

CS 489: Introduction to Natural Language Processing

Lecture 2: Words: Tokenization and Morphology

Instructor: Freda Shi

September 15, 2026



UNIVERSITY OF
WATERLOO

Outline of Today's Lecture

Part I: What is a Word?

- Definitions and challenges across languages.

Part II: Morphology

- Morphemes, word formation, morphological typology.

Part III: Tokenization

- Rule-based tokenization; word types vs. word tokens; Zipf's law.

Part IV: Subword Tokenization

- Byte Pair Encoding (BPE) from scratch.
- WordPiece; Unigram LM tokenizer; Viterbi decoding.

Part V: Cross-lingual Considerations

- Vocabulary sharing, fertility, and low-resource scripts.

Readings: [Sennrich et al. \(2016\)](#); [Kudo \(2018\)](#).

Part I: What is a Word?

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 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]

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.

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.

Chinese: 这只猫很可爱。

Japanese: この猫はかわいい。

Thai: แมวน่ารัก

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.

Many languages (Chinese, Japanese, Thai) do not use spaces between words!

Chinese: 这只猫很可爱。

Japanese: この猫はかわいい。

Thai: แมวน่ารัก

What is a Word?

Smallest unit that can be uttered in isolation?

- You could utter this word in isolation: unimpressively.

What is a Word?

Smallest unit that can be uttered in isolation?

- You could utter this word in isolation: unimpressively.
- Also this one: impress.

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.

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?

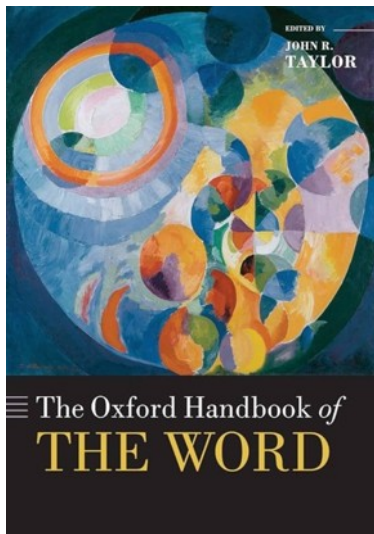
What is a Word?

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?

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



The Oxford Handbook of the Word
(Taylor, 2015)

42 chapters.

Nearly 900 pages.

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

Part II: Morphology

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 function or meaning.
 - Prefix: **un**happy, **pre**define
 - Suffix: cats**s**, walk**ed**
 - Infix: (Malay) gigi (teeth) → **ge**rigi (serration)
 - Circumfix: (German) mach → **ge**mach**t** (past participle)
 - Interfix: speed**o**meter

Types of Word Formation

Inflection: adding morphemes to indicate grammatical information.

- walk → walk~~ed~~
- cat → cat~~s~~

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

- happy → happi~~ness~~
- define → pre~~define~~

Types of Word Formation

Inflection: adding morphemes to indicate grammatical information.

- walk → walk**ed**
- cat → cat**s**

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

- happy → happi**ness**
- define → **pre**define

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

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

Morphological Typology: Isolating Languages

In languages like Classical Chinese, Vietnamese, and Thai:

- Words are often monomorphemic, and grammatical meanings are commonly expressed by separate words or particles.
- Compounding is an important source of multi-morpheme words.

Morphological Typology: Isolating Languages

In languages like Classical Chinese, Vietnamese, and Thai:

- Words are often monomorphemic, and grammatical meanings are commonly expressed by separate words or particles.
- Compounding is an important source of multi-morpheme words.

Compounding example (Mandarin):

高 + 地 = 高地

high + land = highland

高 + 速 = 高速

high + speed = high-speed

Morphological Typology: Isolating Languages

In languages like Classical Chinese, Vietnamese, and Thai:

- Words are often monomorphemic, and grammatical meanings are commonly expressed by separate words or particles.
- Compounding is an important source of multi-morpheme words.

