



Tokenization and Embedding

Lei Li



Language
Technologies
Institute

Carnegie Mellon University

School of Computer Science

Recap

- Key components in Transformer and LLM Architecture
 - Positional Embedding (to distinguish tokens at different pos)
 - Multihead attention
 - Residual connection
 - layer norm
 - FFN and SwishGLU
 - RoPE

Recap

- Training Technique:
 - Dropout
 - Label smoothing
 - optimization alg
- Pretraining: Mask-labeled recovery of random spans + next token prediction
- Multitask supervised fine-tuning with instruction templates
 - T5, InsturctGPT

Outline

- Subword tokenization: Byte-Pair-Encoding
 - Code walk through
- Information-theoretic vocabulary (VOLT)
- Practical Considerations in LLM
- Vocabulary sharing and impact on multilingual performance
- Tokenizer-free model (BLT)

Tokenizer – split text into basic units

Many words don't map to one token: indivisible.

↓ tokenizer

Many words don't map to one token: indivisible.

[7085, 2456, 836, 470, 3975, 284, 530, 11241, 25, 773, 452, 12843, 13]

↓ embedding table lookup

2.3	-3.2	8.3	5.4	2.1	3.9	-8.9	3.8	3.9	3.3
4.5	5.9	4.5	7.1	1.0	5.3	5.0	3.1	0.7	5.0
...	...	...	...	...	...	...	...	...	...
3.8	1.2	3.8	9.0	9.3	3.1	4.2	0.8	9.2	5.8

Simple Tokenization – Word-level

- Word-level Tokenization
 - Break by space and punctuation.
 - English, French, German, Spanish

The most eager is Oregon which is enlisting 5,000 drivers in the country's biggest experiment.

- Special treatment: numbers replaced by special token [number]
- How large is the Vocabulary? Cut-off by frequency, the rest replaced by [UNK]

Vocabulary – simple example

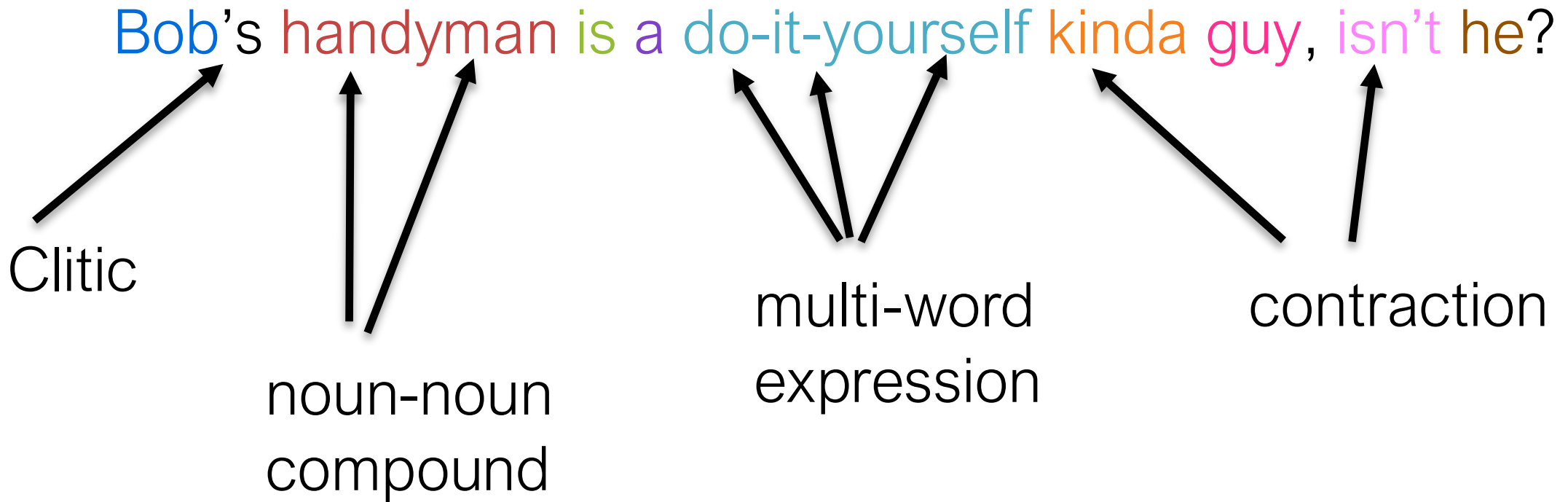
```
from collections import defaultdict

def build_word_dict(file_path):
    word_dict = defaultdict(int)
    with open(file_path, 'r', encoding='utf-8') as file:
        for line in file:
            words = line.split() # Split by spaces
            for word in words:
                word_dict[word] += 1

    return word_dict
```

What is a word?

How many words?



Words

- Orthographic definition
 - strings separated by white spaces
 - spoken language: units corresponding to written word separated by pause
 - problem: Bob's handy man is a do-it-yourself kinda guy, isn't he?
- What about languages that do not use white spaces?

他昨天晚上去看了消失的她

he yesterday night watched lost in stars

Pros and Cons of Word-level Tokenization

- Easy to implement
- Cons:
 - Out-of-vocabulary (OOV) or unknown tokens, e.g. Covid
 - Tradeoff between parameters size and unknown chances.
 - Smaller vocab => fewer parameters to learn, easier to generate (deciding one word from smaller dictionary), more OOV
 - Larger vocab => more parameters to learn, harder to generate, less OOV
 - Hard for certain languages with continuous script: Japanese, Chinese, Korean, Khmer, etc. Need separate word segmentation tool (can be neural networks)

Character-level Tokenization



T h e m o s t e a g e r i s O r a n g e ...

- Each letter and punctuation is a token
- Pros:
 - Very small vocabulary (except for some languages, e.g. Chinese)
 - No Out-of-Vocabulary token
- Cons:
 - A sentence can be longer sequence
 - Tokens do not representing semantic meaning

Subword-level Tokenization

The most eager is Oregon which is enlisting 5,000 drivers in the country's biggest experiment.

