Accurate Word Probability Estimation in Causal LLMs

Ning Shi, Grzegorz Kondrak

Alberta Machine Intelligence Institute (Amii)
Department of Computing Science
University of Alberta, Edmonton, Canada

When a Token Is Not a Word

Text String

"My name is Shining."

Token Sequence (Llama-3.2-3B)

["My", "Ġname", "Ġis", "ĠSh", "ining", "."]

Beginning of Word (BOW) Tokenizer

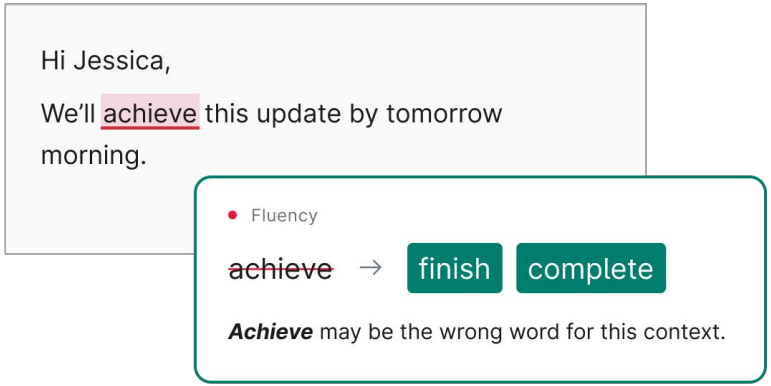
"Ġ" is the special bow mark representing the space.

"Shining" -> "ĠSh" + "ining"

Why Do We Need Word Probabilities?

Many downstream tasks operate at the word level.

- Surprisal (psycholinguistic modeling of expectation)
- Perplexity (language models benchmarking)
- Scoring (fluency, reranking)
- Lexical Semantics (word sense disambiguation, lexical substitution)
- Natural Language Inference / Reasoning (I will show this later.)



Why Do We Need Word Probabilities?

Many downstream tasks operate at the word level.

- Surprisal (psycholinguistic modeling of expectation)
- Perplexity (language models benchmarking)
- Scoring (fluency, reranking)
- Lexical Semantics (word sense disambiguation, lexical substitution)
- Natural Language Inference / Reasoning (I will show this later.)

The problem

Causal LLMs output probabilities **over tokens**, **not words**.

Language Model or Next Token Predictor?

Language Modeling

LMs are defined to assign a probability to each possible next word

Next Token Prediction

Predict the next token given the previous context

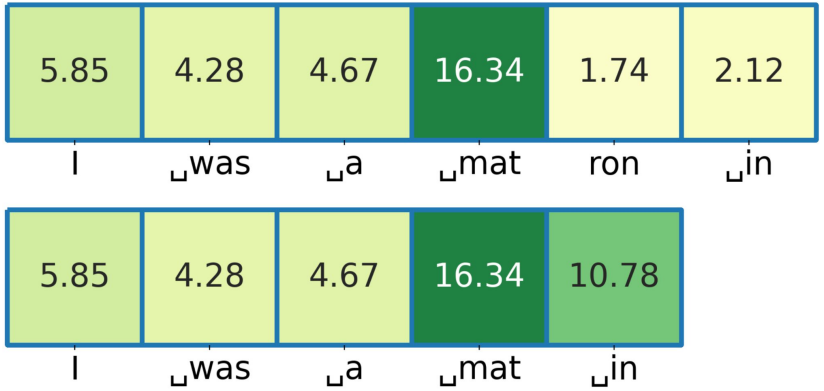
Next Word Prediction

Predict the next word given the previous context

Jurafsky, D., & Martin, J. H. (2024). Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition (3rd ed., draft).

Problems

Causal LLMs do not have an end-of-word (EOW) symbol. As the model can not tell when a word ends, so it always assumes more tokens might follow.



Surprisal values calculated with leading whitespaces.

Leading Whitespaces of Language Models' Subword Vocabulary Pose a Confound for Calculating Word Probabilities (Oh & Schuler, EMNLP 2024)

Problems

We can not use token probability to represent word probability.

Many tokens are just the start of a word, not a full word.

Input: “My name is”

Step 0: “ĠK” (0.0094)

But what does “ĠK” mean?

