LIME: MAKING LLM DATA MORE EFFICIENT WITH LINGUISTIC METADATA EMBEDDINGS

Anonymous authors

Paper under double-blind review

ABSTRACT

Pre-training decoder-only language models relies on vast amounts of high-quality data, yet the availability of such data is increasingly reaching its limits. While metadata is commonly used to create and curate these datasets, its potential as a direct training signal remains under-explored. We challenge this status quo and propose LIME (**Linguistic Metadata Embeddings**), a method that enriches token embeddings with metadata capturing syntax, semantics, and contextual properties. LIME substantially improves pre-training efficiency. Specifically, it adapts up to 56% faster to the training data distribution, while introducing only 0.01% additional parameters at negligible compute overhead. Beyond efficiency, LIME improves tokenization, leading to remarkably stronger language modeling capabilities and generative task performance. These benefits persist across model scales (500M to 2B). In addition, we develop a variant with shifted metadata, LIME⁺¹, that can guide token generation. Given prior metadata for the next token, LIME⁺¹ improves reasoning performance by up to 38% and arithmetic accuracy by up to 35%.

1 Introduction

Autoregressive language models have emerged as a prominent area of research due to their impressive capabilities. However, training these 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, models up to 2B parameters require the same—or even increased—amount of training data, as their language modeling performance tends to regress (Hoffmann et al., 2022).

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 of improving the utility of existing datasets. To compensate this shortcoming, methods of accumulating LLM pre-training datasets shift from mere quality filtering to synthetization 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 indicate 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 pre-trained subword tokenizer with linguistically informed annotations, namely POS and 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, with \mathtt{LIME}^{+1} we shift the metadata embeddings by one to guide the generation with look-ahead metadata.

In our experiments, we demonstrate that LIME substantially enhances language modeling performance. By incorporating metadata, LIME improves data efficiency during training, enabling models to adapt up to 56% faster to the training data distribution. Additionally, LIME mitigates issues caused by artificial tokenization splits, keeping the meaning of subword tokens together, as supported by both qualitative and quantitative analyses. Finally, with LIME⁺¹ we demonstrate that models achieve up to 35% higher accuracy when the metadata class of the token to be predicted is revealed in advance. Crucially, we apply this guidance in tasks such as for reasoning and arithmetic, where the relevant metadata is naturally available rather than artificially constructed. These benefits of metadata annotations persist consistently throughout our scaling ablations of 500M, 1B and 2B parameter models. Our results raise important questions about inefficient pre-training data usage in standard causal language model training.

Our main contributions and findings are summarized as follows:

- 1. We introduce LIME and LIME⁺¹, our approach to augment token embeddings with linguistic metadata in Sec. 3.
- 2. We demonstrate how LIME improves language modeling capabilities, in particular next-token prediction, consistently across various model sizes (Sec. 4.2).
- 3. LIME models excel in generative downstream tasks (Sec. 4.3) and keep split word tokens together by improving natural language word cohesion (Sec. 4.4).
- 4. LIME⁺¹ enables inference-time metadata steering which improves reasoning and arithmetic capabilities (Sec. 4.5).

2 Related Work

Before introducing LIME, we outline key areas of prior work that motivate and inform our approach. Specifically, we review tokenization strategies and their implications for model efficiency, the role of metadata in LLM pre-training, the use of linguistic annotations as auxiliary supervision, and recent efforts to integrate metadata directly at the embedding level.

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. Recent work explores byte-level or tokenizer-free models such as ByT5 (Xue et al., 2023) and MegaByte (Yu et al., 2023), and T-FREE (Deiseroth et al., 2024), which embeds words via character trigrams, capturing morphological overlaps with smaller embeddings.

Metadata in LLM Pre-training. Pre-training refers to an LLM learning from scratch on large corpora to establish a foundation for downstream adaptation. 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, CTRL (Keskar et al., 2019) prepends source-domain metadata, Dhingra et al. (2022) prepend timestamps to improve memorization, Liu et al. (2020) add language identifiers for multilingual training, and Khalifa et al. (2024) include document identifiers to improve source attribution. Most recently, Allen-Zhu & Li (2025) demonstrated that prepending a special token to useful data significantly increases the model's capacity ratio,

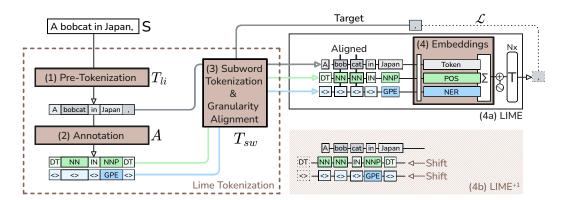


Figure 1: LIME and LIME⁺¹ architecture. (1) Input text S is split by the linguistic tokenizer (T_{li}) . (2) Linguistic splits are annotated, e.g. with \overline{POS} and \overline{NER} tags. (3) Subword tokenization (T_{sw}) is applied to the linguistic tokens and annotations are aligned to the new splits. (4) Tokens and metadata are embedded, fused together and passed into consecutive transformer blocks.

while Gao et al. (2025) provided empirical evidence by showing a 33% improvement in pre-training efficiency when URL metadata was prepended. 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: ERNIE (Sun et al., 2021) leverages entity-aware masking guided using NER, while syntax-aware pre-training models incorporate dependency structures into attention mechanisms (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.

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, (Guu et al., 2020) propose retrieval-augmented pre-training using a knowledge-enriched encoder. The joint learning of language and knowledge embeddings within a masked language modeling framework has been investigated by 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. Yet, enriching token embeddings with fine-grained linguistic metadata remains an open frontier. We address this gap with LIME, showing that it delivers scalable improvements in both efficiency and performance.

3 LIME: LINGUISTIC METADATA FOR LLMS

In this section, we present LIME which integrates linguistic metadata for LLMs. LIME consists of four stages: (1) linguistic pre-tokenization, (2) metadata annotation, (3) subword tokenization & granularity alignment, (4) and metadata embedding, as shown in Fig. 1.