- Goal:
 - moderate size vocabulary
 - no OOV
- Idea:
 - represent rare words (OOV) by sequence of subwords
- Byte Pair Encoding (BPE)
 - not necessarily semantic meaningful
 - Originally for data compression Philip Gage. A New Algorithm for Data Compression, 1994

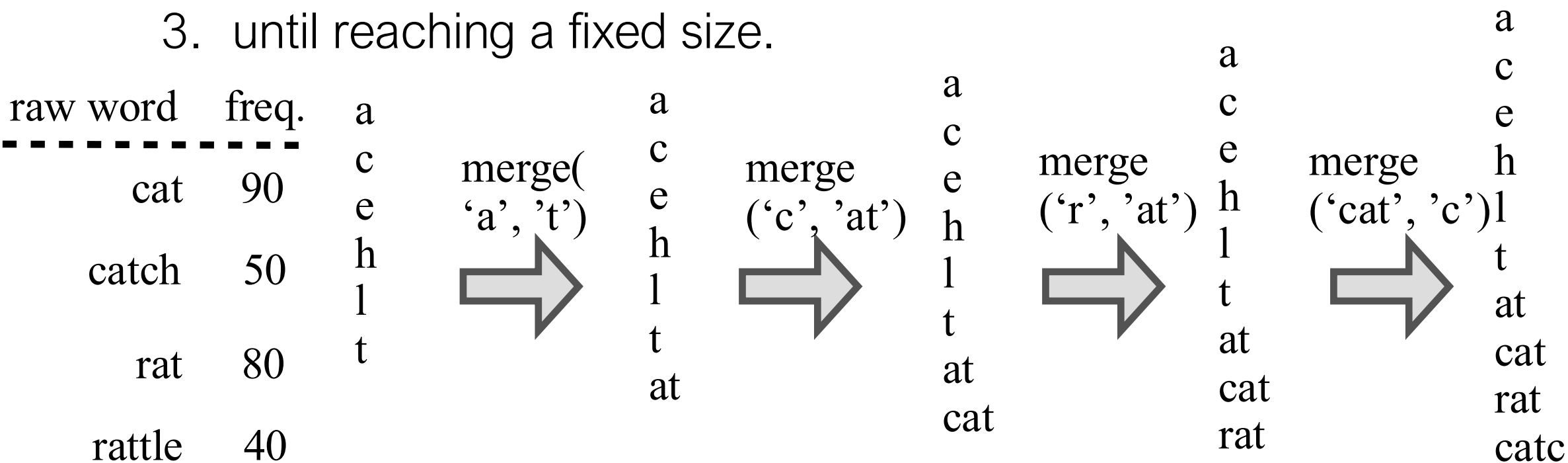
Byte Pair Encoding (BPE) for Text

Building Vocabulary

1. Initialize vocabulary with all characters as tokens (also add end-of-word symbol) and frequencies
2. Loop until vocabulary size reaches capacity
 - 1) Count successive pairs of tokens in corpus
 - 2) Rank and select the top frequent pair
 - 3) Combine the pair to form a new token, add to vocabulary
3. Output final vocabulary

Byte-Pair-Encoding Tokenization

1. starting from chars
2. repeatedly, merge most frequent pairs to form new tokens
3. until reaching a fixed size.



BPE Tokenization

- Split text by space or other delimiters
- Repeat
 - greedy find the longest prefix that matches a token in BPE dictionary
 - split and process the remaining parts until no more text left

Code Example

- https://github.com/lmsystem/lmsys_code_examples/blob/main/tokenization/tokenization.ipynb

Outline

- Subword tokenization: Byte-Pair-Encoding
 - Code walk through
- ➔ • Information-theoretic vocabulary (VOLT)
- Practical Considerations in LLM
- Vocabulary sharing and impact on multilingual performance
- Tokenizer-free model (BLT)

Measuring Vocabulary

- Compression

- average number of bytes per token

$$BPT = \frac{\# \text{ utf8 bytes}}{\# \text{ tokens}}$$

- normalized sequence length

$$NSL = \frac{\# \text{ tokens}}{\# \text{ tokens in LLaMA}}$$

- normalized entropy (next)

Find Optimal Vocabulary

Numerous possible vocabularies at the sub-word level.



Which one leads to better NLG/MT performance?

Repeated full training and testing are required to find the optimal vocabulary!(BPE-Search)

VOLT: Using entropy to learn vocabulary

- Normalized Entropy $\mathcal{H}(v) = -\frac{1}{l_v} \sum_{i \in v} P(i) \log P(i)$
 - l_v : average number of chars for v's all tokens
 - token prob.
- It measures semantic-information-per-char
 - Smaller is favorable. Less ambiguity and easy to generate

