

Token Erasure as a Footprint of Implicit Vocabulary Items in LLMs

Sheridan Feucht David Atkinson Byron C. Wallace David Bau

Northeastern University

{feucht.s, atkinson.da, b.wallace, d.bau}@northeastern.edu

Abstract

LLMs process text as sequences of tokens that roughly correspond to words, where less common words are represented by multiple tokens. However, individual tokens are often semantically unrelated to the meanings of the words/concepts they comprise. For example, Llama-2-7b’s tokenizer splits the word “northeastern” into the tokens [`_n`, `ort`, `he`, `astern`], none of which correspond to semantically meaningful units like “north” or “east.” Similarly, the overall meanings of named entities like “Neil Young” and multi-word expressions like “break a leg” cannot be directly inferred from their constituent tokens. Mechanistically, how do LLMs convert such arbitrary groups of tokens into useful higher-level representations? In this work, we find that last token representations of named entities and multi-token words exhibit a pronounced “erasure” effect, where information about previous and current tokens is rapidly forgotten in early layers. Using this observation, we propose a method to “read out” the implicit vocabulary of an autoregressive LLM by examining differences in token representations across layers, and present results of this method for Llama-2-7b and Llama-3-8b. To our knowledge, this is the first attempt to probe the implicit vocabulary of an LLM.¹

1 Introduction

Despite their widespread use, the specific mechanisms by which LLMs are able to “understand” and generate coherent text are not well understood. One mystery is the process by which groups of subword tokens are converted into meaningful representations, a process described by Elhage et al., 2022 and Gurnee et al., 2023 as *detokenization*.

Current language models process text as a series of tokens drawn from a set token vocabulary: One token can correspond to a single word (`_fish`),



Monk’s compositions and improvisations feature dis.son.ances and angular mel.o.dic twists, often using flat nin.th.s, flat fifth.s, unexpected chrom.atic notes together, low bass notes and st.ride, and fast whole tone runs, combining a highly per.cussive attack with abrupt, dram.atic use of switched key releases, sil.ences, and hes.i.tations.

score	tokens	score	tokens
0.582	dramatic	0.315	stride
0.555	twists	0.234	melodic
0.415	low bass	0.203	silences
0.339	flat ninth.s,	0.183	s,
0.321	Monk’	0.028	together,
		0.016	, and fast whole

Figure 1: We observe “erasure” of token-level information in later layers of LLMs for multi-token words and entities (top). We hypothesize that this is a result of a process that converts token embeddings into useful lexical representations, and introduce a new method for enumerating these lexical items (bottom).

or to a piece of a larger word (`mon` in “salmon”). The vocabulary of tokens available to a model is typically determined before training with byte-pair encoding (Sennrich et al., 2016), which is based on a specific dataset and can lead to unintuitive results. For example, Llama-2-7b’s (Touvron et al., 2023) tokenizer breaks the word “northeastern” into the tokens [`_n`, `ort`, `he`, `astern`], none of which correspond to semantically meaningful units like “north” or “east.” Capitalization also creates unexpected issues: for example, the word “Hawaii” is split into two tokens if the first letter is capitalized [`_Hawai`, `i`], but four if the first letter is lowercase [`_ha`, `w`, `ai`, `i`]. In spite of these challenges, large models are apparently able to “understand” such idiosyncratic tokenizations of multi-token words with few observable effects on downstream performance (Gutiérrez et al., 2023), unless these weaknesses are directly targeted (Wang et al., 2024; Batsuren et al., 2024). How is this possible?

We hypothesize that during pretraining, LLMs

¹Code and data available at footprints.baulab.info

develop an *implicit vocabulary* that maps from groups of arbitrary tokens to semantically meaningful units. These lexical units may be multi-token words (“northeastern”), named entities (“Neil Young”), or idiomatic multi-word expressions (“break a leg”) and can be understood as “item[s] that function as single unit[s] of meaning” in a model’s vocabulary (Simpson, 2011). Lexical items are also non-compositional: Just as the meaning of “break a leg” cannot be predicted from the individual meanings of “break” and “leg,” the meaning of “patrolling” cannot be predicted from its constituent tokens *pat* and *rolling*. This arbitrariness necessitates some kind of storage system, implicit or otherwise (Murphy, 2010).