Compounding example (Mandarin):

高 + 地 = 高地

high + land = highland

高 + 速 = 高速

high + speed = high-speed

Inflection is limited:

Grammatical distinctions such as plurality are often expressed by separate words, particles, or context rather than obligatory suffixes.

Derivation:

Often achieved through compounding rather than affixation.

Morphological Typology: Agglutinative Languages

In languages like Turkish, Finnish, and Swahili:

- Words are formed by concatenating morphemes, each with a single meaning.
- Morpheme boundaries are relatively clear.

Example (Turkish):

ev	house
evler	houses
evlerden	from houses
evlerdendir	it is from houses

Example (Finnish):

talossaniikin = “in my house too”

Morphological Typology: Fusional Features

Fusional morphology appears in languages such as Spanish and Russian, and in some parts of English and Arabic:

- A single morpheme can carry multiple grammatical meanings.
- Morpheme boundaries are often blurred.

Example (Spanish):

hablo = “I speak”

The suffix **-o** encodes first person, singular, present tense, and indicative mood, all at once.

Morphological Typology: Fusional Features

Fusional morphology appears in languages such as Spanish and Russian, and in some parts of English and Arabic:

- A single morpheme can carry multiple grammatical meanings.
- Morpheme boundaries are often blurred.

Example (Spanish):

hablo = “I speak”

The suffix -o encodes first person, singular, present tense, and indicative mood, all at once.

Contrast with agglutinative: In Turkish, each piece of meaning is a separate, identifiable suffix; in Spanish, meanings fuse into a single morpheme.

Why does this matter for NLP?

Different typologies pose different challenges for tokenization and modeling.

Morphological Typology: More Patterns

Polysynthetic languages (e.g., Inuktitut, Mohawk):

- Many morphemes are packed into a single word, often expressing what would be a whole sentence in English.
- Example (Inuktitut): tusaatsiarunнанngittualuujunga = “I can’t hear very well.”

Morphological Typology: More Patterns

Polysynthetic languages (e.g., Inuktitut, Mohawk):

- Many morphemes are packed into a single word, often expressing what would be a whole sentence in English.
- Example (Inuktitut): tusaatsiarunнанngittualuujunga = “I can’t hear very well.”

Non-concatenative (root-and-pattern) morphology (e.g., Arabic, Hebrew):

- A consonantal root interleaves with a vowel template, rather than adding prefixes or suffixes linearly.
- Example (Arabic root k-t-b, meaning “write”): kataba (he wrote), yaktubu (he writes), kitāb (book), kutub (books).

Implication for NLP: word-level tokenization struggles with both patterns; subword units are essential.

Morphological Decomposition

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

- walked = walk + ed
- greatness = great + ness

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) + ize) + ation
The final e in -ize is deleted before -ation.

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) + ize) + ation
The final e in -ize is deleted before -ation.

There is ambiguity in hierarchical decomposition!

The door is unlockable.

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) + ize) + ation
The final e in -ize is deleted before -ation.

There is ambiguity in hierarchical decomposition!

The door is unlockable.

un + (lock + able) = “cannot be locked”

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) + ize) + ation
The final e in -ize is deleted before -ation.

There is ambiguity in hierarchical decomposition!

The door is unlockable.

un + (lock + able) = “cannot be locked”

(un + lock) + able = “can be unlocked”

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 as it appears in text.

wordform	lemma
run	run
ran	run
running	run

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 as it appears in text.

wordform	lemma
run	run
ran	run
running	run

Stemming: reducing words to their stems (approximately) by removing affixes.

More engineering-oriented; used in applications such as retrieval.

average → **averag**, machinery → **machineri**, airplanes → **airplan**

Part III: Tokenization

Tokenization

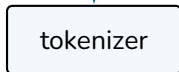
Tokenization: the process that converts running text (i.e., a sequence of characters) into a sequence of tokens.