Token	count
a	200
e	90
c	30
t	30
s	90

$$\mathcal{H}(v) = 1.37$$

Token	count
a	100
aes	90
cat	30

$$\mathcal{H}(v) = 0.14$$

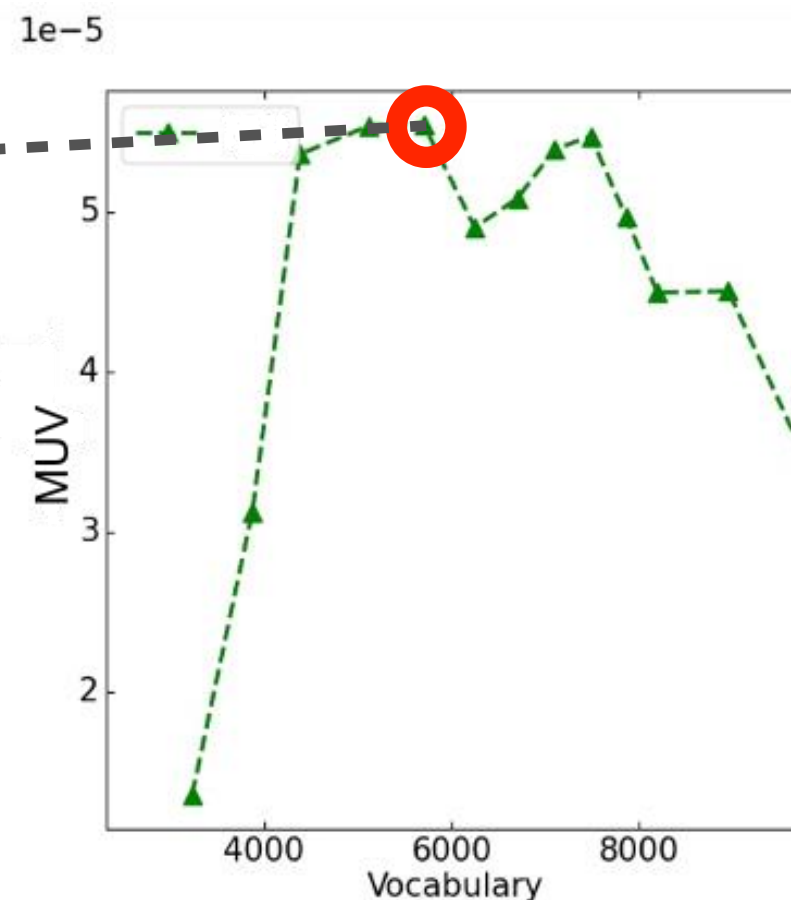
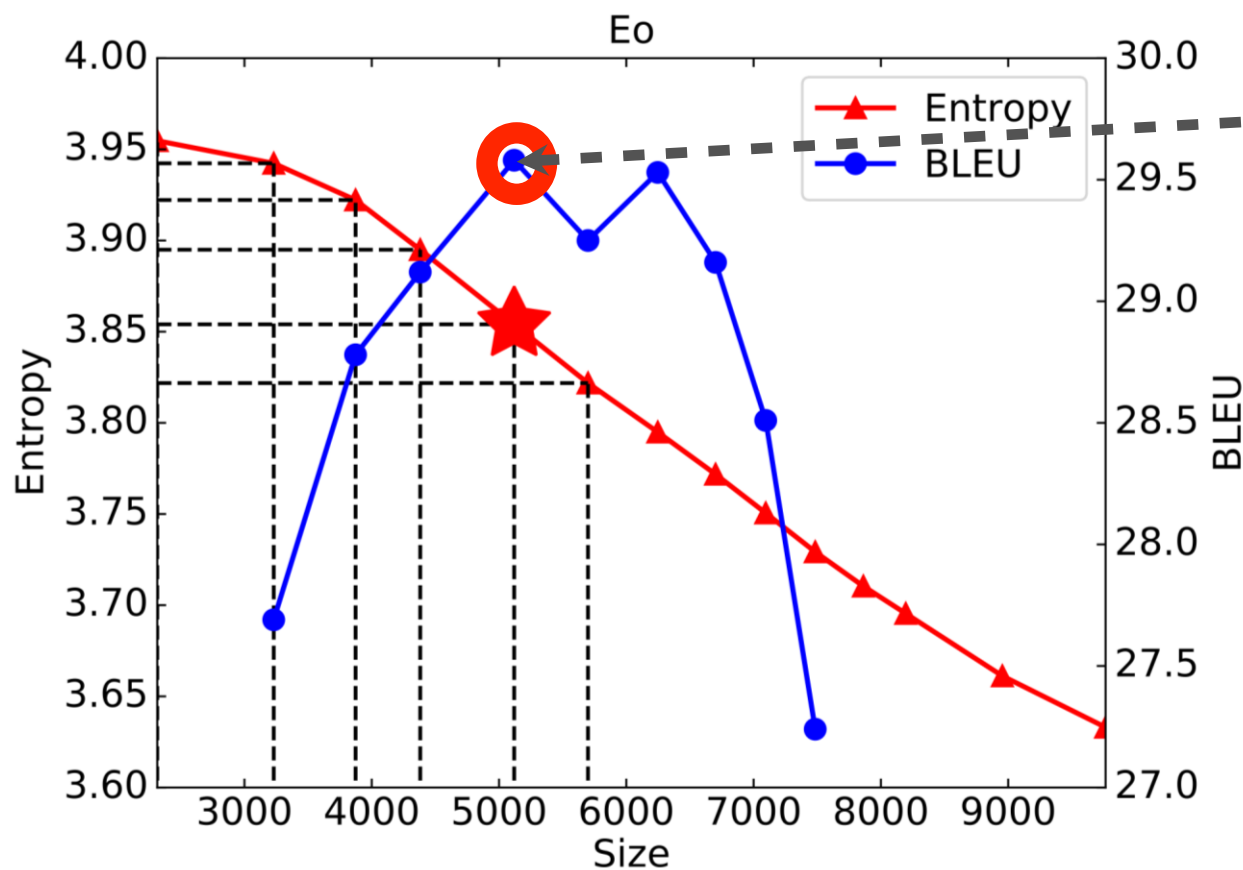


MUV: Utility of Information for Adding Tokens

- Value: Normalized Entropy 
- Cost: Size 
- Marginal Utility of information for Vocabulary (MUV)
 - $M_{v_k \rightarrow v_{k+m}} = - \frac{H(v_k) - H(v_{k+m})}{m}$
 - Negative gradients of normalized entropy to size
 - How much value each token brings

MUV is good indicator for MT performance

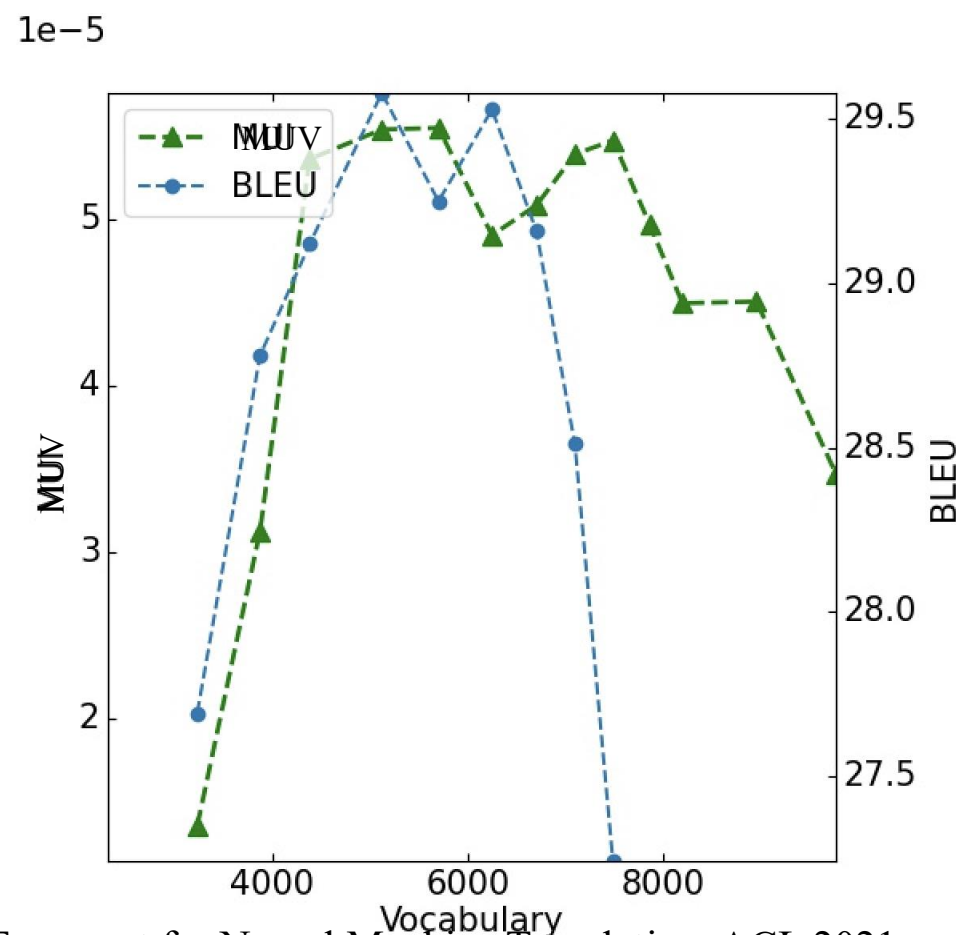
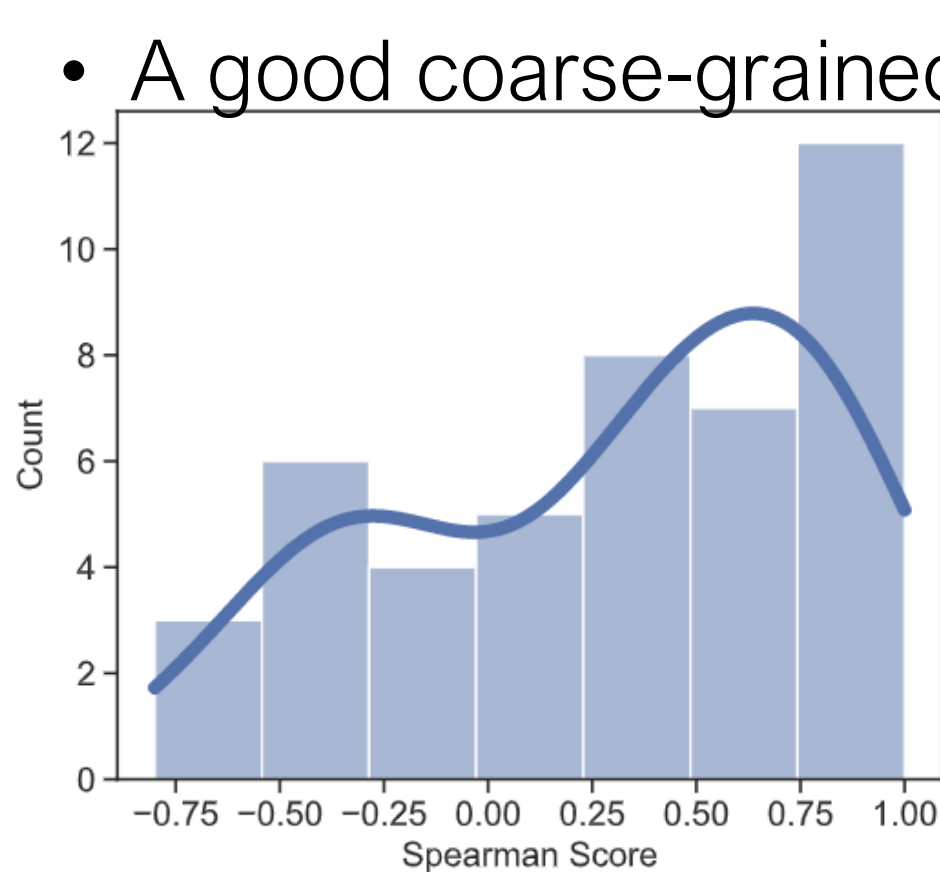
- Cost-effective point in MUV curve
 - maximum MUV => best BLEU



