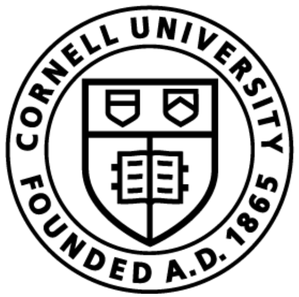
Lecture 12: Tokenizers, Retrieval Augmented LLMs



Cornell Bowers C·IS
Computer Science

Claire Cardie, Tanya Goyal

CS 4740 (and crosslists): Introduction to Natural Language Processing

Announcements

- HW4 due on May 6, 11.59 pm.
  - No penalty submission allowed up to May 9, 11.59 p.m.
- **Review session next class.**
  - Topics-wise or questions?

Today

- Tokenization
- RAG
- MiSC topics

Subword Tokenization

- BERT vocabulary = 30K tokens
  - More works exist in the dictionary!

- What will happen if we create a 100K+ word vocabulary?
  - The final matrix multiply (projecting from last layer to vocabulary) will dominate computation.
  - Adding a ton of rare words that are rarely used. Will these embeddings get learnt?
  - Morphology?

- Character-level models?
  - Don’t work great in practice
  - Sequence lengths can be very long.

Subword Tokenization

- Goal: Instead of words, create a vocabulary of “word pieces” that are between characters and full words.

positional embeddings —> _position al _embed ding s

Represents start of a word

- A number of ways the subword vocabulary can be constructed:
  - BPE: Byte Pair Encoding (Sennrich et al, 2016; Gage, 1994)
  - WordPiece (Schuster and Nakajima, 2012)
  - Unigram LM tokenization (Kudo, 2018)

BPE Encoding

Algorithm 1 Byte-pair encoding (Sennrich et al., 2016; Gage, 1994)

```
Input: set of strings D, target vocab size k
procedure BPE(D, k)
    V ← all unique characters in D
          (about 4,000 in English Wikipedia)
    while |V| < k do                    ▷ Merge tokens
        t_L, t_R ← Most frequent bigram in D
        t_NEW ← t_L + t_R               ▷ Make new token
        V ← V + [t_NEW]
        Replace each occurrence of t_L, t_R in
            D with t_NEW
    end while
    return V
end procedure
```

D = {‘l o w </w>’: 9,
‘l o w e r </w>’: 2,
’n e w e s t </w>’: 6,
‘w i d e s t </w>’: 3}

V = {l, o, w, e, r, …. </w>}

Most frequent bigram= ‘lo’: 11

V’ = {l, o, lo, w, e, r, …. </w>}

D’ = {‘lo w </w>’: 9,
‘lo w e r </w>’: 2,
’n e w e s t </w>’: 6,
‘w i d e s t </w>’: 3}

Other tokenizers?

- **WordPiece**
  - Google never open-sourced their implementation, but based on literature:
  - Merge criterion:

$t_L, t_R \leftarrow$ Most frequent bigram in D

$t_{\text{NEW}} \leftarrow t_L + t_R$ ▷ Make new token

score=freq($t_L t_R$)/ [freq(t_L) x freq(t_R)]

  - Prioritizes merging where individual chunks are less likely in vocab.

- **Unigram LM tokenizer**
  - Instead of greedy bottom-up approach, top-down.
  - Finds chunks which makes the sequence more likely under the unigram LM

Comparisons

Original:	furiously	**Original:**	tricycles	**Original:**	nanotechnology
BPE:	_fur ously	**BPE:**	_t ric y cles	**BPE:**	_n an ote chn ology
Uni. LM:	_fur ious ly	**Uni. LM:**	_tri cycle s	**Uni. LM:**	_nano technology

