CS 489/698: Introduction to Natural Language Processing

Lecture 2: Words and Morphology

Instructor: Freda Shi

fhs@uwaterloo.ca

January 7th, 2026

UNIVERSITY OF WATERLOO

What is a word?

Lexical Semantics

"A single distinct meaningful element of speech or writing, used with others (or sometimes alone) to form a sentence and typically shown with a space on either side when written or printed."

[Source: Oxford Languages]

What is a word?

Things in dictionaries?

"One of the most prolific areas of change and variation in English is vocabulary; new words are constantly being coined to the name or describe new inventions or innovations, or to better identify aspects of our rapidly changing world… Most general English dictionaries are designed to include only those words that meet certain criteria of usage across wide areas and over extended periods of time…"

[Source: Merriam-Webster; https://www.merriam-webster.com]

What is a word?

Things between spaces and punctuation?

This is English: The cat is cute.

This is French: Le chat est mignon.

This is Spanish: El gato es lindo.

This is Chinese: 猫很可爱。

This is Japanese: 猫はかわいい。

This is Thai: แมวน่ารักจัง.

What is a word?

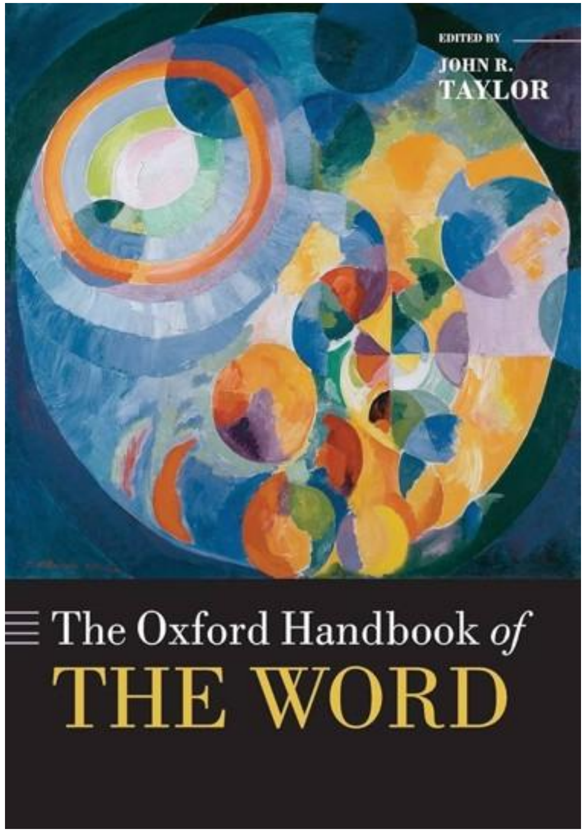
Smallest unit that can be uttered in isolation?

- You could utter this word in isolation: *unimpressively*.
- Also this one: *impress*.
- Probably also these when you talk about morphology: *un*, *ive*, *ly*.

Are they all words?

What is a word?

Each of the above points captures some, but very likely not all aspects of what a word is.



42 chapters.

Nearly 900 pages.

Covers a lot of aspects of what makes a word word, "to anyone who shares a fascination with words."

Outline of Today's Lecture

Linguistic morphology

- The study of internal structures of words.

Lexical semantics

- The study of meanings of words.

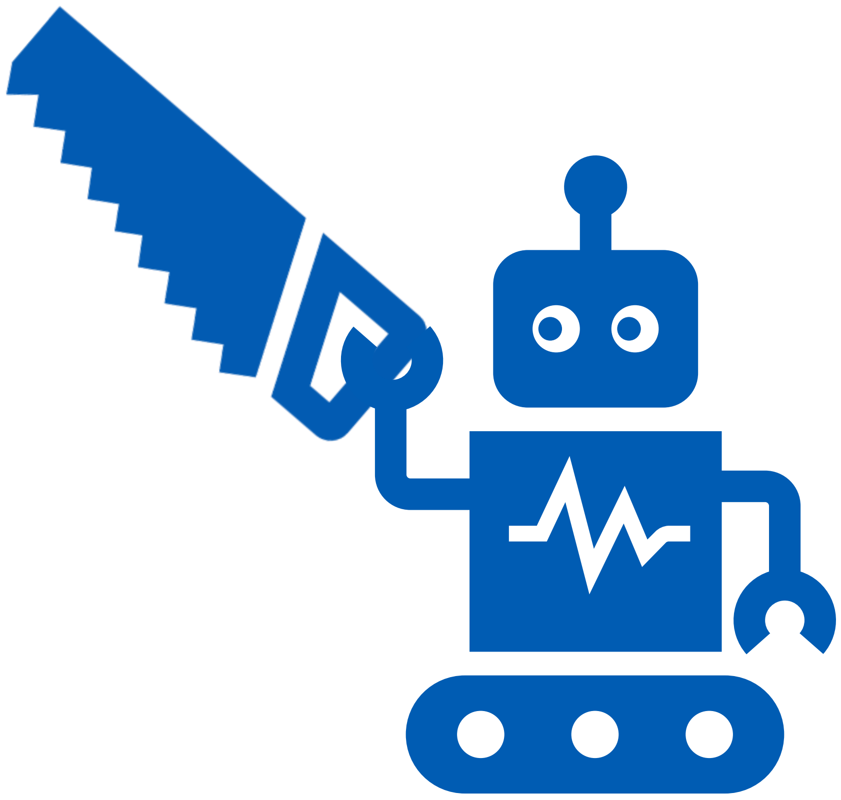
Word tokenization

- The process of splitting texts into "words" (tokens).