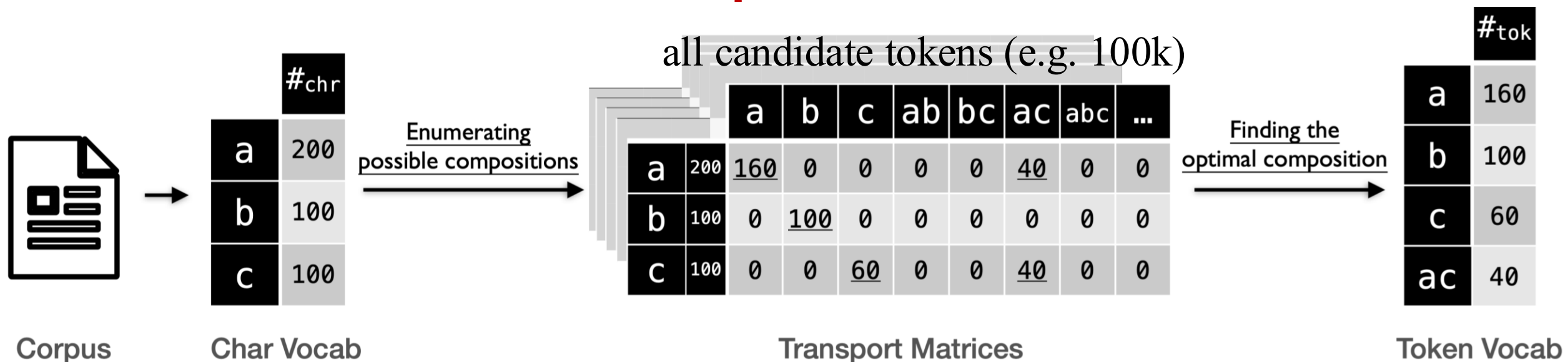
MUV Indicates MT Performance

- MUV and BLEU are correlated on two-thirds of tasks

- A good coarse-grained evaluation



VOLT: Vocabulary Building via Transportation



- Maximizing MUV for vocabulary
 - $\max - (H(V_{t+1}) - H(V_t))$
- Instead, maximizing the lower bound ==> Optimal Transport
 - $\max_t (\max H(V_t) - \max H(V_{t+1}))$

Reducing MUV Optimization to OT

- The vocabulary with the maximum MUV
 - Maximum gap between IPC of a vocabulary (with size t) and that of a smaller vocabulary (with size $< t$)
 - $\max - (H(V_{t+1}) - H(V_t))$
- Intractable, instead to maximize lower-bound
- $\Rightarrow \max_t (\max H(V_t) - \max H(V_{t+1}))$
- Finding $\max_v H(v) \Rightarrow$ Optimal Transport

VOLT: Finding the Optimal Vocabulary

- Entropy-regularized Optimal Transport

$$\min_{P \in \mathbb{R}^{m \times n}} \langle D, P \rangle - H(P)$$

subject to

$$\forall i \in Char, \sum_{j \in V_n} P_{i,j} = \hat{P}(i)$$

$$\forall j \in V_n, \left| \sum_{i \in Char} P_{i,j} - \hat{P}(j) \right| = \epsilon$$

- Sinkhorn's algorithm (from [Sinkhorn 1967])

Transportation matrix P

Char \ Tok	a	ab	bc
a	$P_{a,a}$	$P_{a,ab}$	$P_{a,bc}$
b	$P_{b,a}$	$P_{b,ab}$	$P_{b,bc}$
c	$P_{c,a}$	$P_{c,ab}$	$P_{c,bc}$

Cost matrix D

Char \ Tok	a	ab	bc
a	0	$\ln 2$	∞
b	∞	$\ln 2$	$\ln 2$
c	∞	∞	$\ln 2$

Encoding and Decoding with VOLT

- VOLT uses a greedy strategy to encode text with a constructed sub-word level vocabulary similar to BPE.
- The vocabulary includes all basic characters.
 - To encode text, it first splits sentences into character-level tokens.
 - Then, we merge two consecutive tokens into one token if the merged one is in the vocabulary solved by OT.
 - This process keeps running until no tokens can be merged.
 - Out-of-vocabulary tokens will be split into smaller tokens.