How exactly do LLMs deal with these cases mechanistically? In this paper, we begin to answer this question by investigating token-level information stored in LLM representations.

- We find that last token positions of multi-token words and named entities “erase” token-level information in early layers for both Llama-2-7b (Touvron et al., 2023) and Llama-3-8b (Meta, 2024).
- We develop a heuristic for scoring the “lexicality” of a given sequence of tokens, and use it to “read out” a list of an LLM’s lexical items given a large dataset of natural text.

We interpret this erasure effect as a “footprint” of a mechanism in early layers that orchestrates the formation of meaningful lexical items.

2 Background

Previous work has shown that knowledge about a multi-token entity is often stored in the last token of that entity. For example, Meng et al. (2022) found that factual information about a subject like “The Space Needle” would be concentrated in the representation for *le*. Geva et al. (2023) find evidence for a *subject enrichment stage* during factual recall, where information about an entity is collected at its last token in early layers, which is also seen in other work on factual recall using the same dataset (Katz et al., 2024), and corroborated by research on athlete → sport lookups (Nanda et al., 2023). This phenomenon may be due to the autoregressive nature of decoder transformer models: models cannot enrich “Space” with information about Seattle until after “Needle” is seen, as “Space” could refer

to a number of unrelated concepts (“Space Jam,” “Space Station”).²

Other work in interpretability has also started to uncover evidence of models encoding lexical items. Elhage et al. (2022) observe neurons in early layers that fire on the last tokens of multi-token words, names of famous people, generic nouns, compound words, and LaTeX commands. They also find late-layer neurons that seem to be relevant to *retokenization*, i.e., conversion from internal representations back into tokens. For example, a retokenization neuron might fire on *_st* and promote *rag* in order to facilitate the output of the word “straggler.” Gurnee et al. (2023) also find examples of polysemantic neurons in Pythia models (Mallen and Belrose, 2023) that activate for a number of multi-token constructions like “apple developer,” “Bloom.ington,” and “research.gate.”

3 Linear Probing of Hidden States

3.1 Method

If last token positions are so important (Section 2), then what do these representations encode? Perhaps the last hidden state directly stores information about other subject tokens (e.g., *_Wars* might contain some encoding for *_Star* in its hidden state). To test this hypothesis, we investigate hidden states for both Llama-2-7b and Llama-3-8b, as they have significantly different token vocabulary sizes $|\mathcal{V}|$ (32k and 128k tokens, respectively).

Let d denote the hidden dimension of the model. We train linear probes $p_i^{(\ell)} : \mathbb{R}^d \rightarrow \mathbb{R}^{|\mathcal{V}|}$ to take a hidden state $h_t^{(\ell)} \in \mathbb{R}^d$ at layer ℓ and token position t and predict the value of a nearby token $t + i$. For example, a probe trained to predict the previous token for hidden states at layer 5 would be denoted by $p_{-1}^{(5)}$.

We train probes for all layer indexes $0 \leq \ell < 32$ and offsets $i \in \{-3, -2, -1, 0, 1\}$. We also train probes in the same manner on the embedding layer ($\ell = -1$) and on the final outputs of the network before the language modelling head ($\ell = 32$). We trained probes on a random sample of 428k tokens from the Pile (Gao et al., 2020) using AdamW for 16 epochs with a batch size of 4 and a learning rate of 0.1. Hyperparameters were selected based on validation performance on a separate Pile sample

²This is not a hard-and-fast rule; it depends on entity frequency and context cues. For example, if a model sees *_The*, *_E*, and *iff*, it may already know that these tokens refer to “The Eiffel Tower” without needing to see *e1* and *Tower*.