colder

replayed

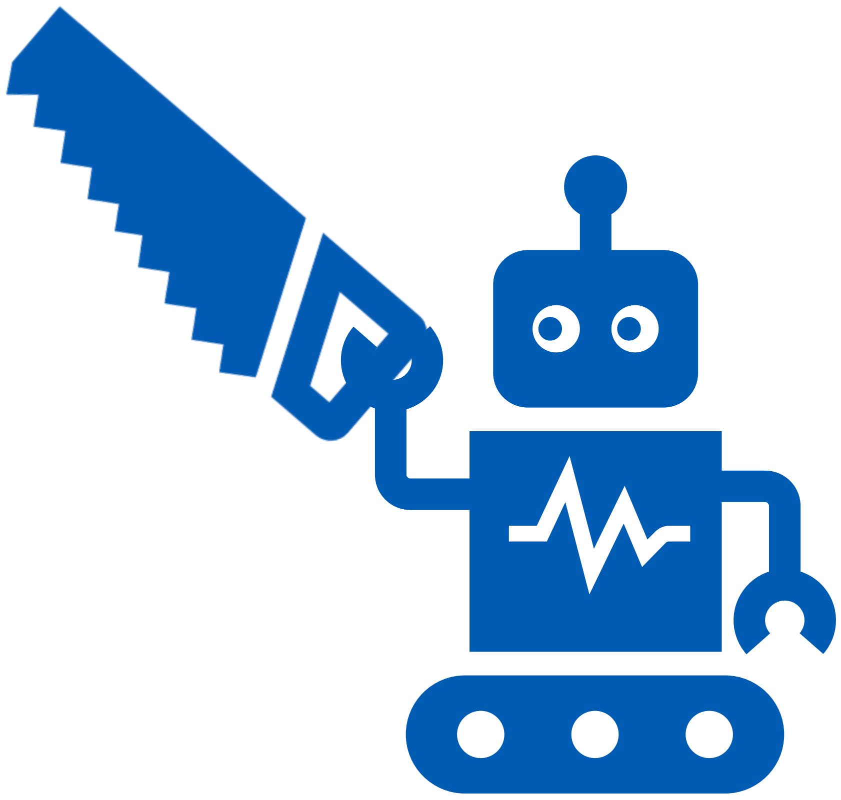
gameplay



cold|er

re|play|ed

game|play



Morphology

The study of how words are built from smaller meaning-bearing units.

Types of morphemes:

- Stem: a core meaning-bearing unit.
- Affix: a piece that attaches to a stem, adding some function or meaning.

  Prefix: unhappy, predefine

  Suffix: cats, walked

  Infix: (Malay) Gigi (*teeth*) → Gerigi (*tooth blade*)

  Circumfix: (German) mach (root of *machen; to make*) → gemacht (*made; past participle*)

  Interfix: speedometer

[See more in Chap. 6.2 of Doner. The Linguistic analysis of word and sentences structures]

Types of Word Formation

Inflection: adding morphemes to a word to indicate grammatical information.

- *walk → walked*
- *cat → cats*

Derivation: adding morphemes to a word to create a new word with a different meaning.

- *happy → happiness*
- *define → predefine*

Compounding: combining two or more words to create a new word.

- *key + board → keyboard*
- *law + suit → keyboard*
- *book + case → bookcase*

Isolating Language

In languages like Classical Chinese, Vietnamese, and Thai

- Each word form typically consists of one single morpheme.
- There is little morphology other than compounding.

➢Inflection

们: 我们, 你们, 他们

mén: wǒmén, nǐmén, tāmén

plural: we, you (pl.), they

➢Derivation

家: 艺术家

jiā: yì shù jiā, artist

➢Compound

Morphological Decomposition

Usually, morphological decomposition is simply splitting a word into its morphemes:

walked = walk + ed

greatness = great + ness

But it can actually be a hierarchical structure:

unbreakable = un + (break + able)

internationalization = (((inter + nation) + al) + iz[e]) + tion

There is ambiguity in hierarchical decomposition!

The door is unlockable.

Morphology in NLP

Individual tasks that address morphology:

- Lemmatization: putting words/tokens in a standard format.
  - Lemma: canonical/dictionary form of a word.
  - Wordform: fully inflected or derived form of a word as it appears in text.

wordform	lemma
run	run
ran	run
running	run

Morphology in NLP

Individual tasks that address morphology:

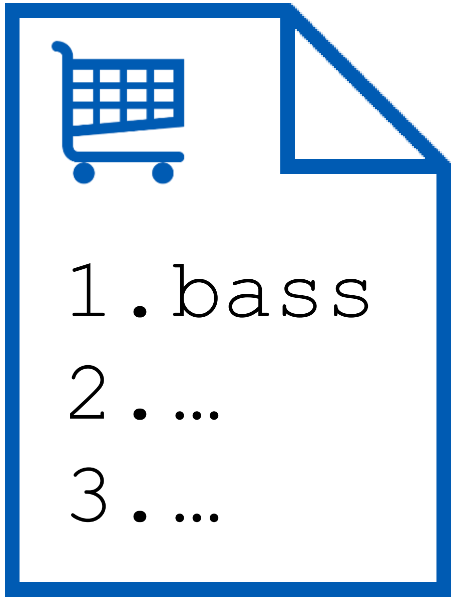
- Stemming: reducing words to their stems (approximately) by removing affixes. More conventional engineering-oriented approach used in applications such as retrieval.

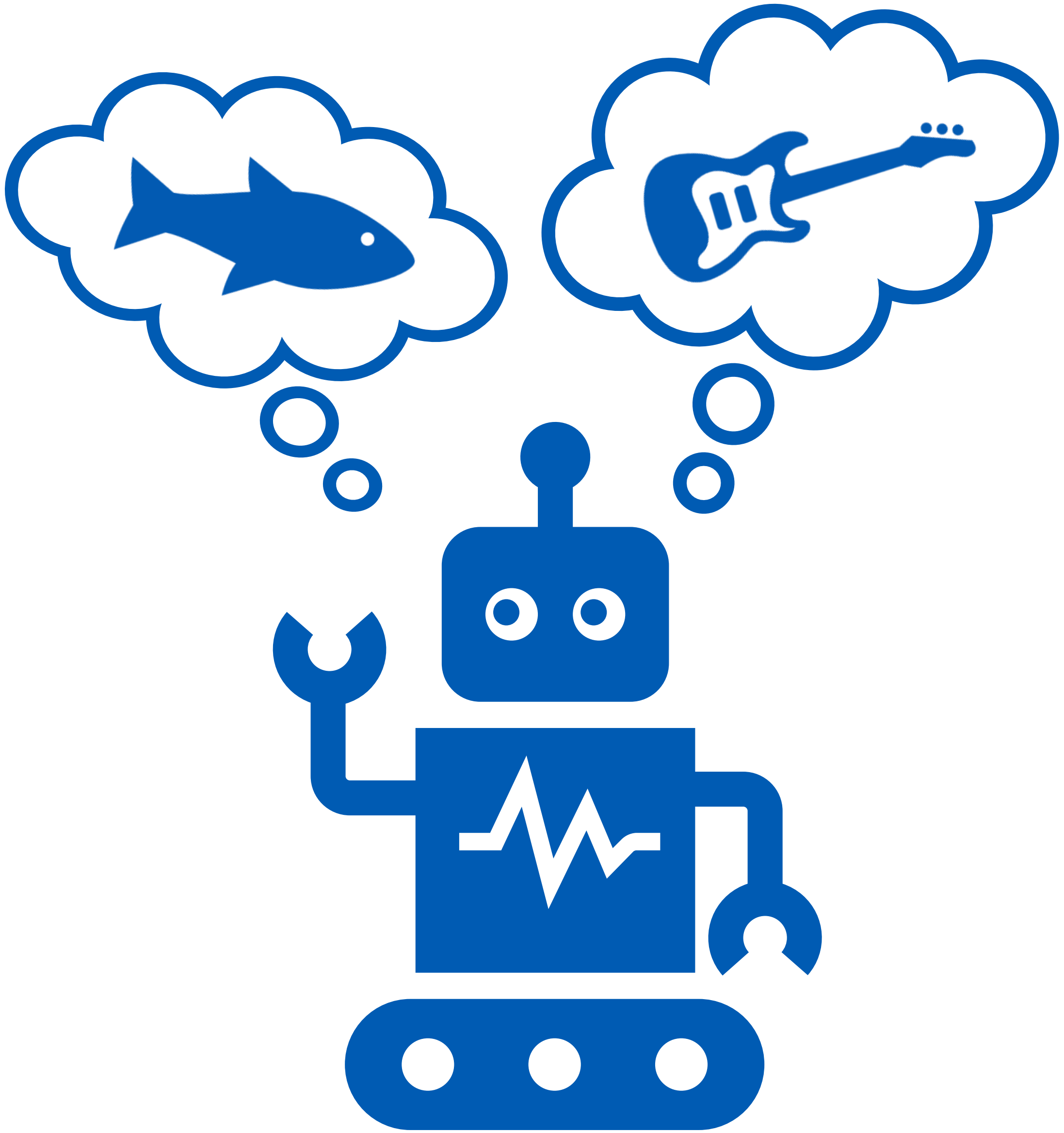
Caillou is an average, imaginative four-year-old boy with a love for forms of transportive machinery such as rocket ships and airplanes.