Outline

- Subword tokenization: Byte-Pair-Encoding
 - Code walk through
- Information-theoretic vocabulary (VOLT)
-  • Practical Considerations in LLM Tokenization
- Vocabulary sharing and impact on multilingual performance
- Tokenizer-free model (BLT)

Practical Consideration in LLMs

- deduplication
- sentencepiece
- Code (programming languages)
- Numbers
- multilingual

Corpus Deduplication and Filtering

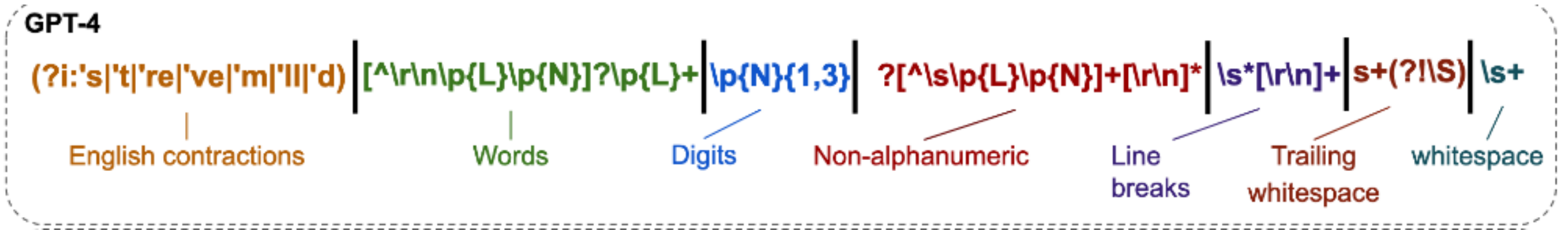
- LLaMA 3 deduplicate at
 - url level dedup
 - document level dedup using minHash
 - line level dedup using SHA-1 64 bit hash code for every 30m docs
 - to remove boilerplate, e.g. navigation menu, cookie warning, contact info
- Filter
 - n-gram repeats in one line
 - “dirty word” counting
 - token distribution KL divergence too different from corpus

Modern Subword Tokenization

- BBPE: byte-level BPE (universal for all languages)
 - treating language as sequence of unicode bytes
- Wordpiece:
 - like BPE, but instead of merge with most frequent pairs, merge a and b, if $p(b|a)$ will be maximized
- SentencePiece:
 - Uniform way to treat space, punctuation
 - Use the raw sentence, replacing space ' ' with _ (U+2581)
 - Then split character and do BPE

Handling Code: Pre-tokenization

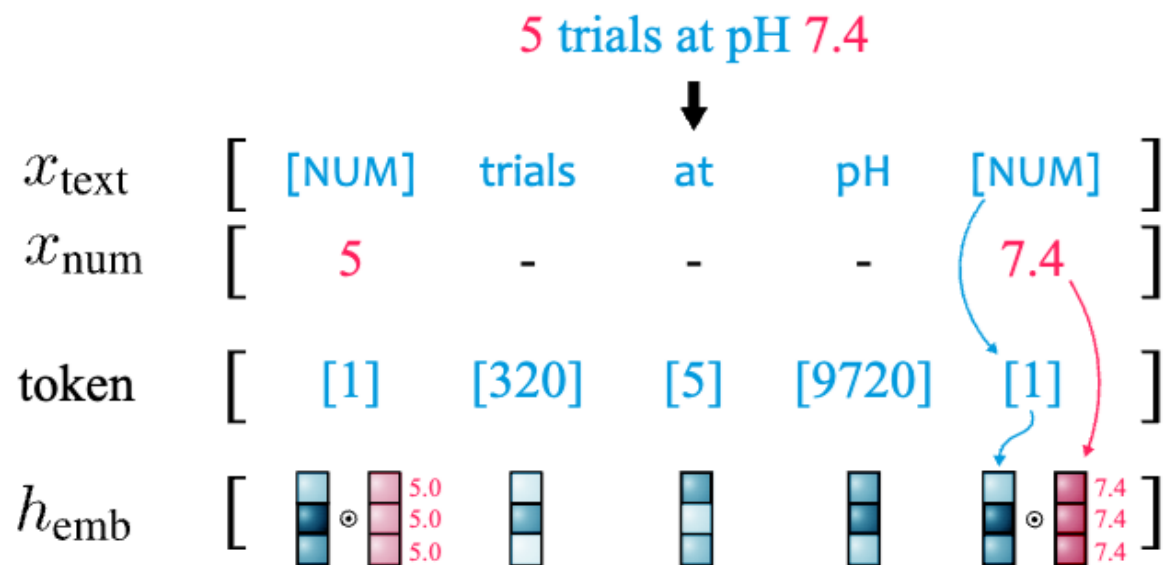
Using regular expression to split the sequences