Illustrative Example. Before introducing technical details, we summarize $\mathtt{LIME}(^{+1})$. The method enriches input text with linguistic metadata (e.g., POS or NER tags) before passing it to an LLM. As illustrated in Fig. 1, a sentence is first split into linguistic tokens (e.g., A bobcat in Japan \rightarrow four tokens), which are then annotated with metadata such as POS tags. Since LLMs operate on subword units, the metadata is aligned with the subword tokenizer output to preserve dimensions. The tokens and metadata are then embedded, combined, and fed into the transformer. LIME applies each token's own metadata, while LIME⁺¹ shifts embeddings to use the next token's metadata, giving the model richer linguistic context. These stages are further illustrated in Tab. 1.

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

(0)	Input Sequence		A_bobcat_in_Japan					
(1) (2)	Linguistic Pre-Tokenization Metadata Annotation	A DT			_in IN	_Japan 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 ⁺¹	{ A } +{ NN }	{ _bob } +{ NN }	{ cat } +{ IN }	{ _in } +{ NNP }	{ _Japan } +{ X }		

- (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 T_{sw} , we introduce a rule-based linguistic tokenizer T_{li} for pre-tokenization. Introducing T_{li} 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 $a\in C$ and C being the set of pre-defined annotation symbols. The annotation function g can consist of rule-based methods, heuristics or classification models. As illustrated in Fig. 1, our method allows to define and integrate multiple annotators, e.g., POS and NER annotations.
- (3) Subword Tokenization & Granularity Alignment. Given the subsequent subword tokenizer $T_{sw}=(f_{sw},V_{sw})$, its vocabulary and annotation function will naturally differ from those of the rule-based word tokenizer T_{li} used in the first stage, as shown in Fig. 1. This entails that $n_{li}=|T_{li}(s)|\neq|T_{sw}(s)|=n_{sw}$ has to be aligned for an input text s. In the case of $n_{li}< n_{sw}$, we define the annotation function $g':V^{n_{li}}\to C^{n_{sw}}$ that resolves this granularity mismatch as follows: g' tracks for every token in $T_{sw}(s)$ its word context defined by $T_{li}(s)$ and repeats the relevant annotation until a granularity match is achieved. In the case of $n_{li}>n_{sw}$, where T_{li} splits a input sequence in more tokens than T_{sw} , we keep the finer granularity of T_{li} by expressing every token in $T_{li}(s)$ with tokens from the vocabulary V_{sw} . For example, the word "don't" could linguistically be split as $T_{li}(s)=\text{don}$ 't while the subword tokenizer may produce $T_{sw}=\text{don}$ 't. In that case we keep the slightly less compressed word boundaries of $T_{li}(s)$ and their corresponding annotations as the subsequent tokenization of T_{sw} , 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 consisting of tuple $(t_i,(a_{i,d})_{d\in D})$ 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 one-hot encoded to a vector of length $|V_{sw}|$ and multiplied with the embedding matrix $W^L \in \mathbb{R}^{|V_{sw}| \times h}$, h being the hidden size hyperparameter, producing the language token embedding $E_L(t_i)$. 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:

$$E(t_i) = E_L(t_i) + \sum_{d \in D} w_d E_M^d(a_{i,d}) . \tag{1}$$

Note that the annotation embedding process is applied to individual respective matrices $W^{(d)} \in \mathbb{R}^{|C_d| \times h}$ producing $E^d_M(a_{i,d})$. E_L 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⁺¹, is tailored for scenarios where the metadata annotation of the next token (a_{i+1}) is known in advance, such as demonstrated in Sec. 4.5. During training and inference (Fig. 4), the base embedding E_L is augmented using the metadata

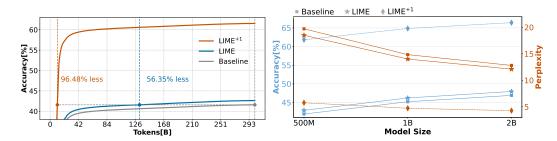


Figure 2: **Left:** Next-token accuracy improves with metadata embedding layers. Our LIME_{500M} model requires 56% less pre-training data to achieve the same token prediction accuracy as Baseline. **Right:** Accuracy and perplexity improvements translate consistently across model sizes.

embeddings of the *next* token, rather than the current one. This look-ahead embedding is defined as:

$$E(t_i) = E_L(t_i) + \sum_{d \in D} w_d E_M^d(a_{i+1,d}) .$$
 (2)

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 now present empirical evidence demonstrating the benefits of LIME and LIME⁺¹. We start with the experimental setup followed by an investigation of model prediction qualities (Sec. 4.2) and evaluations on popular benchmarks (Sec. 4.3). We then provide qualitative and quantitative analyses that LIME models improve token coupling (Sec. 4.4) and finally show that LIME⁺¹ excels in reasoning as well as arithmetic performance (Sec. 4.5).

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.2). We applied the three-stage LIME Tokenization process described in Sec. 3, with T_{li} being the english spaCy tokenizer and T_{sw} the SentencePiece (Kudo & Richardson, 2018) Gemma tokenizer, and otherwise optimizing the standard cross-entropy loss $\mathcal L$ on the language head (disregarding metadata). As mentioned in Sec. 3, extending tokenization with T_{li} increases token count, in our case by 1.19%, but compresses the used vocabulary by 1.03%, and as such frees embedding parameters (cf. App. A.3). Models trained with Lime Tokenization but no additional metadata embedding layers are referred to as Baseline in the following.

We extended the Baseline models with two metadata domains available in spaCy: Syntactic Part of Speech (POS), with $|C_{\rm pos}|=51$ (App. A.9), and semantic Named Entity Recognition (NER), with $|C_{\rm ner}|=20$ (App. A.10). Both annotation embedding layers are weighted equally with $w_{\rm pos}=w_{\rm ner}=1$ (Eq. 1). For both LIME and LIME⁺¹, the additional embedding layers, having 71 entries in total, add less than 0.01% to the total parameters. Since it requires only one vectorized lookup and addition in the forward pass, it adds negligible runtime overhead. LIME(+1) inference is illustrated in Fig. 4.