↓

Caillou is an **averag imagin** four year old **boi** with a love for **form** of **transport machineri** such as rocket **ship** and **airplan**

Lexical Semantics

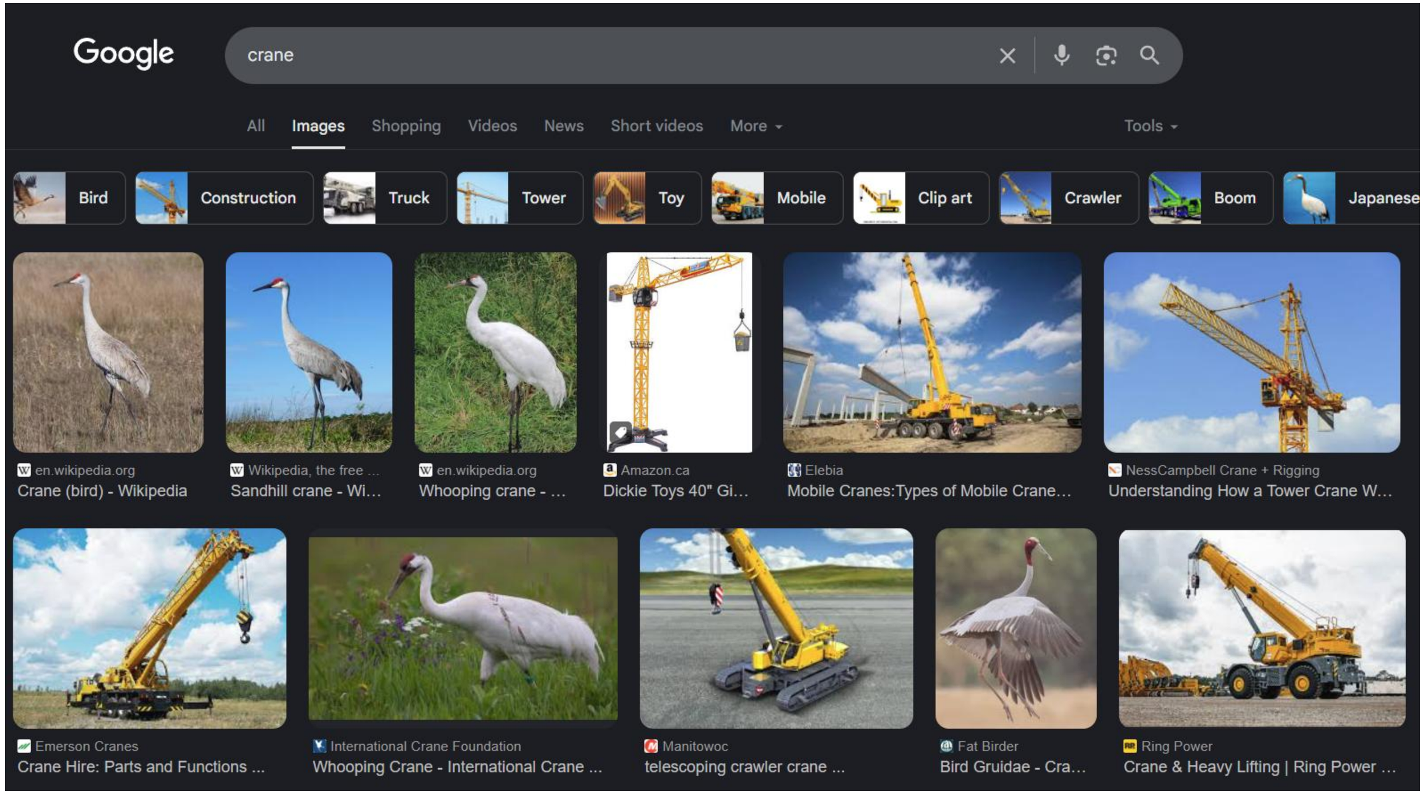




Variability and Ambiguity in Words

Lemmatization and stemming tackles the problem of variability---multiple forms could share the same or similar meanings.

On the other hand, one wordform could refer to multiple meanings.



Polysemy vs. Homonymy

- Polysemy: a word has multiple related meanings.
  - She is a star.
  - (pointing to the sky) The star is shining.
- Homonymy: a word has multiple meanings originated from different sources.
  - I need to go to the bank as I don’t have enough cash.
  - I am sitting on the bank of the river.

Question: which one do you think is the case for crane?

Synonyms

Synonyms (informal definition): words that have the same meanings according to some criteria.

- *couch vs. sofa*
- *big vs. large*
- *water vs.* H_2O

- There are very few (or no) examples of perfect synonymy.
- Synonymy is a relation between senses rather than words.