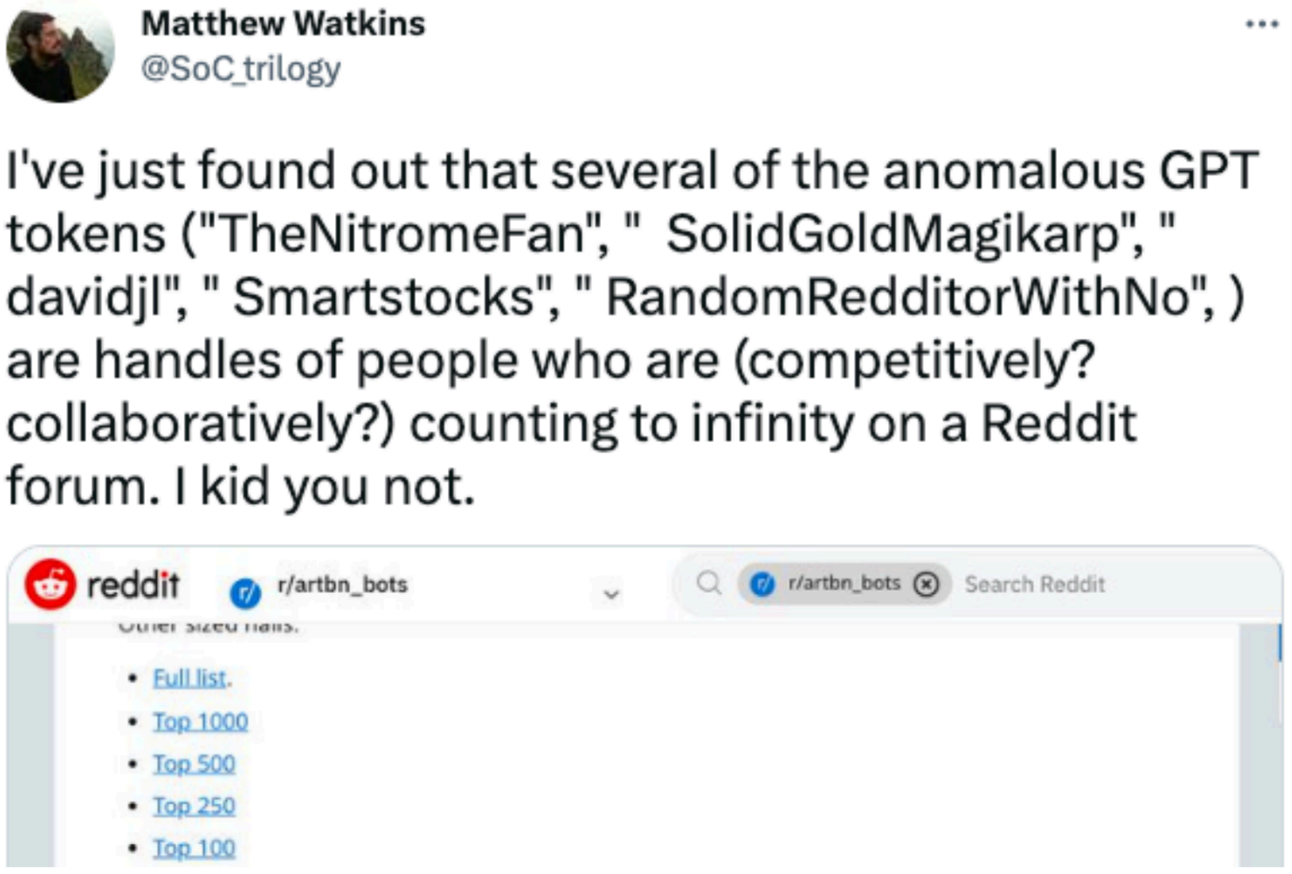
Original:	Completely preposterous suggestions
BPE:	_Comple t ely _prep ost erous _suggest ions
Unigram LM:	_Complete ly _pre post er ous _suggestion s

Original:	corrupted	**Original:**	1848 and 1852,
BPE:	_cor rupted	**BPE:**	_184 8 _and _185 2,
Unigram LM:	_corrupt ed	**Unigram LM:**	_1848 _and _1852 ,

- BPE tokens may not be semantically meaningful.
- Unigram LM leads to slightly better BERT (Bostrom and Durrett 2020)

GPT-3 vocab?

- BPE tokenizer
- GPT-3: 50K vocab size



Rank	User	Counts
1	/u/davidjl123	163477
2	/u/Smartstocks	113829
3	/u/atomicimploder	103178
4	/u/TheNitromeFan	84581
5	/u/SolidGoldMagikarp	65753
6	/u/RandomRedditorWithNo	63434
7	/u/rideride	59024
8	/u/Mooraell	57785
9	/u/Removedpixel	55080
10	/u/Adinida	48415
11	/u/rschaosid	47132

Slide credit: Greg Durrett

Are all languages treated equally?

- Will a sentence in English and Spanish having same number of words have similar number of tokens? Think of how BPE vocabulary is constructed…
- Language premium for A relative to B = $|t(s_A)|/|t(s_B)|$
- Flores-200 — contains translations of 842 distinct web articles, totaling 3001 sentences.
- Premiums wrt to English:
  - Chinese costs 91% more tokens.
  - Spanish costs 55% more.