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

Tokenization

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

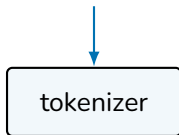
`“Oh!” said Lydia stoutly, “I am not afraid; for
though I _am_ the youngest, I’m the tallest.”`



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."`



`" 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

- Check for consistency before comparing results across tokenizers.
- See `nltk.tokenize`, which also works for sentence tokenization.

Tokenization across Languages

Some languages (Chinese, Japanese, Thai) write without whitespace between words, which makes tokenization considerably harder.

Chinese Treebank: 姚明 进入 总决赛 3 words (“YaoMing reaches finals”)

Peking University: 姚 明 进 入 总 决 赛 5 words (“Yao Ming reaches overall finals”)

Even among the same language, different annotation standards disagree on word boundaries.

Word Types vs. Word Tokens

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

Word Types vs. Word Tokens

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

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

Token: an instance of a type in the text.

Word Types vs. Word Tokens

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

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

Token: an instance of a type in the text.

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

Word Types vs. Word Tokens

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

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

→ 21 types.

Token: an instance of a type in the text.

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

Word Types vs. Word Tokens

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

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

→ 21 types.

Token: an instance of a type in the text.

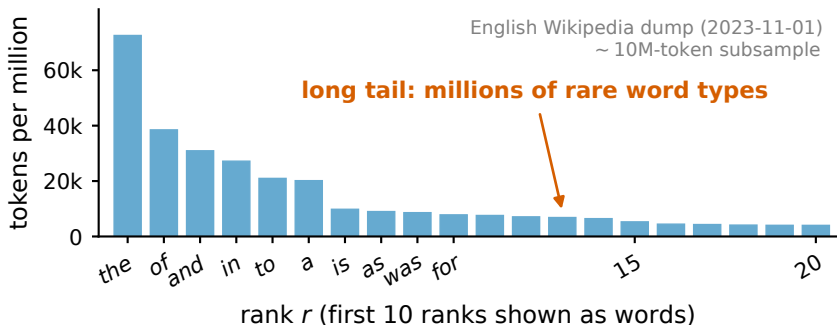
→ 29 tokens.

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

Zipf's Law

The frequency of a word is (roughly) inversely proportional to its rank:

$$f(r) \propto \frac{1}{r^\alpha}, \quad \alpha \approx 1.$$



A handful of words are extremely common, and most are rare: the **long tail**.

How Are Words Distributed?

The long tail means many distinct word types but most are rare.

Example: variants of “really” on Twitter:

[cf. [Brody & Diakopoulos \(2011\)](#)]

224571	really	216	reallyy	89	reeeally
1189	rly	156	relly	84	reaaally
1119	realy	146	reallllly	82	reaally
731	rlly	132	rily	72	reeeeeally
590	realllly	104	reallyyy	65	reaaaaally
234	realllly	89	realllllly	...	

With word-level tokenization, each variant is a separate vocabulary entry.
Can we do better?

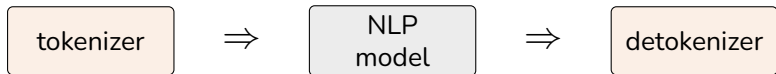
Part IV: Subword Tokenization

Tokenization in Modern NLP Systems

There are many word types, but words share internal structures and meanings (recall morphology).

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

Raw text input



Raw text output

Idea: subword tokenization: split words into smaller, reusable pieces.

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.

Three most popular methods:

- Byte Pair Encoding (BPE): Gage (1994), Sennrich et al. (2016).
- WordPiece: Schuster & Nakajima (2012).
- Unigram LM tokenizer: 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 *Proc. ACL* (pp. 1715–1725).

Kudo, T. (2018). Subword regularization. In *Proc. ACL* (pp. 66–75).

BPE: The Key Idea

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

BPE: Step-by-Step Example

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

BPE: Step-by-Step Example

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, ...