It could be the start of: Kelsey, Kait, Kari, etc.

(“Kelsey” -> [“ĠK”, “elsey”])

Problems

Next-token probability can include both a word and its longer extensions, so using it directly overestimates the true probability of the word.

Input: “My name is”

Step 0: “Ġ John” (0.0073)

Step 1:

“Ġand” (0.1085) -> “My name is John and”

“athan” (0.0183) -> “My name is Johnathan”

Problems

Using token probabilities directly can cause the total probability of all words to exceed 1, which breaks basic probability rules.

Given "I was a matron", suppose the LM gives:

P("Ġmat") = 0.6 (mat)

P("Ġmat") * P("ron" | "Ġmat") = 0.6*0.9 = 0.54 (matron)

0.6 + 0.54 > 1

This violates the Kolmogorov axiom.

Leading Whitespaces of Language Models' Subword Vocabulary Pose a Confound for Calculating Word Probabilities (Oh & Schuler, EMNLP 2024)

Bug Fix

$$\mathcal{P}(w \mid C) = \mathcal{P}(\mathbf{S}^w \mid \mathbf{S}^C)$$

$$\mathcal{P}(w \mid C) = \mathcal{P}(\mathbf{s}^w \mid \mathbf{s}^C) \underbrace{\frac{\sum_{s \in \mathcal{S}_{\text{bow}}} \mathcal{P}(s \mid \mathbf{s}^C, \mathbf{s}^w)}{\sum_{s \in \mathcal{S}_{\text{bow}}} \mathcal{P}(s \mid \mathbf{s}^C)}}_{\text{“bug” fix}}$$

Numerator: probability that generation stops after the word

Denominator: probability that any word would begin after the context

How to Compute the Probability of a Word (Pimentel & Meister, EMNLP 2024)

Bug Fix

$$\mathcal{P}(w \mid C) = \mathcal{P}(\mathbf{S}^w \mid \mathbf{S}^C)$$

$$\mathcal{P}(w \mid C) = \mathcal{P}(\mathbf{s}^w \mid \mathbf{s}^C) \underbrace{\frac{\sum_{s \in \mathcal{S}_{\text{bow}}} \mathcal{P}(s \mid \mathbf{s}^C, \mathbf{s}^w)}{\sum_{s \in \mathcal{S}_{\text{bow}}} \mathcal{P}(s \mid \mathbf{s}^C)}}_{\text{“bug” fix}}$$

Works well for scoring a known word in a given context.

However, it can not search over the space of possible words (Open Vocabulary)

How to Compute the Probability of a Word (Pimentel & Meister, EMNLP 2024)

Token-to-Word Decoding

EOW Injection

We simulate an End-of-Word (EOW) event by:

Identifying BOW tokens (tokens that can start a new word)

Reallocating their probability mass to a virtual <*eow*> token

This tells the model: “you may stop generating the current word now”

$$\mathcal{P}(\text{“<}eow\text{>”} \mid \mathbf{s}^C, \mathbf{s}^w) = \sum_{s \in \mathcal{S}_{\text{bow}}} \mathcal{P}(s \mid \mathbf{s}^C, \mathbf{s}^w)$$

Token-to-Word Decoding

Dynamic Normalization

At depth 1 (start of word): restrict to BOW tokens

At depth > 1 (inside word): restrict to mid-word tokens only (no BOW)

$$\mathcal{S}' = \begin{cases} \mathcal{S}_{\text{bow}} & \text{if } d = 1 \\ \mathcal{S}_{\text{mid}} & \text{if } d > 1 \end{cases}$$

In practice, it depends on the tokenizer and the language. For example, some words include non-alphabetical characters such as “N’Djamena”.

Token-to-Word Decoding

Beam Search

We do not just want to estimate the probability of one known word.

We want to explore top-k most probable word completions.