- *How big is the plane?*
- *How large is the plane?*

- *Miss Nelson became a kind of big sister to Benjamin.*
- *Miss Nelson became a kind of large sister to Benjamin. (*)*

Antonyms

Antonyms: senses that are opposite with respect to (at least) one dimensionality of meaning.

- *dark and light*
- *dark and bright*
- *hot and cold*
- *in and out*

Hyponymy/Hypernymy, and Meronym/Holonym

- Sense A is a hyponym of sense B if A is more specific, denoting a subclass of B.
- Conversely, B is a hypernym of A.
  - *dog* is a hyponym of *animal*
  - *corgi* is a hyponym of *dog*

- Sense A is a meronym of sense B if A is a part of B.
- Conversely, B is a holonym of A.
  - *hand* is a meronym of *body*
  - Finger is a meronym of *hand*

The WordNet database: https://wordnet.princeton.edu

Word Sense Disambiguation

- Word-Sense Disambiguation (WSD): the task of determining which sense of a word is used in a particular context, given a set of predefined possible senses.
- Relatedly, word sense induction (WSI) requires clustering word usages into senses without predefined ground truths.

Default solution (as of 2026): encode the context of words with a pretrained model, and train a neural network to predict the sense.

Or… prompting a pretrained language model.

The Role of Word Senses

A practical question: We now have powerful neural language models, which do not distinguish word senses. Is WSD still a meaningful task?

A philosophical question in lexical semantics: Do discrete word senses even “exist”?

[Li. 2024. *Semantic minimalism and the continuous nature of polysemy*. Mind and Language]

Tokenization

- Tokenization: the process that converts running text (i.e., a sequence of characters) into a sequence of tokens.

```
“Oh!” said Lydia stoutly, “I am not afraid; for
though I _am_ the youngest, I’m the tallest.”
```

tokenizer

```
“ Oh ! ” said Lydia stoutly , “ I am not afraid ; for
though I _ am _ the youngest , I ’m the tallest . ”
```

Conventions in Rule-Based Tokenizers

	Penn Treebank	Moses
don’t	do n’t	don ’t
aren’t	are n’t	aren ’t
can’t	ca n’t	can ’t
won’t	wo n’t	won ’t

It is important to check and ensure consistency when comparing results across different tokenizers.

See `nltk.tokenize`, which also works for sentence tokenization.

[https://www.nltk.org/api/nltk.tokenize.html]

Tokenization across Languages

There is no explicit whitespace between words in some languages, and tokenization becomes highly nontrivial in these cases.

姚明 进入 总决赛 “YaoMing reaches finals”	Chinese Treebank
姚 明 进入 总 决赛 “Yao Ming reaches overall finals”	Peking University

Word Types vs. Word Tokens

```
“ oh ! ” said lydia stoutly , “ i
am not afraid ; for though i _ am _
the youngest , i ’m the tallest . ”
```

3	i	1	!	1	oh
2	,	1	.	1	said
2	_	1	;	1	stoutly
2	am	1	afraid	1	tallest
2	the	1	for	1	though
2	“	1	lydia	1	youngest
2	”	1	not	1	’m

Word Types vs. Word Tokens

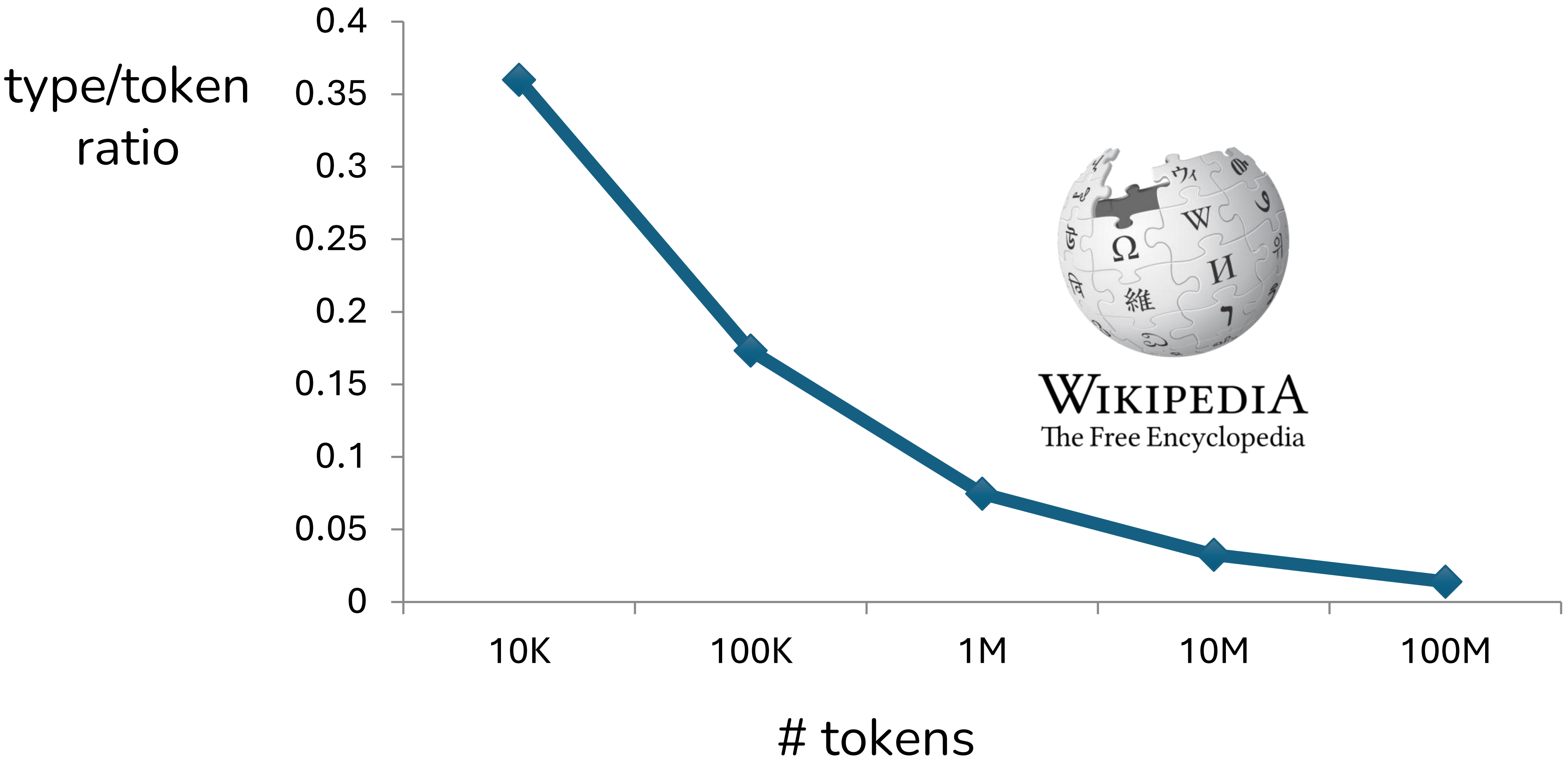
3	i	1	!	1	oh
2	,	1	.	1	said
2	_	1	;	1	stoutly
2	am	1	afraid	1	tallest
2	the	1	for	1	though
2	“	1	lydia	1	youngest
2	”	1	not	1	’m

Type: a unique word (an entry in a vocabulary or dictionary) – 21 types.