BPE: Step-by-Step Example

Training Corpus

cat
cats
concatenation
categorization

Initial Vocabulary

a c e g i n o r s t z

Count Symbol-Pair Frequencies

$\langle a\ t \rangle$: 6, $\langle c\ a \rangle$: 4, $\langle o\ n \rangle$: 3, ...

Updated Vocabulary

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

BPE: Step-by-Step Example

Training Corpus

cat
cats
concatenation
categorization

Initial Vocabulary

a c e g i n o r s t z

Count Symbol-Pair Frequencies

$\langle a\ t \rangle$: 6, $\langle c\ a \rangle$: 4, $\langle o\ n \rangle$: 3, ...

Updated Vocabulary

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

Merge the most frequent pair: $a + t \rightarrow at$.

BPE: Continuing the Merges

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 a t

BPE: Continuing the Merges

Training Corpus

c at
c at 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 a t

Count Symbol-Pair Frequencies

<c at>: 4, <o n>: 3, ...

BPE: Continuing the Merges

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 a t

Count Symbol-Pair Frequencies

<c a t>: 4, <o n>: 3, ...

Updated Vocabulary

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

BPE: Continuing the Merges

Training Corpus

c at
c at 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 a t

Count Symbol-Pair Frequencies

$\langle c \ a \rangle$: 4, $\langle o \ n \rangle$: 3, ...

Updated Vocabulary

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

Merge: $c + at \rightarrow cat$. Repeat until the vocabulary reaches the desired size.

BPE: Training Algorithm

1. Initialize the vocabulary V with all individual characters in the corpus.
2. Split every word in the corpus into characters.
3. Repeat until $|V|$ reaches the desired vocabulary size:
 - (a) Count the frequency of every adjacent symbol pair across the corpus.
 - (b) Find the most frequent pair (s_1, s_2) .
 - (c) Merge every occurrence of (s_1, s_2) into a new symbol s_1s_2 .
 - (d) Add s_1s_2 to the vocabulary V .
 - (e) Record the merge rule: $s_1 + s_2 \rightarrow s_1s_2$.

Output:

- A vocabulary V of subword tokens.
- An ordered list of merge rules.

Applying Trained BPE to New Text

In addition to the vocabulary, we need the merge rules (in order).

Vocabulary

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

Applying Trained BPE to New Text

In addition to the vocabulary, we need the merge rules (in order).

Vocabulary

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

Merge Rules

a t $\rightarrow$ at

c at $\rightarrow$ cat

Applying Trained BPE to New Text

In addition to the vocabulary, we need the merge rules (in order).

Vocabulary

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

Word to be tokenized

c a t e g o r y

Merge Rules

a t $\rightarrow$ at

c at $\rightarrow$ cat

Applying Trained BPE to New Text

In addition to the vocabulary, we need the merge rules (in order).

Vocabulary

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

Word to be tokenized

c a t e g o r y

Merge Rules

a t $\rightarrow$ at

c at $\rightarrow$ cat

Result

cat e g o r <UNK>

Start from individual characters and apply merge rules in order.

If there is an unknown character, add a special <UNK> token (unlikely with byte-level BPE or with a basic character set as initial vocabulary).

BPE: Optimality

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

BPE Vocabulary ($|V| = 13$)
a c e g i n o r s t z at cat

BPE: Optimality

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

BPE Vocabulary ($|V| = 13$)
a c e g i n o r s t z at cat

A Better Vocabulary ($|V| = 13$)
a c e g i n o r s t z cat on

BPE: Optimality

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

BPE Vocabulary ($|V| = 13$)
a c e g i n o r s t z a t c a t

A Better Vocabulary ($|V| = 13$)
a c e g i n o r s t z c a t o n

The vocabulary on the right yields a shorter encoding of the training corpus (since “on” appears in concatenation, categorization).

BPE: Optimality

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

BPE Vocabulary ($|V| = 13$)
a c e g i n o r s t z at cat

A Better Vocabulary ($|V| = 13$)
a c e g i n o r s t z cat on

The vocabulary on the right yields a shorter encoding of the training corpus (since “on” appears in concatenation, categorization).