Exhaustive search over all token paths is intractable: $O(|\mathcal{S}|^D)$

We keep the top beam width candidates, so

$$O(D \cdot |\mathcal{S}'| \cdot W), \text{ where } \mathcal{S}' \ll \mathcal{S}$$

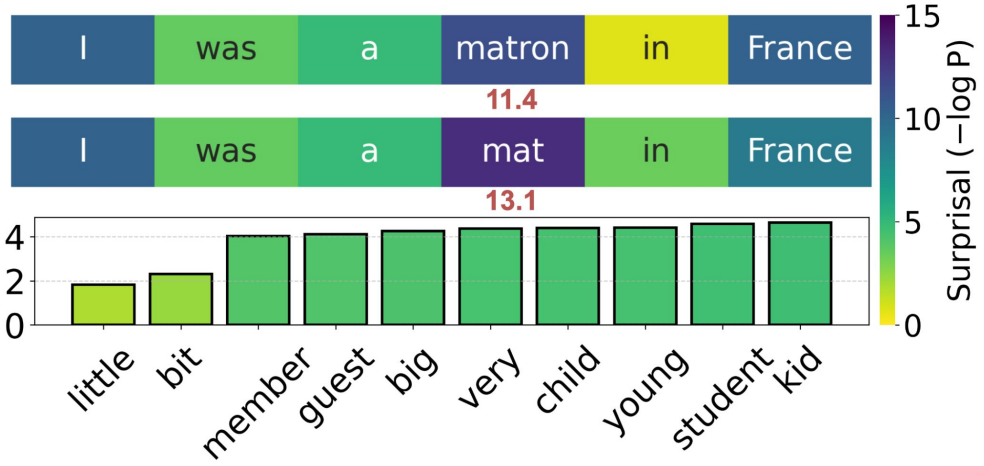
Token-to-Word Decoding

Algorithm 1 Token-to-Word Decoding

Input: tokenized context $\mathbf{S}^C$, beam width W beam depth D, language model $\mathrm{F}(\cdot)$

$B_0 \leftarrow \{\langle 0, \mathbf{S}^C \rangle\}$
for $d \in \{1, \ldots, D\}$:
  $B \leftarrow \varnothing$
  for $\langle p, \mathbf{S} \rangle \in B_{d-1}$:
    if $\mathbf{S}$.last() = "<*eow*>" :
      B.add($\langle p, \mathbf{S} \rangle$); **continue**
    $\mathcal{P} \leftarrow \mathrm{F}(\mathbf{S})$
    if $d = 1$:
      $\mathcal{P} \leftarrow \mathrm{Normalize}(P, \mathcal{S}_{\text{bow}})$
    else:
      $\mathcal{P} \leftarrow \mathrm{InjectEOW}(\mathcal{P}, \mathcal{S}_{\text{bow}})$
      $\mathcal{P} \leftarrow \mathrm{Normalize}(\mathcal{P}, \mathcal{S}_{\text{mid}})$
    for $(p', s) \in \mathrm{Top}(\mathcal{P}, \mathrm{W})$:
      B.add($\langle p \cdot p', \mathbf{S} \circ s \rangle$)
  $B_d \leftarrow B$.top(W)
return B_D.sort()

Unit Test



Surprisal values for the example proposed by Oh and Schuler (2024).

Leading Whitespaces of Language Models' Subword Vocabulary Pose a Confound for Calculating Word Probabilities (Oh & Schuler, EMNLP 2024)

CON2LM - Contradiction Detection

Surprisal Word

Given a text T, a word w in T is a surprisal word if after reading the text up to and including w it can be concluded that the text contains a contradiction.

Hypothesis

A text T involves a contradiction if and only if T contains a surprisal word w.