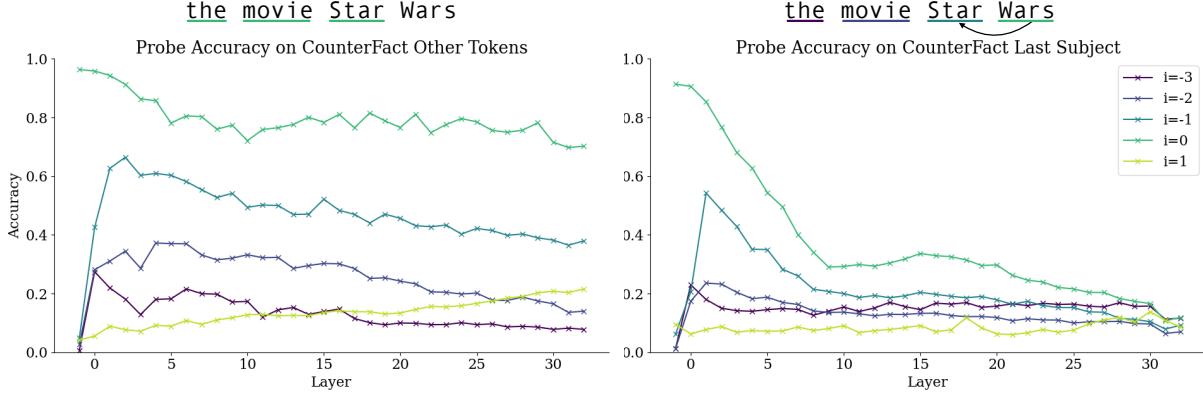


Figure 2: Top-1 test accuracy on COUNTERFACT subject last tokens versus other tokens in the dataset for probes trained on Llama-2-7b hidden states ($n = 5063$). i represents the position being predicted (e.g., $i = -1$ is previous token prediction; $i = 1$ is next-token prediction). We observe an “erasure” effect in last subject tokens that is not present for other types of tokens: these last subject tokens consistently “forget” about preceding tokens and themselves. Appendix A shows Llama-3-8b results and in-distribution performance on Pile tokens.

(279k tokens) after a random sweep. Each probe takes 6–8 hours to train on an RTX-A6000.

3.2 COUNTERFACT Subjects

After training probes in Section 3.1, we test them on the COUNTERFACT dataset (Meng et al., 2022), which consists of prompts about subjects that require factual knowledge to complete correctly (e.g. “Mount Passel is in Antarctica”). We filter the dataset to include only prompts that the model answers correctly, yielding 5,063 examples for Llama-2-7b and 5,495 examples for Llama-3-8b. To augment this dataset, we also sampled and filtered down [album/movie/series → creator] pairs from Wikidata (Vrandečić and Krötzsch, 2014) and embedded them in prompts in the same manner, yielding a total of 12,135 correctly-answered prompts for Llama-2-7b and 13,995 for Llama-3-8b.

Figure 2 shows probe test results on COUNTERFACT last subject tokens (right) versus every other type of token in the dataset (left). We see a striking “erasure” effect for last tokens of COUNTERFACT subjects, where these hidden states consistently “forget about” preceding and current tokens. Subject tokens that are not in the last position (e.g., _Star) do not exhibit this pattern (Appendix A, Figure 13). This striking drop in token accuracy is reminiscent of the subject enrichment stage described by Geva et al. (2023), suggesting that the tokens _Star and _Wars may be overwritten in the process of representing the concept of *Star Wars*.

We also observe the same phenomenon when testing on named entities identified by spaCy in Wikipedia articles (Appendix A, Figure 12), sug-

gesting that this effect is not an artifact of the short templates found in the COUNTERFACT dataset. It also does not seem to be a result of any imbalances in probe training data (Appendix B).

3.3 Multi-Token Words

Intuitively, the process of converting a multi-token sequence like [_n, ort, he, astern] into a meaningful representation of the word “northeastern” resembles the process of converting [_E, iff, e1, Tower] into “Eiffel Tower.” We hypothesize that models treat multi-token words in the same way that they treat multi-token entities, and test our probes from Section 3.1 on multi-token words. After sampling 500 articles (~256k tokens) from the 20220301.en split of the Wikipedia dump (Foundation, 2022), we split by white-space to naively identify word boundaries. As predicted, we see the same “erasing” pattern for multi-token words that we do for multi-token entities (Figure 3). This suggests that they may be processed in a similar manner in early layers. Appendix A shows similar results for Llama-3-8b.

4 Building a Vocabulary

After examination of probe behavior for multi-token words and entities, we hypothesize that this “erasure” effect is a result of the implicit formation of lexical representations in early layers. To characterize this phenomenon, we propose an *erasure score* ψ to identify token sequences that follow the pattern observed in Section 3. We then introduce an approach to “reading out” a list of implicit vocabulary entries for a given model using this score.