Can we find the optimal vocabulary? → Unigram LM tokenizer.

Note: BPE merges alone cannot reach it, since forming cat consumes two merge steps (via at or ca), so {cat, on} requires $|V| \geq 14$.

WordPiece

Used by BERT, DistilBERT, and other Google models.

Similar to BPE, but the merge criterion is different:

- BPE merges the most frequent pair.
- WordPiece merges the pair that maximizes the likelihood of the training data under a language model.

WordPiece

Used by BERT, DistilBERT, and other Google models.

Similar to BPE, but the merge criterion is different:

- BPE merges the most frequent pair.
- WordPiece merges the pair that maximizes the likelihood of the training data under a language model.

Specifically, merge the pair (s_1, s_2) that maximizes:

$$\frac{P(s_1 s_2)}{P(s_1) P(s_2)}$$

where probabilities are estimated from corpus frequencies.

Intuition: merge the pair whose joint occurrence is most “surprising” relative to independent occurrences, i.e., the pair with the highest **pointwise mutual information (PMI)**.

Notation: non-initial subwords are prefixed with ##.

Example: tokenization $\rightarrow$ token ##ization

Unigram LM Tokenizer (SentencePiece)

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

Given a segmentation $\langle x_1, x_2, \dots, x_n \rangle$ of a sentence, the unigram LM assumes:

$$P(\langle x_1, x_2, \dots, x_n \rangle) = \prod_{i=1}^n P(x_i), \quad P(x_i) = \frac{\text{count}(x_i)}{\sum_{x \in V} \text{count}(x)}.$$

Unigram LM Tokenizer (SentencePiece)

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

Given a segmentation $\langle x_1, x_2, \dots, x_n \rangle$ of a sentence, the unigram LM assumes:

$$P(\langle x_1, x_2, \dots, x_n \rangle) = \prod_{i=1}^n P(x_i), \quad P(x_i) = \frac{\text{count}(x_i)}{\sum_{x \in V} \text{count}(x)}.$$

Key difference from BPE:

- BPE builds up a vocabulary by iteratively merging pairs (**bottom-up**).
- Unigram LM starts large and iteratively **removes** tokens whose removal least degrades the overall likelihood (**top-down**).

Toolkit: github.com/google/sentencepiece

Viterbi Decoding: The Key Idea

Goal: given a string $X_{1:m}$, vocabulary V with unigram probabilities P , find the best segmentation

$$x^* = \arg \max_{\text{segmentations of } X} \prod_i P(x_i).$$

Viterbi Decoding: The Key Idea

Goal: given a string $X_{1:m}$, vocabulary V with unigram probabilities P , find the best segmentation

$$x^* = \arg \max_{\text{segmentations of } X} \prod_i P(x_i).$$

Challenge: an m -character string has 2^{m-1} possible segmentations (2^{39} at $m=40$); enumeration is infeasible $\Rightarrow$ dynamic programming (Viterbi, 1967).

Viterbi Decoding: The Key Idea

Goal: given a string $X_{1:m}$, vocabulary V with unigram probabilities P , find the best segmentation

$$x^* = \arg \max_{\text{segmentations of } X} \prod_i P(x_i).$$

Challenge: an m -character string has 2^{m-1} possible segmentations (2^{39} at $m=40$); enumeration is infeasible $\Rightarrow$ dynamic programming (Viterbi, 1967).

DP formulation. Take states = character boundaries $0, \dots, m$; a transition $j \rightarrow i$ exists iff $X_{j+1:i} \in V$.

- δ_i : probability of the best segmentation of the prefix $X_{1:i}$.
- β_i : start boundary of its last token (a backpointer).

Viterbi Decoding: The Key Idea

Goal: given a string $X_{1:m}$, vocabulary V with unigram probabilities P , find the best segmentation

$$x^* = \arg \max_{\text{segmentations of } X} \prod_i P(x_i).$$

Challenge: an m -character string has 2^{m-1} possible segmentations (2^{39} at $m=40$); enumeration is infeasible $\Rightarrow$ dynamic programming (Viterbi, 1967).