Token: an instance of a type in the text – 29 tokens.

Type/Token Ratio

How does the type/token ratio change when adding more data?



Type/Token Ratio: Wikipedia vs. Twitter

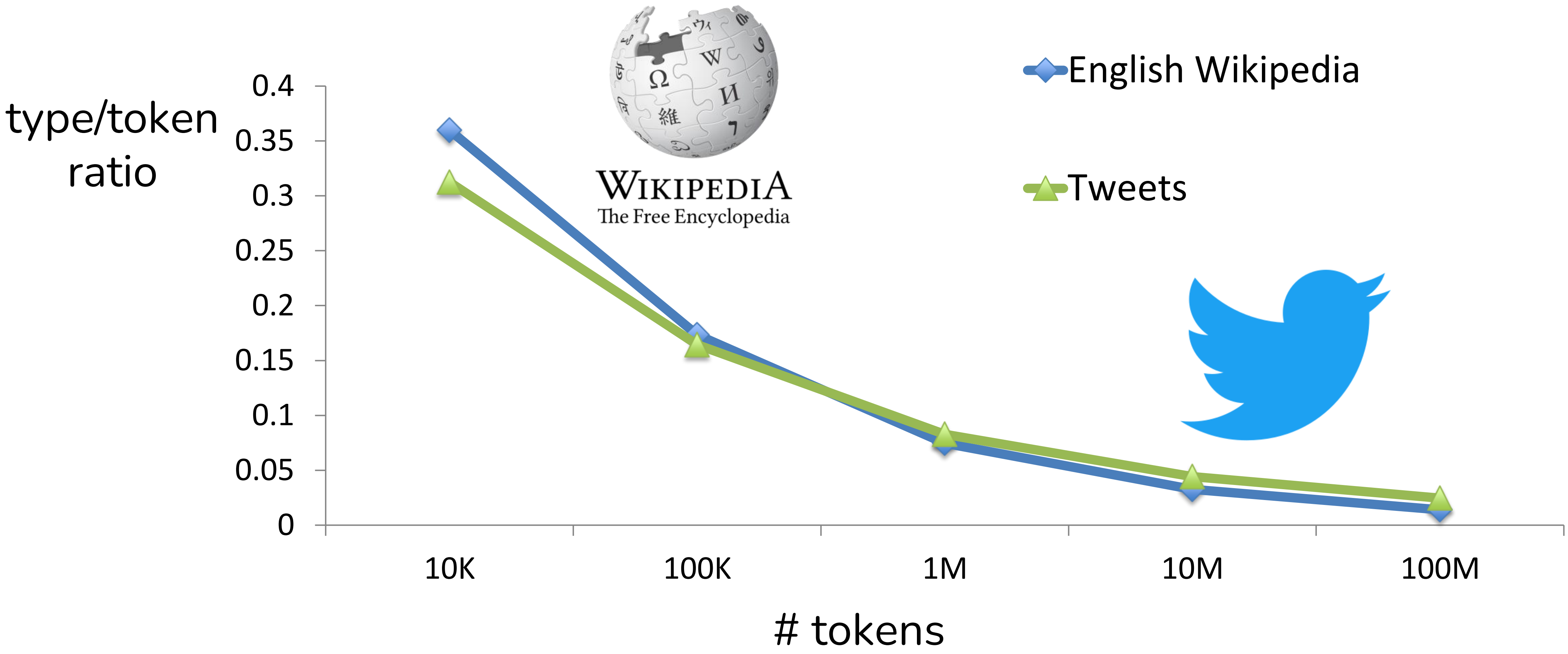
How do the type/token ratio curves compare between Wikipedia and Twitter?

WIKIPEDIA
The Free Encyclopedia

vs.

Type/Token Ratio: Wikipedia vs. Twitter

How do the type/token ratio curves compare between Wikipedia and Twitter?



224571	really	38	really2	12	reaaaaaally
1189	rly	37	reaaaaally	12	rreally
1119	realy	35	reallyyyyy	11	reaallyy
731	rlly	31	reely	11	realllllyyy
590	reallly	30	realllyyy	11	reeeallly
234	realllly	27	reaaly	11	reeeeallly
216	reallyy	27	realllyy	10	reaaaly
156	relly	26	realllyyyy	10	reallyreallyreally
146	reallllly	25	realllllllly	9	r)eally
132	rily	22	reaaallly	9	really-really
104	reallyyy	21	really-	9	reallys
89	realllllly	19	reeaally	9	reeeeeeeally
89	reeeally	18	reallllyyy	8	realky
84	reaaally	16	reaaaallly	8	reallyyyyyy
82	reaally	15	reaallly	8	reallyyyyyyy
72	reeeeally	15	reallllllllly	8	reeeaally
65	reaaaally	15	reallllyy	7	r3ally
57	reallyyyy	15	reallyreally	7	raelly
53	rilly	15	realyy	7	reaaaaaaally
50	reallllllly	14	reallllyyyy	7	realllllllllllly
48	reeeeeally	14	reeeeeeally	7	reallllllyyy
41	reeally	13	reeeaaally	7	reeeeaaally