4.2 LINGUISTIC METADATA-EMBEDDINGS IMPROVE LANGUAGE MODELING

We first analyze the effect of metadata embeddings on general pre-training dynamics. Across all three model sizes, LIME variants reach the same next-token accuracy as their baselines earlier in training. Fig. 2 (left) illustrates these gains for the 500M model. First, we observe that LIME_{500M}

https://spacy.io/api/tokenizer

Table 2: LIME excels at generative tasks. Improvements to Base are indicated by \uparrow , to LIME with \Uparrow . We highlight (yellow) generative-format tasks. Exemplified on 500M, other model sizes in App. A.5.

	Base _{500M}	LIME _{500M}	LIME ⁺¹
ARC-Easy [6]	57.00 ± 1.57	\uparrow 57.20±1.57	↑58.10±1.56
BoolQ [5]	49.50 ± 1.58	47.90 ± 1.58	↑59.40±1.55
COPA [34]	62.00 ± 4.88	↑64.00±4.82	61.00 ± 4.90
HellaSwag [50]	36.20 ± 1.52	↑37.00±1.53	↑43.10±1.57
LAMBADA [29]	26.50 ± 1.40	↑29.80±1.45	↑49.00±1.58
OpenBookQA [25]	31.00 ± 2.07	↑32.60±2.10	↑34.20±2.12
PIQA [3]	69.90 ± 1.45	69.40 ± 1.46	68.20±1.47
TriviaQA [15]	8.20±0.87	↑ 9.80±0.94	↑19.50±1.25
WinoGrande [35]	51.70±1.58	↑52.60±1.58	↑54.60±1.58
Mean	$43.56{\scriptstyle\pm1.88}$	↑44.48±1.89	↑49.68±1.95

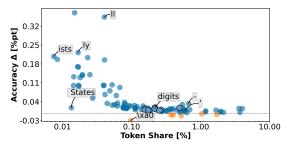


Figure 3: LIME_{500M} token prediction accuracy increases within semantic and syntactic metadata class boundaries: Among the 100 most impactful (share $\times \Delta$) tokens, our model exhibits improved token coupling for suffix-, digit-tokens and an exemplary entity group token _States.

achieves the final accuracy of its Baseline counterpart (41.90%) using 56.35% fewer tokens. Specifically, both models follow the expected pre-training pattern of rapid early gains. For 500M, gains begin plateauing after about 42B tokens. From that stage onward, LIME maintains a stable accuracy advantage, suggesting that extended pre-training of the baseline cannot substitute for LIME metadata embeddings. This training dynamic is consistent across all three model sizes.

Further, we evaluate model quality across sizes on a DCLM-BASELINE test set of 10,000 samples, reporting next-token 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. Accuracy increases by a constant magnitude of approximately +1 percentage points across all model sizes. Perplexity reduction is, as expected in this range, more pronounced in smaller model sizes. LIME⁺¹ shows a substantial improvement in accuracy and perplexity. Relative to Baseline models, accuracy increases by over 40% across model sizes, while perplexity is reduced by more than 65%, enabling our LIME⁺¹_{2B} model to achieve a perplexity of 4.30. Training was stable across all methods and model sizes. For more detailed results and learning curves, please refer to App. A.4. Next, we demonstrate that these results directly translate to downstream task performance.

4.3 LIME MODELS EXCEL IN GENERATIVE TASKS

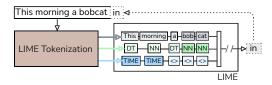
We evaluate downstream transfer on standard benchmarks (Tab. 2), highlighting results for the 500M model; comparable trends hold for all sizes, detailed in App. A.5. Each task was run with 1,000 samples using randomly selected 5-shot contexts.

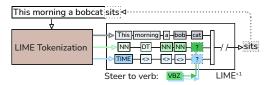
The results show that LIME models on average slightly improve over their respective Baseline models. On tasks that are evaluated using generative greedy sampling (e.g., LAMBADA/TriviaQA), our method improves model performance substantially across sizes. LIME⁺¹ models, on the other hand, improve in 500M and 1B on 7 of 9 tasks, and on 5 tasks in 2B parameters, when being compared to their respective Baseline and LIME. The substantial and constant improvement of LIME models on greedily sampled tasks is further amplified by LIME⁺¹ models: For TriviaQA, we observe a relative improvement of 138%, 90%, and 64%, and in LAMBADA of 85%, 61%, 53% for 500M, 1B, 2B, compared to their respective Baseline.

4.4 LIME KEEPS TOKENS TOGETHER

Building on the previous finding that LIME boosts performance in generative tasks, we aim to better understand where and how these improvements arise at the token level. To this end, we first conduct a qualitative study of token-level accuracy shifts, followed by a detailed quantitative analysis across word lengths, model sizes, and architectures.

²The remaining tasks are multiple-choice log-likelihood evaluations compared to argmax matching.





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

(b) LIME⁺¹ models can steer in metadata directions. Shifting metadata by one and injecting VBZ and <> for __cat steers the model to generate a verb token next.

Figure 4: Inference with LIME and LIME⁺¹ models.

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 prediction accuracy and weight them 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, that finish words consisting of multiple tokens, exhibit large positive accuracy shifts. This holds moreover for apostrophe and hyphen, 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. Accumulating all positive and negative accuracy shifts resembles the total accuracy gain of LIME_{500M} as reported in Sec. 4.2 (more details in App. A.6).

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 $x \ge 2$.⁴ For singleword tokens (x = 1), we still obtain gains of roughly 0.7%. This highlights that metadata embeddings not only have a strong effect on the cohesion of subword sequences, but also help improving incontext relevance. LIME⁺¹ models further amplify the previously described improvements. As expected, the look-ahead metadata of LIME⁺¹ compellingly improve single-word token prediction by +21%. Moreover, models trained that way even further double the token-coupling accuracies improvements to +10%. These findings underline to what extent certainty on token metadata can still improve next-token prediction, even on models with up to 2B parameters. More details in App. A.6.

4.5 METADATA GUIDANCE UNLOCKS HIDDEN REASONING AND ARITHMETIC ABILITIES

Finally, we demonstrate the impact of token-metadata on two common tasks, reasoning (FLenQA) and symbolic addition arithmetic (ARI-ADD).