example: `.append` => a single token

Handling numbers: Enable math in LLM

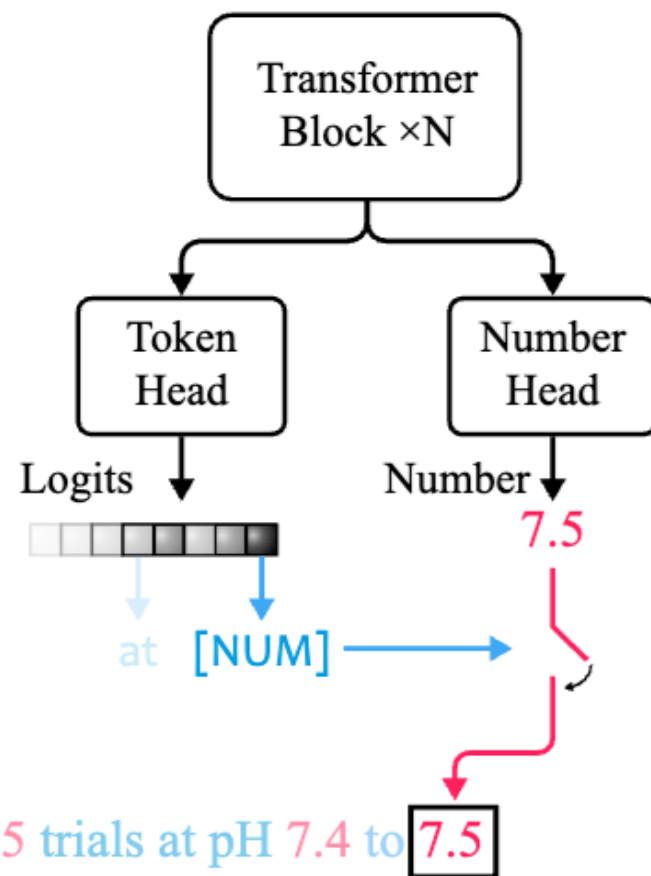
Encoding



Comparison to text-based encoding

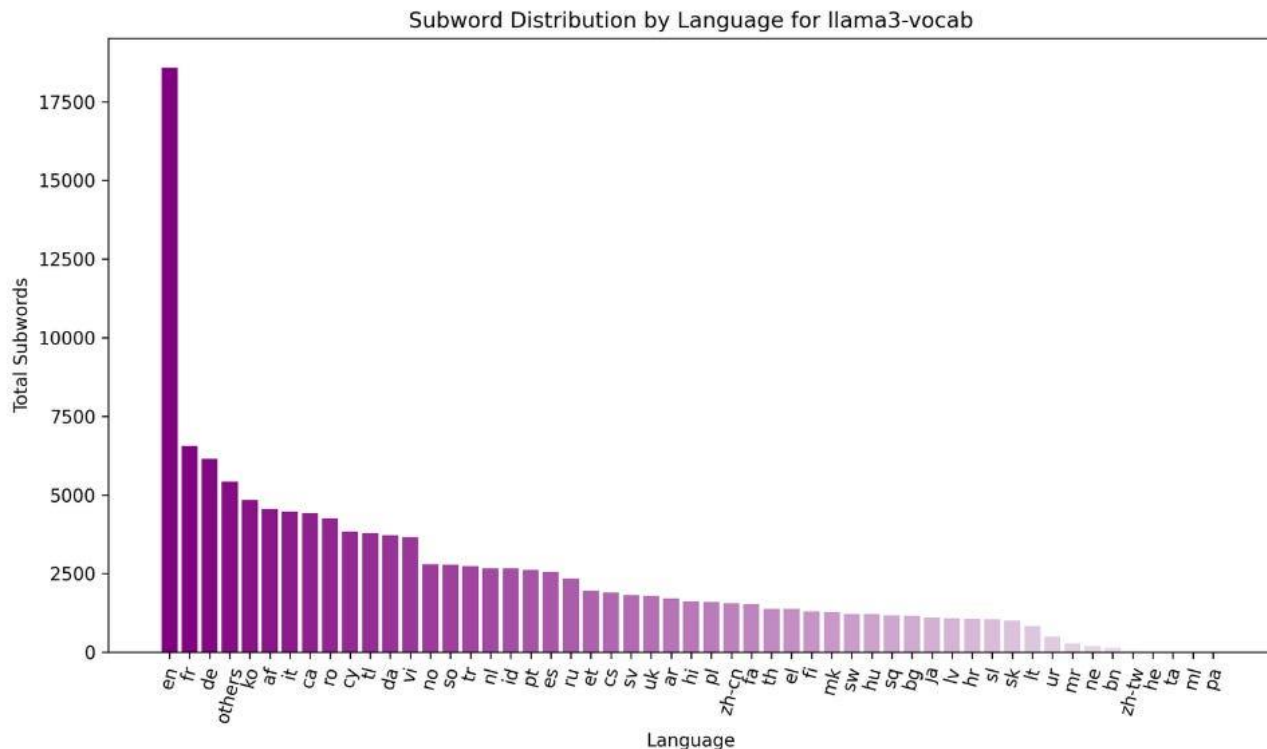
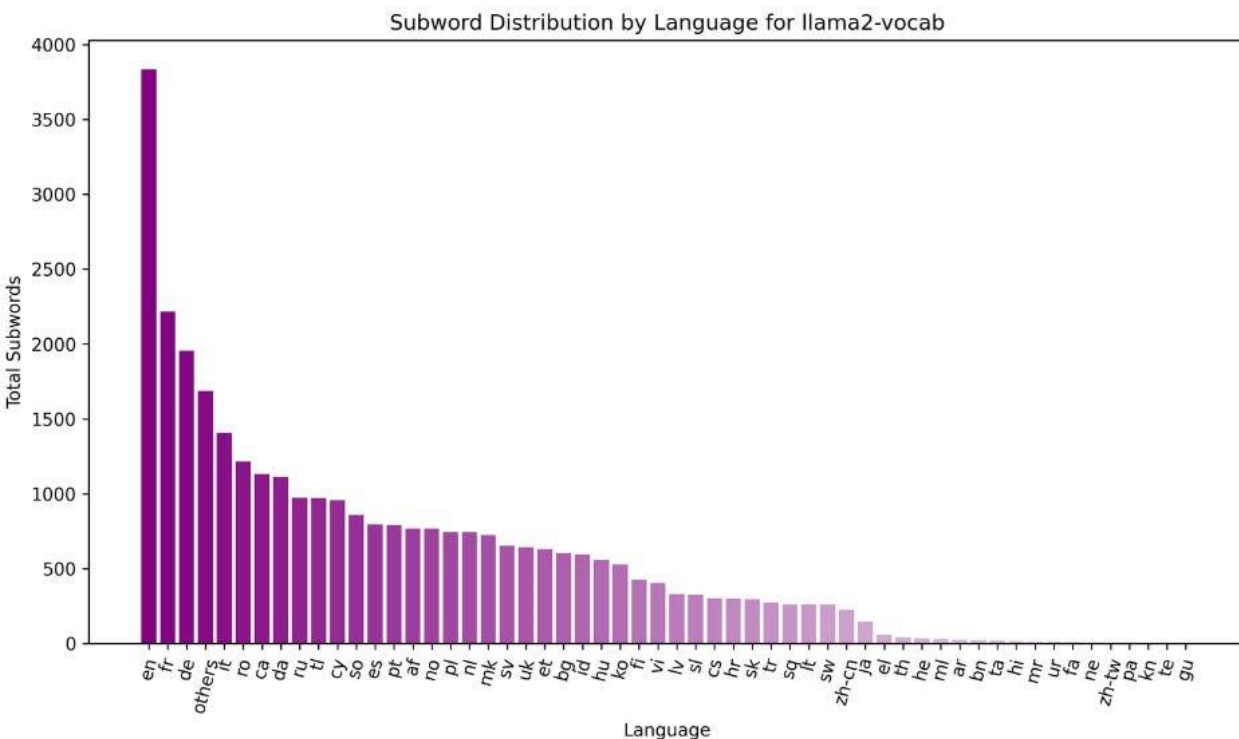
x_{text}	[	+	500	e-2	trials	at	pH	+	740	e-2	]
token	[	[82]	[9283]	[402]	[320]	[5]	[9720]	[82]	[9523]	[402]	]

Decoding



LLaMA3's multilingual vocabulary

- 32k (LLaMA 2) ➔ 128k tokens (LLaMA 3.1)
 - 100k from openAI's tiktoken (from original 200k)
 - 28k allocated to multilingual



How to construct multilingual vocabulary?