$$\text{Surprisal}(w \mid C) = -\log \mathcal{P}(w \mid C)$$

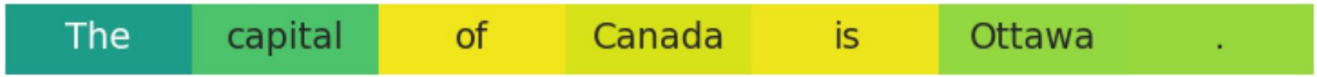
Results - Wiki Capital

Wiki Capital

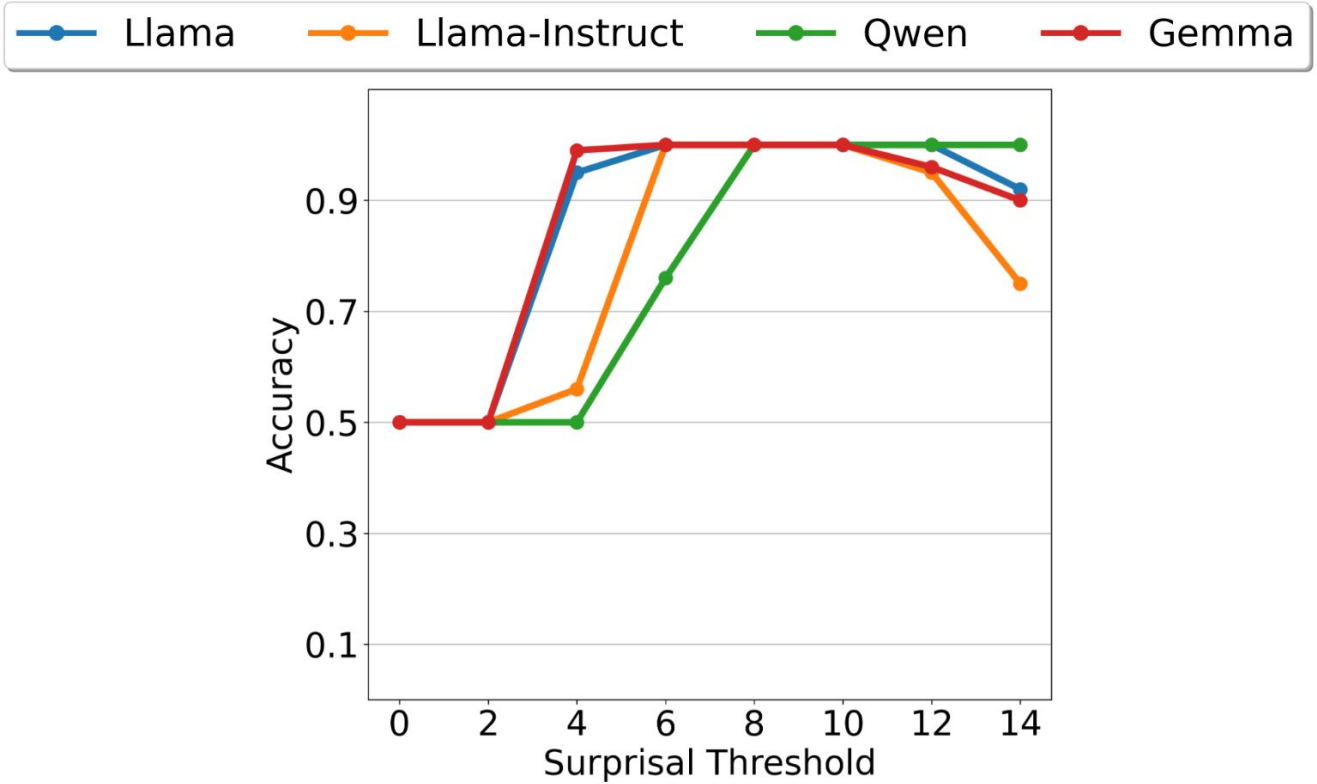
Context: None

Context: The capital of Canada is Edmonton.

Context: Since the capital of Canada is Edmonton, therefore



Results – Wiki Capital



Error Cases – bAbI

Open Challenges

Invalid Token Paths

Not all token continuations lead to valid words. And due to the nature of the softmax function, LLM will never assign zero probability to a token.

> “My name is Shining.”
>
> [“My”, “Ġname”, “Ġis”, **“ĠSh”**, **“ining”**, “.”]
>
> [“My”, “Ġname”, “Ġis”, **“ĠSh”**, **[Any Token]**, “.”]

Open Challenges

Word Probability Underestimation

While only the top (greedy) path is often used in practice, BPE tokenization allows multiple ways to form the same word.

Input: "The capital of Canada is"

Greedy path: [“ĠOttawa”] (0.2309)

A possible path: [“ĠO” (0.00015), “tt” (0.0156), “awa” (0.0047)]

Takeaways

- Token is not Word.
- Naive Estimation Fails.
- Existing “Bug Fix” does not support open vocabulary.
- Token-to-word decoding is one of the solutions.
- It enables lexical surprisal analysis and contradiction detection.
- Challenges remain.
    - Invalid Token Paths
    - Word Probability Underestimation