Task Definition. FLenQA (Levy et al., 2024) requires reasoning across multiple, increasingly noisy contexts. As such it provides a measure of a model's ability to robustly handle reasoning scenarios. We use the FLenQA task 'People in Rooms' (PIR) that prompts to combine 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 ground truth "blue walled" to appear in the subsequent words. Note that using LIME½B, we syntactically steer (Fig. 4b) the model to, e.g., in the prior example, predict an adjective token (JJ and ✓) by overloading the last token (a) of the context. The benchmark is grouped into 5 noise levels, adding irrelevant information of 250, 500, 1000, 2000, and 3000 tokens, and each group consisting of 400 samples.

Second, for the arithmetic task, we again try to leverage the already obtained arithmetic performance from DCLM-BASELINE without further finetuning. We found the prompt "The result is: {number}+{number} = " performs 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 $5 \le \{number\} \le 49$ and average the results over 500 unique number pairs.

³Details are found in App. A.6.

⁴Note that, the category of tokens in words of $x \ge 2$ accounts for 10% of the number of single-word tokens, since we apply an English-optimized tokenizer to a primarily English corpus.

Table 3: **Left:** LIME models have improved prediction accuracies on tokens within natural language words of length $x \ge 2$ (in tokens) and slightly improve on words consisting of one token (x=1). n denotes the total token count of the respective length. **Right:** Reasoning and arithmetic capabilities are significantly improved with LIME $_{2B}^{+1}$. Evaluated on FLenQA and ARI-ADD, an addition task. More details in App. A.7.

size	x	n	Base	LIME	LIME ⁺¹	Task	Base _{2B}	$\texttt{LIME}_{\texttt{2B}}$	LIME _{2B}
500M	≥2	424,960	69.68	73.95 ↑ 4.27	80.41 ↑ 10.73	FLQA-250	42.0	52.0 ↑ 10.0	80.0 ↑ 38.0
1B	≥2	424,960	73.32	$77.48 \uparrow 4.16$	83.43 ↑ 10.11	FLQA-500	49.5	$65.3 \uparrow 15.8$	73.5 ↑ 24.0
2B	≥2	424,960	75.51	79.24 ↑ 3.73	84.95 ↑ 9.44	FLQA-1000	34.8	$47.0 \uparrow 12.2$	65.3 ↑ 30.5
500M	1	5,049,553	33.91	34.59 ↑ 0.68	55.10 ↑ 21.19	FLQA-2000	40.3	44.3 ↑ 4.0	52.5 ↑ 12.2
1B	1	5,049,553	37.23	$37.89 \uparrow 0.65$	58.35 ↑ 21.12	FLQA-3000	28.2	30.0 ↑ 1.8	39.3 ↑ 11.1
2B	1	5,049,553	39.08	39.80 ± 0.72	59.94 ↑ 20.86	ARI-ADD	22.6	26.9 ↑ 4.3	58.7 ↑ 36.1

Results. In Tab. 3 (right), we report performances across the different FLenQA variants and the ARI-ADD task. For all tasks we only show the 2B accuracies, as the other model sizes further degenerated. Nevertheless, the described behavior is consistent across sizes and found in App. A.7 for completion.

In FLenQA we first observe a clear trend of decreasing performance with increasing noise context across all models, as it is expected. LIME already yields notable improvements over Baseline, especially for shorter contexts (e.g., +15.8 on FLQA-500). This again indicates that enriching the embedding with metadata is already helpful to improve next-token accuracy. In contrast, LIME⁺¹ consistently delivers two-digit improvements across all variants, still achieving 39.3% on FLQA-3000 and improving even 38% on FLQA-250. This trend highlights the effectiveness of syntactic steering in reasoning scenarios when the target class is clear. It is moreover beneficial as noise mitigation, maintaining robust performance even in more challenging settings.

For ARI-ADD, we first want to highlight that all generated outputs, across all models, were numbers. The drop in accuracy therefore refers to generation of the wrong digits. Simply annotating the single digits with number metadata already improves accuracy by 4.3%. Through syntactic steering, i.e. prioring the model to continue with digits at the __token of the prompt, proves crucial in unlocking the full potential of the model on the arithmetic task and improves accuracy by 36.1%.

5 DISCUSSION

Building on the promising results presented above, we now discuss key observations and potential directions for future work with LIME models.

Pre-Training Efficiency. LIME models consistently reach baseline token accuracy and perplexity with substantially fewer training tokens. This suggests that linguistic metadata provides meaningful information that default embeddings would otherwise need to learn expensively, and moreover do not converge to for 302 billion tokens. The observed benefits scale to larger model sizes; we conducted experiments with models of up to 2B parameters. LIME metadata embeddings reshape the inductive biases of LLMs. Whereas standard approaches compress linguistic and metadata signals into a single embedding space, LIME keeps them disentangled at the embedding mapping, offering weak supervision. Furthermore, we find that the language modeling improvements of LIME do not consistently generalize to standard downstream predictive tasks, i.p. those based on logit comparison, which is consistent with the observations of Tay et al. (2022) and Wettig et al. (2024). Providing the LLM with accurate look-ahead metadata, as in LIME⁺¹, acts as a constraint on the search space for the next token, leading to substantial improvements in accuracy and perplexity across model sizes.

Token Coupling. Linguistic metadata helps models maintain coherence both within subword boundaries and beyond single-word tokens (Sec. 4.4). Our analyses show that LIME strengthens token coupling across words of all lengths and improves predictions for digits and entities. Using lookahead metadata, as with LIME⁺¹, further amplifies accuracy gains. By binding fragmented tokens into coherent units, metadata embeddings not only boost accuracy but also resolve inconsistencies inherent to subword tokenization, such as handling prefixes, suffixes, and numbers, which even renders reasoning abilities more robust and precise. These results highlight metadata as a powerful

 inductive bias and a potential complement to tokenizer architectures, providing structural guidance that would otherwise be missing.