7 reeeealy
7 reeeeeeeeeally
7 relaly
6 r-e-a-l-l-y
6 r-really
6 reaaaaaallly
6 realllllllllly
6 realllyyyyy
6 realyl
6 reeeaaaally
6 reeeaaallly
6 reeeaaalllyyy
5 reaaaaallly
5 reaaaallll y
5 reaalllyy
5 realllllllllllllly
5 reallllllllllly
5 reeallyyy
5 reeeeaaallly
5 reeeeaally
5 reeeeeeeeally
5 rellly
5 rrly
5 rrrreally
4 reaaaaly
4 reaaalllly
4 reaaalllyy
4 reaalllly
4 reaalllyyy
4 realllllllyyyy
4 realllllyyyy
4 reeaaaally
4 reeealy
4 reeeeeeeeeeally
4 rllly
3 r34lly
3 r]eally
3 reaaaaaaaally
3 reaaaaaly
3 reaaaalllll y
3 reaaaallyy
3 reaaallyy
3 reaallllly
3 reaallyyyy
3 realiy
3 reallllllllllllllllly
3 realllllllyy
3 realllllllyyyy
3 realllllllyyyyyyy
3 reallllyyyyy
3 realluy
3 really)
3 reallyl
3 reallyyyyyyyyy
3 reeaaallly
3 reeaalllly
3 reeaalllyyy
3 reeaaly
3 reeallly
3 reealy
3 reeeaaallllyyy
3 reeeaallly
3 reeeeaaaaally
3 reeeealllly
3 reeeeealllly
2 reaaaaaaaaaally

2 reaaaaaaaaally
2 reaaaaaaaallly
2 reaaaaaalllllyyy
2 reaaaaallllly
2 reaaaaalllly
2 reaaalllllyyy
2 reaaallllyyy
2 reaaalllyyy
2 reaallllyy
2 reaallllyyy
2 reaallyyy
2 reaalyy
2 reallllllllllllllllly
2 reallllllllllllllly
2 reallllllllyy
2 reallllllllyyyyy
2 realllllllyyyyy
2 realllllyy
2 realllllyyyyyy
2 reallllyyyyyy
2 realllyyyyyy
2 really*
2 really/
2 reallyyyyyyyy
2 reallyyyyyyyyyyyy
2 realyyy
2 reaqlly
2 reeaaally
2 reeaallly
2 reeaalllyy
2 reeaallyy
2 reeallyy
2 reeeallyy
2 reeeeaaaalllyyy
2 reeeeaaaally
2 reeeeaaalllly
2 reeeeaaallllyyyy
2 reeeeallllyyy
2 reeeeallyyy
2 reeeeeaaallllly
2 reeeeeaaally
2 reeeeeaally
2 reeeeeallly
2 reeeeeealy
2 reeely
2 rellys
2 rellyy
2 reqally
2 rlyyy
2 rlyyyy
2 rreeaallyy
2 rrreally
1 r-r-r-really
1 r3aly
1 r3ly
1 raaahhhlllaaayyyy
1 raeally
1 re-e-e-eally
1 re-eaaaaaaaly
1 re-he-he-he-ealy
1 re-he-he-heeeeally
1 re3ally
1 rea(l)ly
1 reaaaaaaaaaaaaaaaaaaally
1 reaaaaaaaaaaaaaaaally
1 reaaaaaaaaaaaaaaalllllly

1 reaaaaaaaaaaaaaally
1 reaaaaaaaaaaaally
1 reaaaaaaaaaaalllllly
1 reaaaaaaaaaaallllly
1 reaaaaaaaaaaally
1 reaaaaaaaaaalllllllyyyyyyyyy
1 reaaaaaaaaalllly
1 reaaaaaaaallllllyyyyy
1 reaaaaaaaallllly
1 reaaaaaaaallllyyyy
1 reaaaaaallllly
1 reaaaaaalllyyyy
1 reaaaaalllllllyyyy
1 reaaaaalllllyyy
1 reaaaaalllllyyyy
1 reaaaaalllllyyyyy
1 reaaaaalllllyyyyyyy
1 reaaaaallllyy
1 reaaaaallllyyyyy
1 reaaaaallyy
1 reaaaalllllly
1 reaaaallllllyy
1 reaaaalllllyyy
1 reaaaalllllyyyyy
1 reaaaallllyyy
1 reaaaallllyyyyy
1 reaaaalllyy
1 reaaaalllyyy
1 reaaaalllyyyy
1 reaaaallyyy
1 reaaalllllllly
1 reaaalllllllyyy
1 reaaallllly
1 reaaalllllyyyy
1 reaaalllllyyyyy
1 reaaalllyyyyy
1 reaaallyyy
1 reaaallyyyy
1 reaalllllly
1 reaalllllyyyy
1 reaallllyyyy
1 reaalllyyyy
1 reaalllyyyyy
1 reaalllyyyyyy
1 real(ly
1 realaaay
1 realkly
1 reall(y
1 reallhy
1 reallllllllllllllllllllllllllllllllll
llllllllllllllllllllllllllllllllllly
1 reallllllllllllllllllllllllllllllllll
llllllllllllllllllllly
1 reallllllllllllllllllllllllllllllllll
llllllllllly
1 realllllllllllllllllllllllllllllllllly
1 reallllllllllllllllllllllllllllly
1 realllllllllllllllllllllllllllly
1 reallllllllllllllllllllllyyyyy
1 realllllllllllllllllllllly
1 realllllllllllllllllllly
1 reallllllllllllyyyyyy
1 realllllllllllyyyyy
1 reallllllllllyyy
1 reallllllllllyyyy
1 reallllllllllyyyyy
1 reallllllllllyyyyyy
1 realllllllllyyyy
1 realllllllllyyyyyyy
1 realllllllllyz
1 realllllyly
1 realllllyyyyy
1 reallllyyyyyyy
1 reallllyyyyyyyyy
1 realllylyyy
1 realllyyyyyyy
1 realllyyyyyyyyy
1 really(really
1 really//
1 really/really/really
1 really100
1 really2x
1 really:')
1 really^^
1 really_
1 reallyii
1 reallyreallyreallyreallyreallyr33lly
1 reallyreallyreallyreallyreallyreally
reallyreallyreallyreallyreallyreally
reallyreallyreallyreallyreally