DP formulation. Take states = character boundaries $0, \dots, m$; a transition $j \rightarrow i$ exists iff $X_{j+1:i} \in V$.

- δ_i : probability of the best segmentation of the prefix $X_{1:i}$.
- β_i : start boundary of its last token (a backpointer).

Recurrence (state transition equation):

$$\delta_0 = 1, \quad \delta_i = \max_{j < i, X_{j+1:i} \in V} \delta_j \cdot P(X_{j+1:i}), \quad \beta_i = \arg \max_{j < i, X_{j+1:i} \in V} \delta_j \cdot P(X_{j+1:i})$$

Viterbi Decoding: The Key Idea

Goal: given a string $X_{1:m}$, vocabulary V with unigram probabilities P , find the best segmentation

$$x^* = \arg \max_{\text{segmentations of } X} \prod_i P(x_i).$$

Challenge: an m -character string has 2^{m-1} possible segmentations (2^{39} at $m=40$); enumeration is infeasible $\Rightarrow$ dynamic programming (Viterbi, 1967).

DP formulation. Take states = character boundaries $0, \dots, m$; a transition $j \rightarrow i$ exists iff $X_{j+1:i} \in V$.

- δ_i : probability of the best segmentation of the prefix $X_{1:i}$.
- β_i : start boundary of its last token (a backpointer).

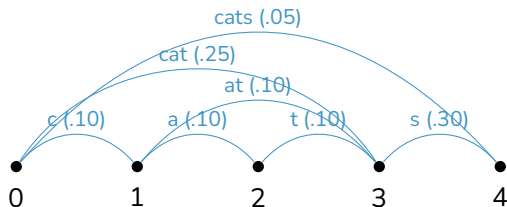
Recurrence (state transition equation):

$$\delta_0 = 1, \quad \delta_i = \max_{j < i, X_{j+1:i} \in V} \delta_j \cdot P(X_{j+1:i}), \quad \beta_i = \arg \max_{j < i, X_{j+1:i} \in V} \delta_j \cdot P(X_{j+1:i})$$

δ_m = probability of the best segmentation; recover tokens by following β back from m . In practice: maximize $\sum \log P$ (same result, no underflow). Complexity $O(m \cdot L_{\max})$; $L_{\max}$ = max piece length.

Viterbi Decoding: Step-by-Step Example

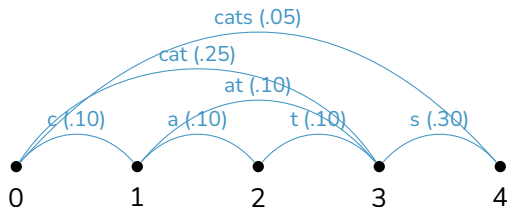
Segmentation lattice for $X = \text{cats}$ ($m=4$; $2^3 = 8$ segmentations); arc labels give $\text{token}(P)$ from V .



Viterbi Decoding: Step-by-Step Example

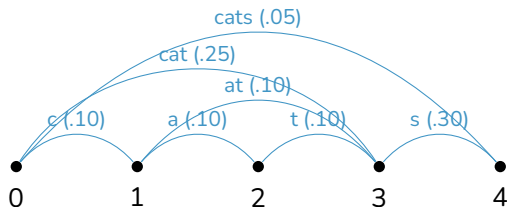
Segmentation lattice for $X = \text{cats}$ ($m=4$; $2^3 = 8$ segmentations); arc labels give $\text{token}(P)$ from V .

Candidates $j \rightarrow i$: $\text{token}(\delta_j \cdot P(\text{token}))$;
orange = best.



Viterbi Decoding: Step-by-Step Example

Segmentation lattice for $X = \text{cats}$ ($m=4$; $2^3 = 8$ segmentations); arc labels give $\text{token}(P)$ from V .