Tokenizer's Language Bias. Tokenization inherently introduces language-specific biases. Due to the fixed vocabulary size and the challenges of modifying embedding layers after training, underrepresented languages often undergo suboptimal segmentation. Such segmentations frequently misalign with semantic boundaries, which can negatively impact overall model performance. Lime Tokenization provides a simple yet effective way to extract a richer data signal per token by leveraging lightweight annotation models. As previously stated this has in particular been beneficial to keep split tokens together. Future work should explore augmenting the embedding space with explicit language identifiers to further improve multilingual robustness. Additionally, the tokenizer-agnostic nature of our method opens the door to extending it beyond subword-based tokenizers, including byte-level, character-level, or even tokenizer-free models (Deiseroth et al., 2024).

Metadata Steering. Metadata inference-time steering with LIME⁺¹ enables controllable generation. In our generative use cases (Sec. 4.5), metadata guidance leads to substantial improvements in reasoning and arithmetics. This indicates that metadata is not only a meaningful training signal but also a useful mechanism at inference, providing a new, interpretable interface for controllable token generation. Unlike fine-tuning methods or steering vector methods, metadata steering operates directly at the embedding layer, requiring no retraining or fine-tuning. The improvements of both LIME and LIME⁺¹ models on the addition task suggest that heterogeneity in the Baseline latent space hinders the early emergence of arithmetic capabilities. While this study focuses on linguistic metadata, our approach can be readily adapted to other domains. In some cases, non-linguistic domains may provide even more informative look-ahead metadata for language modeling. This should particularly be useful in the current research trend towards smaller agentic experts.

Predicting Metadata. Throughout this work we applied spaCy as a metadata annotator for POS and NER tags. Being a model itself, it achieves an accuracy of roughly 97% on POS and F-score of 86% on NER. Albeit not being perfect, we demonstrated consistently improved performance when applying these annotations during training. LIME⁺¹ leverages look-ahead metadata to achieve even stronger performance. Metadata steering, however, relies on knowledge of the next token. We demonstrated common use cases where this information is naturally available, allowing LIME⁺¹ to capitalize on these gains. However, in tasks where such information is absent, either additional supervision is required, or the model must learn to predict the appropriate next-token metadata itself. To show feasibility of the latter, we extended the language modeling head of a LIME⁺¹_{2B} model with an additional metadata head tasked to predict look-ahead POS tags. After pre-training with a balanced loss on both heads, the metadata head achieves a top-3 accuracy of 82.32%, without affecting the language head's performance. Prior work has also explored prediction using intermediate internal representations of large language models (Popovic & Färber, 2024; Ghandeharioun et al., 2024). This demonstrates that simultaneous autoregressive prediction of look-ahead metadata holds strong potential, and, by further steering with it as shown with LIME⁺¹, leverages gains in token accuracy.

6 Conclusion

In this work, we introduced LIME, a novel method to overload token embeddings with linguistic metadata capturing syntax, semantics, and contextual information. Our approach demonstrates improvements in language modeling capabilities across various model sizes, while requiring minimal additional computational resources. We show that LIME models keep split word tokens together and improve cohesion of entities spanning tokens. Our method shows significant improvements on generative tasks. Furthermore, by pre-training with look-ahead metadata embeddings in LIME $^{+1}$, we show that token generation can effectively be steered, which is particularly beneficial for noisy reasoning and arithmetic tasks. These results highlight that LIME is a seamless way of integrating metadata as an auxiliary data signal, enhancing both model efficiency and controllability.

Future work could explore alternative methods for generating the look-ahead metadata used in LIME⁺¹, such as recovering it from the internal states of LLMs, to further enhance autoregressive capabilities. Additionally, evaluating LIME 's compatibility with different tokenization strategies and extending the approach beyond language modeling could unlock broader applications and capabilities.

7 REPRODUCIBILITY

We are committed to ensuring the reproducibility of our work. Upon publication, we will release the code, the final pre-trained models, and detailed instructions necessary to reproduce all final experiments and results presented in this paper.

Meanwhile we have the modified python tokenizer class, the core of this research, attached. Furthermore, we report all hyperparameters in App. A.2.

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. URL https://doi.org/10.18653/v1/2023.emnlp-main.614.
- 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. URL https://openreview.net/forum?id=FxNNiUgtfa.
- 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. URL https://doi.org/10.1609/aaai.v34i05.6239.
- 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. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.
- 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. URL https://doi.org/10.18653/v1/n19-1300.
- Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018. URL http://arxiv.org/abs/1803.05457.
- 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. URL https://doi.org/10.18653/v1/2024.emnlp-main.1217.
- 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. URL https://doi.org/10.1162/tacl_a_00459.
- Tianyu Gao, Alexander Wettig, Luxi He, Yihe Dong, Sadhika Malladi, and Danqi Chen. Metadata conditioning accelerates language model pre-training, 2025. URL https://arxiv.org/abs/2501.01956.

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. URL https://openreview.net/forum?id=5uwBzcn885.

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. URL http://proceedings.mlr.press/v119/guu20a.html.

- 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. URL https://doi.org/10.48550/arXiv.2203.15556.
- 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. URL https://doi.org/10.18653/v1/2023.emnlp-main.459.
- 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. URL https://doi.org/10.1109/TKDE.2023.3310002.
- 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. URL https://doi.org/10.18653/v1/P17-1147.
- 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. URL http://arxiv.org/abs/1909.05858.
- 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. URL https://doi.org/10.48550/arXiv.2404.01019.
- 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. URL https://doi.org/10.18653/v1/d18-2012.
- 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. URL https://doi.org/10.18653/v1/2024.acl-long.818.
- 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* 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, 2024. URL http://papers.nips.cc/paper_files/paper/

2024/hash/19e4ea30dded58259665db375885e412-Abstract-Datasets_and_Benchmarks_Track.html.