- combining documents from multiple (176 in LLaMA3) languages, about 8% of total text, then apply BPE on the joint corpus
- obtain the same amount of BPE token for each language and then merge.
- allocating the capacity of each language by balancing the average log probability (ALP)

Demo

<https://belladoreai.github.io/llama-tokenizer-js/example-demo/build/>

<https://koala.sh/tools/free-gpt-tokenizer>

Outline

- Subword tokenization: Byte-Pair-Encoding
 - Code walk through
- Information-theoretic vocabulary (VOLT)
- Practical Considerations in LLM
- ➡ • Vocabulary sharing and impact on multilingual performance
- Tokenizer-free model (BLT)

Vocabulary Sharing

English: television

Spanish: televisión

French: television

Italian: television

Dutch: televisie

Portuguese: televisão

Swedish: television

Finnish: televisio

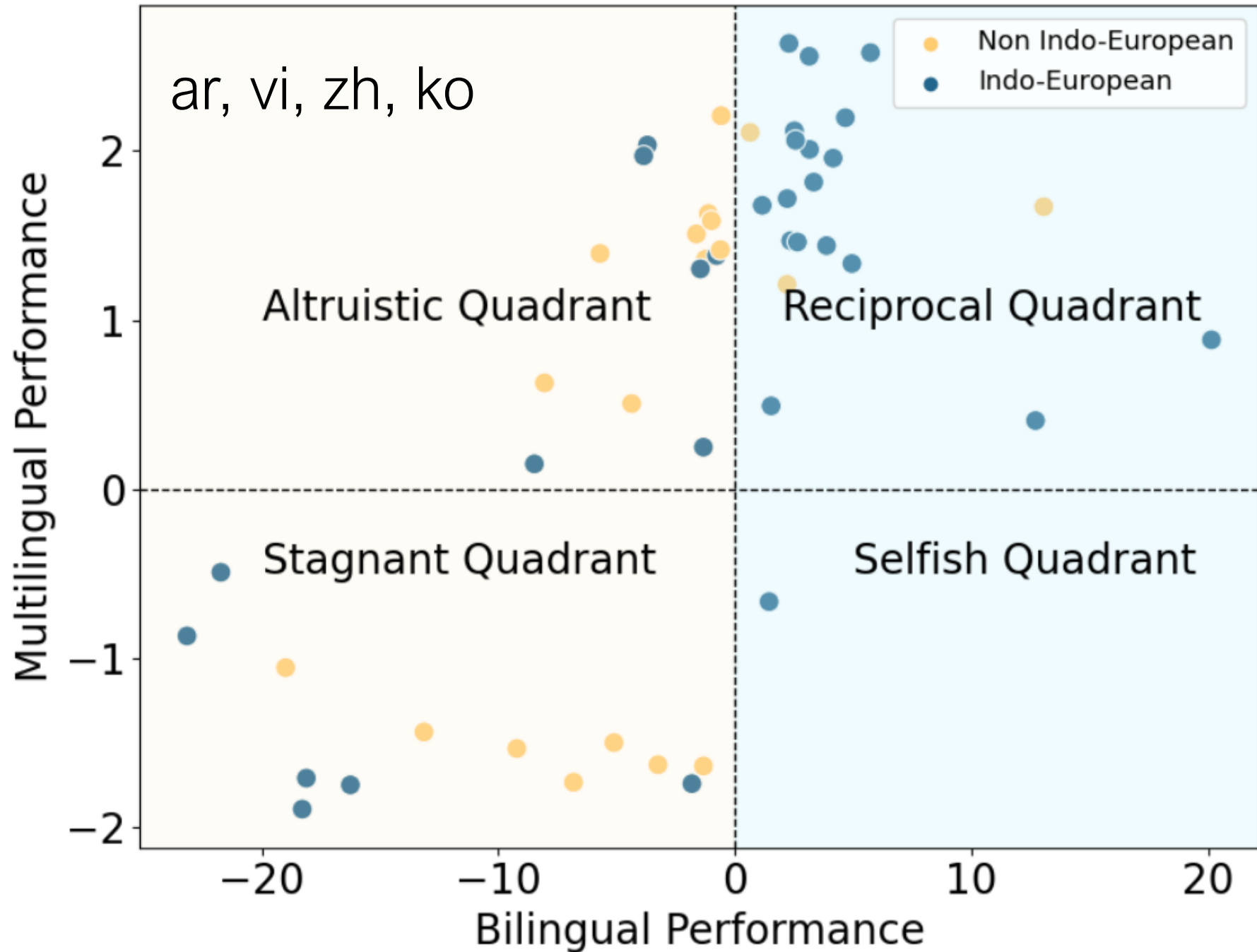
Embedding Finetuning for LLM

- Construct a small instruction-finetuning dataset using 10k bilingual parallel data
- Finetune LLaMA-7B
- Examine the translation performance of
 - The supervision bilingual direction (bilingual)
 - All other directions (multilingual)

Does embedding FT promote bilingual & multilingual translation performance?

Quadrant	Performance		Case Languages
	Bilingual	Multilingual	
Reciprocal	↑	↑	cs, da, fr, de
Altruistic	↓	↑	ar, vi, zh, ko
Stagnant	↓	↓	Km, lo, gu, te
Selfish	↑	↓	hi