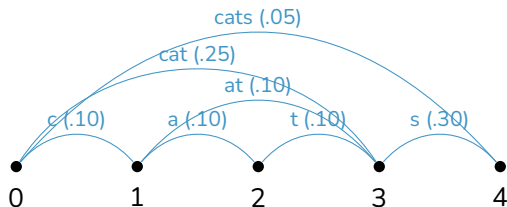
Candidates $j \rightarrow i$: $\text{token}(\delta_j \cdot P(\text{token}))$;
orange = best.

$i=1$: c (.10)

$\Rightarrow \delta_1 = .10, \beta_1 = 0$

Viterbi Decoding: Step-by-Step Example

Segmentation lattice for $X = \text{cats}$ ($m=4$; $2^3 = 8$ segmentations); arc labels give $\text{token}(P)$ from V .



Candidates $j \rightarrow i$: $\text{token}(\delta_j \cdot P(\text{token}))$;
orange = best.

$i=1$: c (.10)

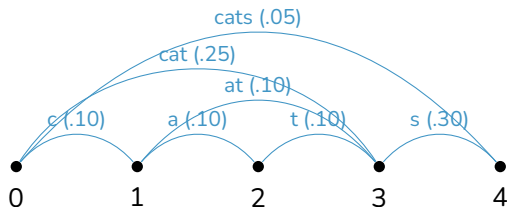
$\Rightarrow \delta_1 = .10, \beta_1 = 0$

$i=2$: a (.010)

$\Rightarrow \delta_2 = .010, \beta_2 = 1$

Viterbi Decoding: Step-by-Step Example

Segmentation lattice for $X = \text{cats}$ ($m=4$; $2^3 = 8$ segmentations); arc labels give $\text{token}(P)$ from V .



Candidates $j \rightarrow i$: $\text{token}(\delta_j \cdot P(\text{token}))$;
orange = best.

$i=1$: c (.10)

$\Rightarrow \delta_1 = .10, \beta_1 = 0$

$i=2$: a (.010)

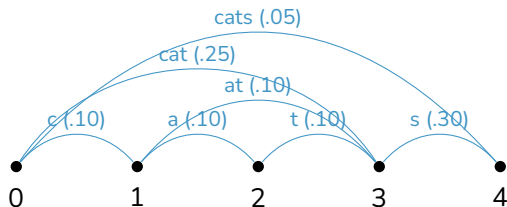
$\Rightarrow \delta_2 = .010, \beta_2 = 1$

$i=3$: t (.001), at (.010), **cat (.25)**

$\Rightarrow \delta_3 = .250, \beta_3 = 0$

Viterbi Decoding: Step-by-Step Example

Segmentation lattice for $X = \text{cats}$ ($m=4$; $2^3 = 8$ segmentations); arc labels give $\text{token}(P)$ from V .



Candidates $j \rightarrow i$: $\text{token}(\delta_j \cdot P(\text{token}))$;
orange = best.

$i=1$: c (.10)

$\Rightarrow \delta_1 = .10, \beta_1 = 0$

$i=2$: a (.010)

$\Rightarrow \delta_2 = .010, \beta_2 = 1$

$i=3$: t (.001), at (.010), **cat (.25)**

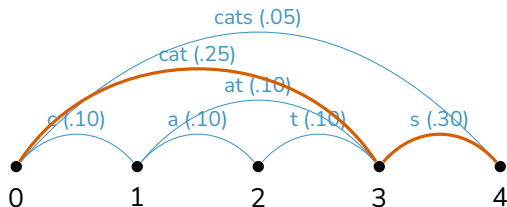
$\Rightarrow \delta_3 = .250, \beta_3 = 0$

$i=4$: **s (.075)**, cats (.050)

$\Rightarrow \delta_4 = .075, \beta_4 = 3$

Viterbi Decoding: Step-by-Step Example

Segmentation lattice for $X = \text{cats}$ ($m=4$; $2^3 = 8$ segmentations); arc labels give $\text{token}(P)$ from V .