Premiums with respect to English

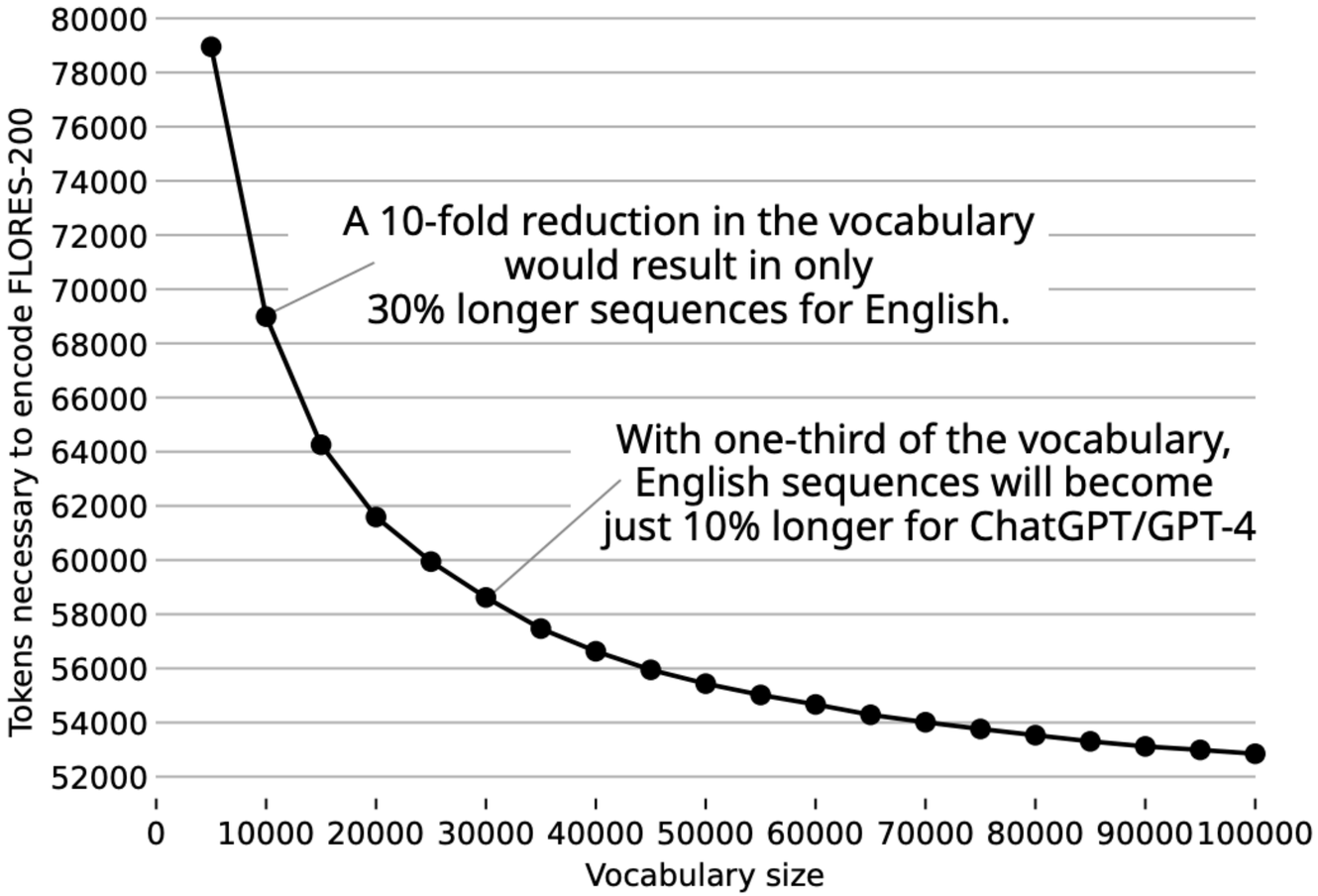
	GPT-2 RoBERTa	ChatGPT GPT-4	FlanT5
Bulgarian	5.51	2.64	—
Burmese	16.89	11.70	—
Chinese (Simplified)	3.21	1.91	—
Dzongkha	16.36	12.33	—
English	1.00	1.00	1.00
French	2.00	1.60	1.60
German	2.14	1.58	1.37
Italian	2.01	1.64	2.18
Japanese	3.00	2.30	—
Jingpho	2.65	2.35	3.41
Maori	2.45	2.35	3.28
Norwegian Bokmål	1.86	1.56	2.24
Odia	13.38	12.48	—
Pangasinan	1.66	1.57	2.18
Portuguese	1.94	1.48	2.21
Romanian	2.48	1.88	1.50
Santali	12.86	12.80	—
Shan	18.76	15.05	—
Spanish	1.99	1.55	2.23
Standard Arabic	4.40	3.04	—
Tumbuka	2.78	2.57	3.29
Vietnamese	4.54	2.45	—

Petrov et al, 2023

Are all languages treated equally?

- Fairness implications?
  - Cost
  - Latency
  - Long context processing

- How much longer will English language tokenization be if we dedicate a fraction of English tokens to other languages?



cl100k_base tokenizer used by ChatGPT, GPT-4

Petrov et al, 2023