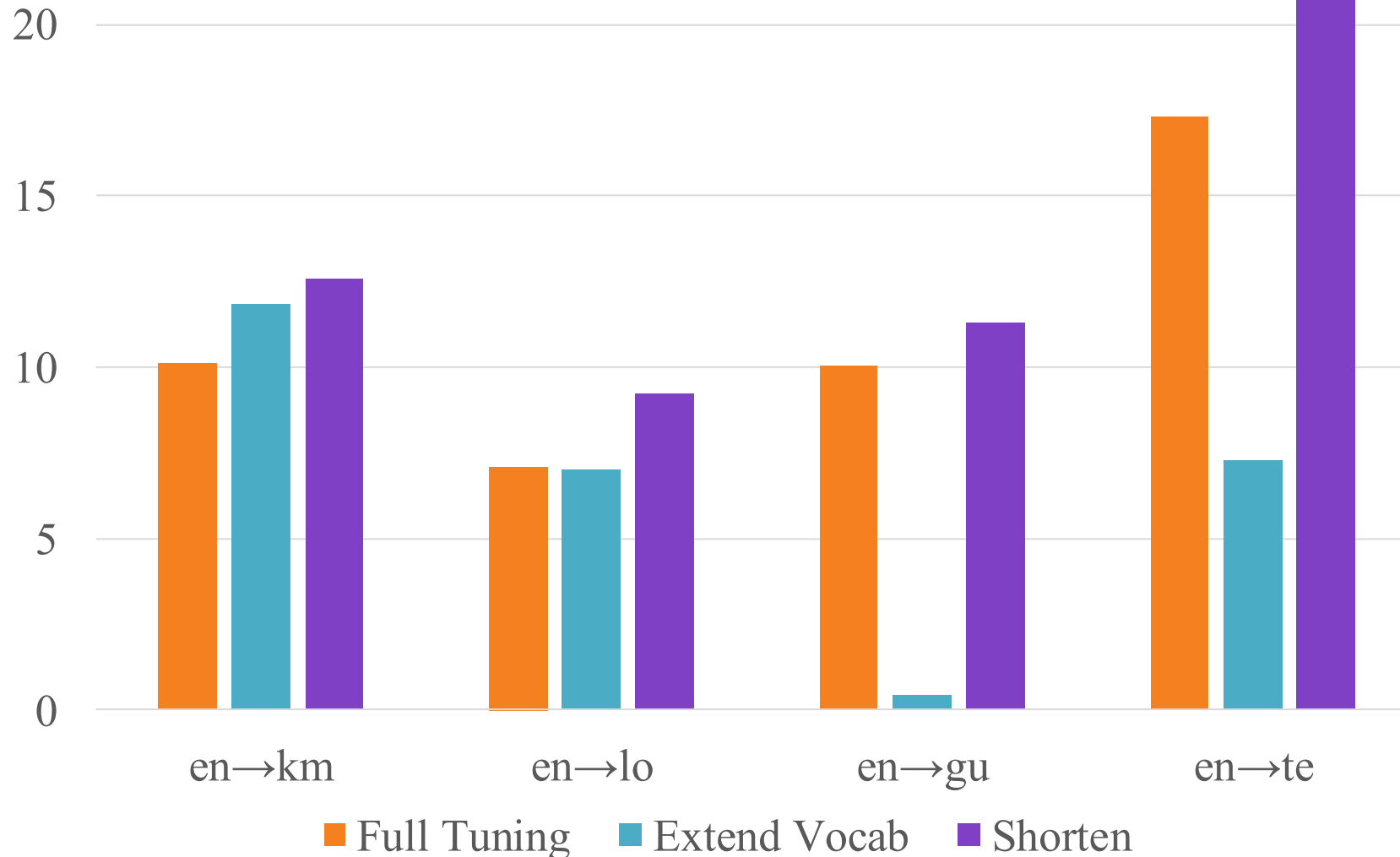
**Fine-tuning
on bilingual
data does
not always
bring
benefits to
supervised
direction!**



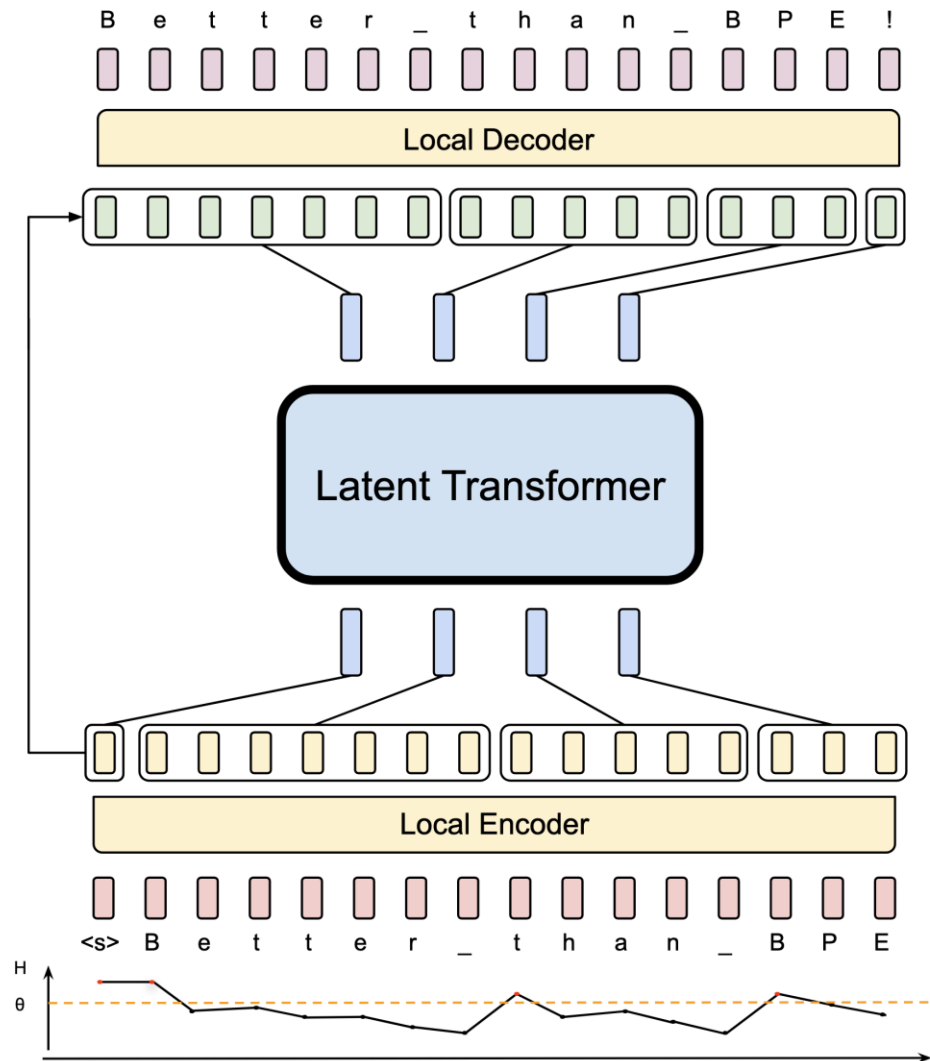
Stagnant Quadrant – Over-tokenization

- Byte-BPE (BBPE) produces longer byte level token sequence than the number of characters
- 饕 [tāo] (gluttonous) → three tokens [227, 234, 260]
- Implication for improvement:
 - shortening: remove the common prefix 227

Stagnant Quadrant: expanding vocab shortening



Tokenizer-free Model – Byte Latent Transformer



5. Small Byte-Level Transformer
Makes **Next-Byte Prediction**

4. **Unpatching** to Byte Sequence
via Cross-Attn

3. Large Latent Transformer
Predicts Next Patch

2. Entropy-Based Grouping of Bytes
Into **Patches** via Cross-Attn

1. Byte-Level Small Transformer
Encodes **Byte Stream**

Summary

- Subword tokenization: Byte-Pair-Encoding
 - iteratively merging most frequent pairs of tokens
- Information-theoretic vocabulary (VOLT)
 - solving entropy constrained optimal transport problem
- Pre-tokenization through regex
- Number treatment
- Vocab sharing impact multilingual performance
 - how to solve languages in stagnant quad

Quiz