1 reallyyyyyyyyyyy
1 really😩
1 realoly
1 realys
1 realyyyyy
1 real•ly
1 reawly
1 ree-hee-heally
1 reeaaaaaaaaaaaaaaaaalllllllllyyy
1 reeaaaaaaaallllllllly
1 reeaaaaallllllly
1 reeaaaalllly
1 reeaaalllly
1 reeaaallllyyy
1 reeaaallyyy
1 reeaaaly
1 reeaalllllyyy
1 reeaallllyyy
1 reeaallyyy
1 reeallllly
1 reealllyyy
1 reealllyyyyy
1 reeeaaaaaaaly
1 reeeaaaaally
1 reeeaaaalllllly
1 reeeaaaallllyyy
1 reeeaaaallllyyyy
1 reeeaaaallly
1 reeeaaaalllyyy
1 reeeaaaaly
1 reeeaaalllllyyyy
1 reeeaaallllyyyy
1 reeeaaallyy
1 reeeaaallyyy
1 reeeaaaly
1 reeeaallllly
1 reeeaallyy
1 reeeaallyyy
1 reeealllllly
1 reeeallllly
1 reeeallllyy
1 reeeallllyyy
1 reeeallllyyyy
1 reeealllyyy
1 reeealllyyyy
1 reeeallys
1 reeeeaaaaaaallllllyyyyyyy
1 reeeeaaaaaallllllyyyyy
1 reeeeaaaaalllllllly
1 reeeeaaaaallllly
1 reeeeaaaaalllly
1 reeeeaaaaalllyyy
1 reeeeaaaallllly
1 reeeeaaaallllyyyyyy
1 reeeeaaallllyy
1 reeeeaaalllyy
1 reeeeaaallyyy
1 reeeeaaaly
1 reeeeaallly
1 reeeeaalllyy
1 reeeeaaly
1 reeeealllllly
1 reeeeallllyy
1 reeeealllyy
1 reeeealllyyy
1 reeeealllyyyy
1 reeeeeaaaaalllllllly
1 reeeeeaaaaally
1 reeeeeaaallllly
1 reeeeeaaallllyyy
1 reeeeeaaallly
1 reeeeeallllyyy
1 reeeeealy
1 reeeeeeaaaaaally
1 reeeeeeaaaaalllllyyyy
1 reeeeeeaaaaally
1 reeeeeeaaaallllly
1 reeeeeeaally
1 reeeeeeaaly
1 reeeeeealllllly
1 reeeeeealllyyyy
1 reeeeeeeaaallllly
1 reeeeeeeaally
1 reeeeeeeallyy
1 reeeeeeealy
1 reeeeeeeeaaaaaaaalllllyyyyyy
1 reeeeeeeeeaaaaaaaaalllllllllyyyyyyyy
1 reeeeeeeeeaaaaaaallllllllyyyyyyyy
1 reeeeeeeeeaaaaaalllyyy
1 reeeeeeeeeaaally
1 reeeeeeeeeaally
1 reeeeeeeeealllllly
1 reeeeeeeeeeaaally

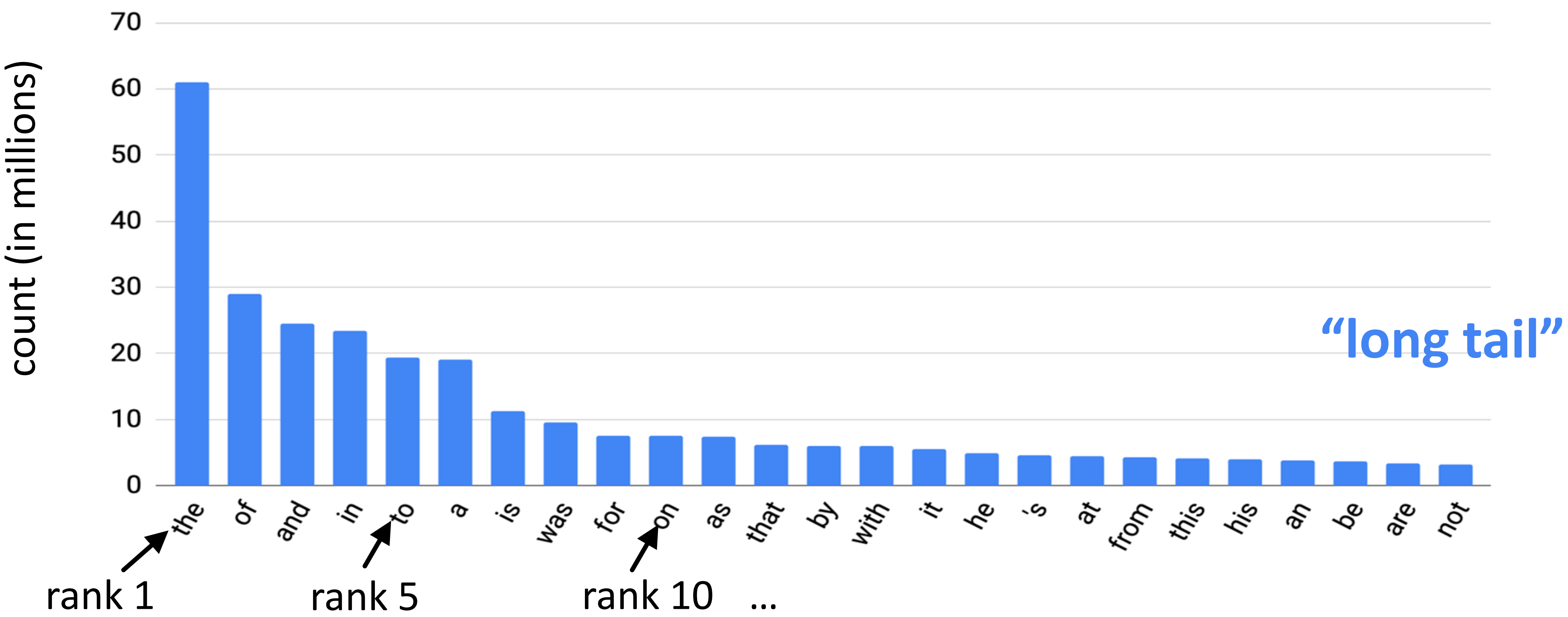
1 reeeeeeeeeeaally
1 reeeeeeeeeeeally
1 reeeeeeeeeeeeally
1 reeeeeeeeeeeeeaaally
1 reeeeeeeeeeeeeeeaaaallly
1 reeeeeeeeeeeeeeeeally
1 reeeeeeeeeeeeeeeeeeeeeesallllllllllllllllllllllllllly
1 reeeeeeelly
1 reeeeeely
1 reeeeelly
1 reeeeely
1 reeeelllllyyy
1 reeellllyy
1 reellly
1 reelllyy
1 reheheally
1 relally
1 rellllllly
1 relllly
1 rellyrell
1 rellzy
1 rieeely
1 rlllllly
1 rllllllyy
1 rlllly
1 rllyrlly
1 rllyy
1 rllyyy
1 rlyrlyrly
1 rlyy
1 rraarreellyy
1 rreaalllyyy
1 rreaally
1 rreeaaallllyyyy
1 rreeaallllyy
1 rreeaallyyy
1 rreealy
1 rreeeaaaaallllyyyy
1 rreeeaaallllyyy
1 rreeeeeeaaaaallllllllyyyyyy
1 rreeeeeeallly
1 rreeeeeely
1 rrreallyyy
1 rrreeeaaalllyyy
1 rrreeealllyyy
1 rrreeeeaalllllyy
1 rrreeeeallly
1 rrrlyyy
1 rrrreeeally
1 rrrreeeeeeaaaaallllllllyyyyyy
1 rrrrreeeeaaalllly
1 rrrrreeeeaaalllyy
1 rrrrrreally
1 rrrrrrealy
1 rrrrrrreeeeeeaaaalllllyyyyyyy
1 rrrrrrrrrreally
1 rrrrrrrrrrrrrrrreeeeeeeeeeeaaaaaaalllllllyyyyyy

How are words distributed?