- 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. URL https://doi.org/10.1162/tacl_a 00343.
- 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. URL https://doi.org/10.18653/v1/2024.naacl-long.179.
- 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 38: Annual Conference on Neural Information Processing Systems 2024*, *NeurIPS 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/c35986bc1ee29b31c1011481b77fe540-Abstract-Conference.html.
- 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. URL https://doi.org/10.48550/arXiv.2403.08295.
- 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. URL https://doi.org/10.18653/v1/d18-1260.
- Benjamin Minixhofer, Edoardo Maria Ponti, and Ivan Vulic. Zero-shot tokenizer transfer. In Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/532ce4fcf853023c4cf2ac38cbc5d002-Abstract-Conference.html.
- 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. URL https://openreview.net/forum?id=tU074jg2vS.
- 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. URL https://doi.org/10.18653/v1/2022.findings-acl.265.
- 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. URL https://doi.org/10.18653/v1/p16-1144.

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 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/370df50ccfdf8bde18f8f9c2d9151bda-Abstract-Datasets_and_Benchmarks_Track.html.

- 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. URL https://doi.org/10.18653/v1/2024.emnlp-main.988.
- 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. URL https://arxiv.org/abs/2112.11446.
- 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. URL https://jmlr.org/papers/v21/20-074.html.
- 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. URL http://www.aaai.org/ocs/index.php/SSS/SSS11/paper/view/2418.
- 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. URL https://doi.org/10.1145/3474381.
- 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. URL https://doi.org/10.18653/v1/2024.emnlp-main.40.
- 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. URL http://papers.nips.cc/paper_files/paper/2022/hash/a1859debfb3b59d094f3504d5ebb6c25-Abstract-Datasets_and_Benchmarks.html.
- 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. URL https://doi.org/10.18653/v1/w16-2209.

- 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. URL https://doi.org/10.18653/v1/p16-1162.
- 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. URL https://aclanthology.org/2025.acl-long.123/.
- 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. URL https://doi.org/10.18653/v1/2020.coling-main.327.
- 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. URL https://arxiv.org/abs/2107.02137.
- 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. URL https://openreview.net/forum?id=f20YVDyfIB.
- 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. URL https://openreview.net/forum?id=ViZcqDQjyG.
- 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. URL https://openreview.net/forum?id=GLGYYqPwjy.
- 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. URL https://doi.org/10.48550/arXiv.2502.10341.
- 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. URL http://papers.nips.cc/paper_files/paper/2023/hash/6b9aa8f418bde2840d5f4ab7a02f663b-Abstract-Conference.html.
- 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. URL http://papers.nips.cc/paper_files/paper/2023/hash/b9e472cd579c83e2f6aa3459f46aac28-Abstract-Conference.html.
- 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. URL http://papers.nips.cc/paper_files/paper/2023/hash/f8f78f8043f35890181a824e53a57134-Abstract-Conference.html.
- 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. URL https://doi.org/10.18653/v1/p19-1472.

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. URL https://doi.org/10.18653/v1/2022.findings-acl. 191.

A APPENDIX

 The appendix includes training curves and numerical results from pre-training, along with the hyperparameters used for both the model architectures and training procedures. We further show the impact of LIME tokenization, and present extended benchmark results corresponding to Sec. 4.3. Sec. 4.4 is supplemented with comprehensive token accuracy tables, while Sec. 4.5 is supplemented with detailed FLenQA and ARI-ADD results together with examples of prompts and completions for each task. Finally, we provide the metadata annotator vocabularies employed throughout our experiments.

A.1 LLM USAGE

We used LLMs to aid and polish writing our paper. The ideation, methodological design, and execution of experiments were solely the responsibility of the authors.

A.2 LIME PRE-TRAINING SETUP DETAILS

Our 2B model architecture is derived from a Gemma-1 architecture with 2.43B parameters⁵. The 1B model architecture follows a Gemma-3-inspired design with 0.92B parameters⁶. 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. 6. 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
	num_layers	12	26	18
	d_model	768	1024	2048
	mlp_factor	6	6	8
	num_heads	4	4	8
Architecture	num_kv_heads	1	1	1
	norm_type	RM	$S, \epsilon = 1$	e-06
	vocab_size		256,000	
	position_embedding_type	rot	ary comp	olex
	rotary_embedding_base		10,000	
	activation_function		GELU	
	mlp_bias		no	
	loss_fn	Cr	oss-Entro	ору
	optimizer		AdamW	,
	beta1, beta2, epsilon	[0.9	0.95, 1	.e-8]
Optimization	learning_rate	3.e-3		
-	lr_schedule	cosine	e decay to	o 3.e-4
	warmup_steps	3	,600 (5%	6)
	weight_decay		0.033	
	gradient_clipping		no	
	dropout		no	
	attention_dropout		no	
Stabilization	embedding_dropout		no	
Stabilization	embedding_grad_scaling	inverse	mini_ba	tch freq.
	precision		bf16	
	global_batch_size	2048	2048	512
	sequence_length	2048	2048	8192
	micro_batch_size	4	4	1
Training	tokens_per_step	4	4,194,30	4
	steps	72,000		
	packing_strategy	co	ncatenat	ion

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

 $^{^6}$ https://huggingface.co/google/gemma-3-1b-pt/blob/main/config.json

A.3 THE IMPACT OF LIME TOKENIZATION

We quantify the case where $n_{li} > n_{sw}$, meaning that T_{li} produces more tokens from an input sequence than T_{sw} (the Gemma tokenizer). By preserving the finer-grained pre-tokenization boundaries of T_{li} , 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 1,000 randomly selected DCLM-BASELINE samples (12.77M tokens) results in a 1.19% increase (12.90M tokens). Tab. 5 lists the most frequent sequences in this dataset where Lime Tokenization introduces additional granularity.

Table 5: Top 15 most frequent sequences where Lime Tokenization produces higher granularity when encoding 1,000 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	don	6.32
) .) -	6.07
),) ,	4.17
. "	. "	3.71
. "	. "	3.64
didn	didn	2.47
doesn	doesn	2.45
can	can	2.29
, "	, "	1.92
, "	, "	1.73
isn	isn	1.59
cannot	cannot	1.33
.)	.)	1.26

A.4 LIME PRE-TRAINING RESULT DETAILS

Pre-training proceeded smoothly, with the loss decreasing consistently and gradient norms remaining stable. $LIME_{1B}$ required 44.64% fewer tokens to fit the training data distribution, while $LIME_{2B}$ required 34.76% fewer tokens (Fig. 5). As shown in Tab. 6 , $LIME^{+1}$ consistently improves accuracy by approximately 19.50% across all model sizes, with perplexity reductions more pronounced at 500M (13.95%) than at 2B (8.54%). As model size increases, the proportion of learnable parameters introduced by LIME decreases, reaching as little as 0.006% for the 2B model.

