECIAL	special token