224571	really	1	rreeeeeeaaaallllllllyyyyyy
1189	rly	1	rreeeeeeallly
1119	realy	1	rreeeeeely
731	rlly	1	rrreallyyy
590	reallly	1	rrreeeaaalllyyy
234	realllly	1	rrreeealllyyy
216	reallyy	1	rrreeeeaalllllyy
156	relly	1	rrreeeeallly
146	reallllly	1	rrrlyyy
132	rily	1	rrrreeeally
104	reallyyy	1	rrrreeeeeeaaaaallllllllyyyyyy
89	realllllly	1	rrrrreeeeaaalllly
89	reeeally	1	rrrrreeeeaaalllyy
84	reaaally	1	rrrrrreally
82	reaally	1	rrrrrrealy
72	reeeeally	1	rrrrrrreeeeeeaaaalllllyyyyyyy
65	reaaaally	1	rrrrrrrrrreally
…		1	rrrrrrrrrrrrrrrreeeeeeeeeeeaaaaaaalllllllyyyyyy

Zipf's Law

Frequency of a word is (roughly) inversely proportional to its rank in the word frequency list.

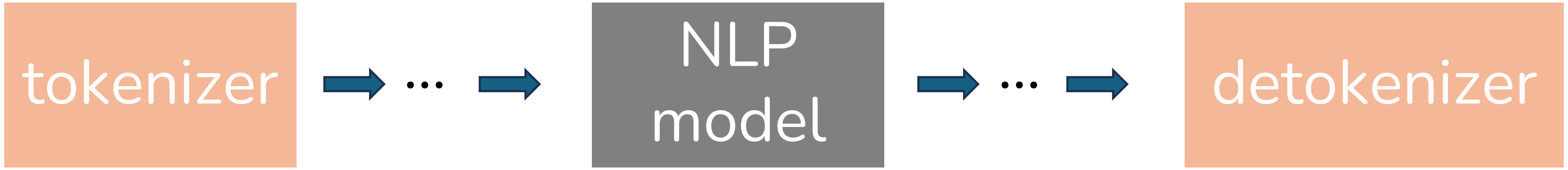


Tokenization in Modern NLP Systems

There are so many word types, but the words have shared internal structures and meanings (recall what we've talked about in morphology).

Modern NLP systems always convert tokens into numerical indices for further processing. Can we do better than assigning each word a unique index?

Raw text input



Raw text output

Data-Driven Subword-Based Tokenizers

Data-driven tokenizers offer an option that learns the tokenization rules from data, tokenizing texts into subword units (a.k.a, wordpieces) using statistics of character sequences in the dataset.

Two most popular methods:

- Byte Pair Encoding (BPE): Gage (1994), Sennrich et al. (2016).
- SentencePiece: Kudo (2018).

[Gage, P. (1994). A new algorithm for data compression. *The C Users Journal*, *12*(2), 23-38.]
[Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of ACL* (pp. 1715-1725).]
[Kudo, T. (2018). Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. In *Proceedings of ACL* (pp. 66-75).]

Byte Pair Encoding

Originally introduced by Gage (1994) for data compression, and later adapted (and revived) by Sennrich et al. (2016) for NLP.

Key idea: merge symbols with a greedy algorithm.

Initialize the vocabulary with the set of characters, and iteratively merge the most frequent pair of symbols to extend the vocabulary.

Training Corpus

```
c a t
c a t s
c o n c a t e n a t i o n
c a t e g o r i z a t i o n
```

Byte Pair Encoding

Training Corpus

c a t
c a t s
c o n c a t e n a t i o n
c a t e g o r i z a t i o n

Initial Vocabulary

a c e g i n o r s t z

Count Symbol-Pair Frequencies

<a t>: 6, <c a>: 4, <o n>: 3, …

Update Vocabulary

a c e g i n o r s t z at

Byte Pair Encoding

Training Corpus

c a t
c a t s
c o n c a t e n a t i o n
c a t e g o r i z a t i o n

Current Vocabulary

a c e g i n o r s t z at

Count Symbol-Pair Frequencies

<c at>: 4, <o n>: 3, …

Update Vocabulary

a c e g i n o r s t z at cat

Repeat this process until the vocabulary reaches the desired size.

Byte Pair Encoding

The BPE proposal is not optimal in terms of compression rate under the same vocabulary size.

Training Corpus

c a t
c a t s
c o n c a t e n a t i o n
c a t e g o r i z a t i o n

Terminate Vocabulary

a c e g i n o r s t z at cat

A Better Vocabulary

a c e g i n o r s t z cat on

Apply Trained BPE to New Corpus

In addition to having the tokens, we will also need to know the "merge rules."

Starting from individual characters, and merge following the rules.

Terminate Vocabulary

`a c e g i n o r s t z at cat`

Merge Rules

`a t → at`

`c at → cat`

Word to be tokenized

`c a t e g o r y`

Result

`cat e g o r y`

If there is unknown character, add a new term (unlikely in real practice).

SentencePiece Tokenization

Kudo (2018): find the vocabulary for a unigram language model that maximizes the likelihood of the training corpus.

$$P(\langle x_1, x_2, \ldots, x_n \rangle) = \prod_{i=1}^{n} P(x_i)$$

$$P(x_i) := \textit{optional-smoothing}\left(\frac{\text{count}(x_i)}{\sum_{x \in V} \text{count}(x)}\right)$$

The Google command-line toolkit that implements this algorithm and some others (including BPE): https://github.com/google/sentencepiece

Byte-Level BPE

A practical question: How do large language models tokenize texts from different languages, with a unified tokenizer and fixed vocabulary size?

That’s great 👍

54 68 61 74 2019 73 20 67 72 65 61 74 20 **1F44D**

All in hexadecimal!

Prepend zeros to fix the length of tokens (to ensure the unique decoding), and do BPE on the bit/multi-bit level.

Next

Word Embeddings