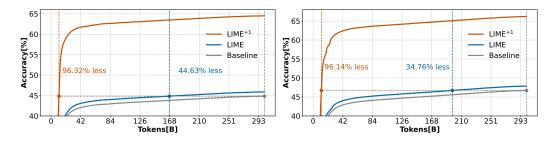


Figure 5: Next token prediction accuracy during Pre-training. Left: 1B models Right: 2B models.

Table 6: Overview of pre-training metrics and parameter scaling across model sizes.

Size	Model	Token Accuracy	Perplexity	Total Parameters	LIME Parameters
500M	Baseline LIME LIME ⁺¹	$41.90 \\ 42.91 \uparrow 1.01 \\ 61.83 \uparrow 19.93$	$19.72 \\ 18.50 \downarrow 1.22 \\ 5.77 \downarrow 13.95$	495,864,576 495,919,104 495,919,104	+54,528 +0,0110% +54,528 +0,0110%
1B	Baseline LIME LIME ⁺¹	45.20 46.21 ↑ 1.01 64.86 ↑ 19.65	$14.84 \\ 14.02 \downarrow 0.82 \\ 4.73 \downarrow 10.11$	919,655,424 919,728,128 919,728,128	+72,704 +0,0079% +72,704 +0,0079%
2B	Baseline LIME LIME ⁺¹	46.92 47.95 ↑ 1.03 66.40 ↑ 19.48	$12.82 \\ 12.10 \downarrow 0.72 \\ 4.28 \downarrow 8.54$	2,426,480,640 2,426,626,048 2,426,626,048	+145,408 +0,0060% +145,408 +0,0060%

A.5 DETAILED BENCHMARK RESULTS

Benchmark results across all model sizes are listed in Tab. 7. All tasks are evaluated using a randomly selected 5-shot context on 1,000 samples, with standard errors reported.

Table 7: Detailed benchmark results across all model sizes. Improvements relative to the respective Baseline are indicated by \uparrow , relative to LIME with \uparrow . Generative-format tasks are highlighted.

		500M			1B			2B	
	Base	LIME	LIME ⁺¹	Base	LIME	LIME ⁺¹	Base	LIME	LIME ⁺¹
ARC-Easy [6]	57.00±1.57 ↑	57.20±1.57	158.10±1.56	67.30±1.48 ↑	68.40±1.47	11.47 ↑68.80±1.47	71.20±1.43 ↑	73.40 ± 1.40	↑ 72.50±1.41
BoolQ [5]	49.50 ± 1.58	$47.90{\scriptstyle\pm1.58}$	↑59.40±1.55	51.60 ± 1.58	$51.60{\scriptstyle\pm1.58}$	1.55 ↑60.70±1.55	64.10±1.52	63.10 ± 1.53	60.30±1.55
COPA [34]	62.00±4.88 ↑	$64.00^{\pm 4.82}$	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 [50]	36.20±1.52 ↑	$37.00{\scriptstyle\pm1.53}$	↑43.10±1.57	52.90±1.58 ↑	53.20 ± 1.58	1.57 ↑55.60±1.57	63.30±1.52	ighthalpha 61.50 \pm 1.54	↑64.40±1.51
LAMBADA [29]	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 [25]	31.00±2.07 ↑	32.60±2.10	↑34.20±2.12	36.40±2.15 ↑	37.80±2.17	1 139.20±2.19	39.80±2.19 ↑	42.00±2.21	↑43.00±2.22
PIQA [3]	69.90±1.45	09.40 \pm 1.46	68.20±1.47	72.60±1.41 ↑	ighthalpha 1.40 73.20	71.20±1.43	74.10±1.39 ↑	75.10 ± 1.37	73.70 ± 1.39
TriviaQA [15]	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 [35]	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	1 149.68±1.95	52.12±1.90 ↑	52.71±1.90	158.40±1.92	53.52±2.19 ↑	53.95±2.18	↑63.83±1.82

A.6 TOKEN COUPLING DETAILS

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

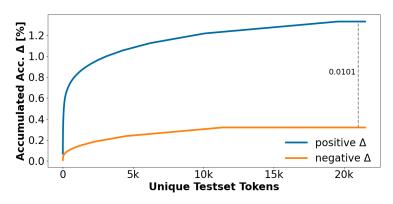


Figure 6: Accuracy shifts resemble the initially observed accuracy improvement of our qualitative analysis of LIME $_{500M}$ and Baseline.

In our experiments, we define natural language 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. 8 provides a detailed view of how token prediction accuracy varies for words of up to x=6 tokens. Additionally, Tab. 9 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 8: Token accuracy within words of length x improves across all model sizes.

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

Table 9: 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 ⁺¹
500M	1	5,049,553	33.91	$34.59 \uparrow 0.68$	55.10 ↑ 21.19
500M	2	328,512	43.28	$46.86 \uparrow 3.58$	54.15 ↑ 10.87
500M	3	262,278	51.89	$54.29 \uparrow 2.40$	66.23 ↑ 14.34
500M	4	82,596	60.97	$62.32 \uparrow 1.35$	69.80 ↑ 8.83
500M	5	20,190	55.88	57.29 ↑ 1.41	69.36 ↑ 13.48
500M	6	3,408	49.32	$50.15 \uparrow 0.83$	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	$50.79 \uparrow 3.44$	58.57 ↑ 11.22
1B	3	262,278	55.27	57.66 ↑ 2.39	69.67 ↑ 14.40
1B	4	82,596	63.67	$65.14 \uparrow 1.47$	72.73 ↑ 9.06
1B	5	20,190	59.26	$61.37 \uparrow 2.11$	72.97 ↑ 13.71
1B	6	3,408	53.58	$55.96 \uparrow 2.38$	65.02 ↑ 11.44
2B	1	5,049,553	39.08	$39.80 \uparrow 0.72$	59.94 ↑ 20.86
2B	2	328,512	49.79	$53.00 \uparrow 3.21$	60.87 ↑ 11.08
2B	3	262,278	57.25	$59.42 \uparrow 2.17$	71.21 ↑ 13.96
2B	4	82,596	65.68	$66.63 \uparrow 0.95$	74.59 ↑ 8.91
2B	5	20,190	61.26	$62.98 \uparrow 1.72$	74.57 ↑ 13.31
2B	6	3,408	57.28	$58.74 \uparrow 1.46$	68.10 ↑ 10.83

A.7 EXTENSIVE FLENQA AND ARI-ADD RESULTS

Here, we present the complete results for reasoning and arithmetic tasks across all model sizes. Tab. 10 shows that the evaluated arithmetic capabilities begin to emerge at the 1B scale, with LIME providing slight improvements and LIME⁺¹ yielding a substantial increase of 12.9%.

Table 10: Complete ARI-ADD results.

Size	Baseline	LIME	LIME ⁺¹
2B	22.6	$26.9 \uparrow 4.3$	58.7 ↑ 36.1
1B	4.1	5.2 ↑ 1.1	$17.0 \uparrow 12.9$
500M	0.5	$0.1 \downarrow 0.4$	1.3 ↑ 0.8

Tab. 11 reports FLenQA results for all model sizes and applicable noise lengths. Variations in spaces and dashes between two words are ignored when matching the ground truth to avoid penalizing correctly predicted but differently formatted completions. LIME⁺¹ consistently improves reasoning performance across scales, with the 500M model showing the largest gain of 51.8% on FLQA-500.

Table 11: Detailed FLenQA results across model sizes. While Baseline and LIME match the first eight generated words, LIME⁺¹ generates and matches only one word.

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

Additionally we present FLenQA results with a different prompt and stricter matching (Tab. 12). The prompt is shorter and less specific (*Ethan is in a*), and the matching criteria are more stringent (first-two words). Under these conditions, LIME_{1B} shows substantial improvement, while Baseline and LIME perform generally lower. LIME⁺¹ improvements across model sizes are also amplified, reaching +75.8% for LIME⁺¹_{500M}.

Table 12: Additional FLenQA results evaluated with strict next-two-words matching using the prompt 'is in a'.

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

A.8 ARI-ADD AND FLENQA COMPLETIONS

We illustrate FLenQA and ARI-ADD evaluations using example prompts and completions from the 2B models (Tab. 13). In the two sampled tokens shown, the completions consist solely of digits, a pattern that holds for all completions across all model variants. This demonstrates that LIME and LIME⁺¹ enhance arithmetic capabilities rather than just predicting digits.

Table 13: ARI-ADD prompt and completion examples generated by our 2B models.

Prompt	Completion	Model
The result is: 12+15 =	27 27 27	Baseline LIME LIME ⁺¹
The result is: 18+26 =	36 44 44	Baseline LIME LIME ⁺¹
The result is: $48+45 =$	10 91 93	Baseline LIME LIME ⁺¹

Further, Tab. 14 contains prompt and completion examples from our FLenQA evaluation, specifically from the 2B model with 1,000 noise (FLQA-1000). Ground truth is matched within the first eight words, while LIME⁺¹ uses only one. For comparability, words such as *white walled* are counted as a single word, consistent with counting *marble-floored* as one word.

Table 14: 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 it is white walled, it is a room, and white walled	Baseline LIME LIME ⁺¹
Kathleen Russo is in a room. It has the following properties:	\n - it is a room \n - it is a room It is marble-floored.\n It is a foyer. It marble-floored	Baseline LIME LIME ⁺¹
Rachel Hancock is in a room. It has the following properties:	it is a room, it is a room it is a room, it is a room wooden-floored	Baseline LIME LIME ⁺¹

A.9 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. 15). Lime Tokenization augments its vocabulary with one additional special token.

Table 15: Glossary of 51 POS tags with their respective descriptions.

1170		, , ,
1141		punctuation mark, sentence closer
1142	•	punctuation mark, sentence closer punctuation mark, comma
1143	-LRB-	left round bracket
1144	-RRB-	right round bracket
1145	-KKD- "	opening quotation mark
1146	"	closing quotation mark
1147		punctuation mark, colon or ellipsis
1148	\$	symbol, currency
1149	AFX	affix
	CC	conjunction, coordinating
1150	CD	cardinal number
1151	DT	determiner
1152	EX	existential there
1153	FW	foreign word
1154	HYPH	punctuation mark, hyphen
1155	IN	conjunction, subordinating or preposition
1156	JJ	adjective (English), other noun-modifier (Chinese)
1157	JJR	adjective (English), other noun mounter (Enmese)
1158	JJS	adjective, superlative
1159	LS	list item marker
1160	MD	verb, modal auxiliary
1161	NN	noun, singular or mass
1162	NNP	noun, proper singular
	NNPS	noun, proper plural
1163	NNS	noun, plural
1164	PDT	predeterminer
1165	POS	possessive ending
1166	PRP	pronoun, personal
1167	PRP\$	pronoun, possessive
1168	RB	adverb
1169	RBR	adverb, comparative
1170	RBS	adverb, superlative
1171	RP	adverb, particle
1172	TO	infinitival "to"
1173	UH	interjection
1174	VB	verb, base form
1175	VBD	verb, past tense
1176	VBG	verb, gerund or present participle
1177	VBN	verb, past participle
	VBP	verb, non-3rd person singular present
1178	VBZ	verb, 3rd person singular present
1179	WDT	wh-determiner
1180	WP	wh-pronoun, personal
1181	WP\$	wh-pronoun, possessive
1182	WRB	wh-adverb
1183	SP	space (English), sentence-final particle (Chinese)
1184	ADD	email
1185	NFP	superfluous punctuation
1186	XX	unknown
1187	_SP	whitespace
	SPECIAL	special token

A.10 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 outputs 20 tags (Tab. 16) and LIME Tokenization extends its vocabulary with one additional special token.

Table 16: 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