Candidates $j \rightarrow i$: $\text{token}(\delta_j \cdot P(\text{token}))$;
orange = best.

$i=1$: c (.10)

$\Rightarrow \delta_1 = .10, \beta_1 = 0$

$i=2$: a (.010)

$\Rightarrow \delta_2 = .010, \beta_2 = 1$

$i=3$: t (.001), at (.010), **cat (.25)**

$\Rightarrow \delta_3 = .250, \beta_3 = 0$

$i=4$: **s (.075)**, cats (.050)

$\Rightarrow \delta_4 = .075, \beta_4 = 3$

Backtrace: $\beta_4 = 3 \Rightarrow$ last token s; $\beta_3 = 0 \Rightarrow$ token cat.

Best segmentation: cat | s ($P = .075$); runner-up cats ($P = .050$).

Unigram LM: Training Algorithm

1. Start with a large seed vocabulary V (e.g., all sufficiently frequent substrings of the corpus up to a max length, plus all single characters).
2. Repeat until $|V|$ reaches the desired size:
 - (a) Re-estimate probabilities (EM). E-step: count token occurrences in the Viterbi-best segmentations of the corpus under the current V ; M-step: normalize counts to obtain new $P(x)$.
 - (b) Score tokens. For each $x \in V$, measure the loss increase caused by removing x : $\Delta(x) = \mathcal{L}(V \setminus \{x\}) - \mathcal{L}(V)$, where $\mathcal{L}(V) = -\sum_{\text{sent. } s} \log P(s | V)$ and $P(s | V)$ uses the Viterbi (best) segmentation under the current vocabulary.
 - (c) Prune. Remove a fraction (e.g., 10–20%) of the tokens with the smallest $\Delta(x)$; single characters are never removed, so every string remains segmentable.
3. Output the final vocabulary with its unigram probabilities $P(x)$.

Top-down only: tokens are only removed, never added; the candidate space is fixed once by the seed vocabulary.

At inference time, the same Viterbi decoding is used to tokenize new text.

Comparison of Subword Tokenizers

	BPE	WordPiece	Unigram LM
Direction	bottom-up	bottom-up	top-down
Merge criterion	frequency	likelihood / PMI	n/a
Pruning criterion	n/a	n/a	likelihood loss
Segmentation	greedy merges	greedy merges	Viterbi
Used by	GPT, Llama	BERT	T5, XLNet

- BPE: simple, fast, widely used; greedy and suboptimal.
- WordPiece: similar to BPE but likelihood-driven merges.
- Unigram LM: probabilistic, allows [subword regularization](#) (sampling different segmentations during training for better robustness).

Byte-Level BPE

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

Byte-level BPE (used by GPT-2, GPT-3, Llama):

- Represent text as a sequence of UTF-8 bytes (256 possible values).
- Run BPE on the byte sequence instead of characters.
- No <UNK> tokens: any text in any script can be encoded.

Example:

“That’s great” → byte sequence: 54 68 61 74 ... (hexadecimal)

BPE then merges frequent byte pairs into multi-byte tokens.

Trade-off: Byte-level BPE can handle any script, but tokens for non-Latin scripts may be very short (1–2 bytes per token), leading to longer sequences.

Cross-lingual Considerations

Cross-lingual Considerations

Why does tokenization matter across languages?

- Vocabulary sharing: Multilingual models (mBERT, XLM-R) train a single tokenizer on data from 100+ languages.
 - High-resource languages (English) dominate the vocabulary.
 - Low-resource languages get fragmented into many small pieces.
- Fertility: the average number of subword tokens per word.
 - English word $\rightarrow \sim 1.2$ tokens on average.
 - Low-resource language word $\rightarrow \sim 3\text{--}5$ tokens.
 - Higher fertility $\Rightarrow$ longer sequences $\Rightarrow$ more computation, less context in the window.

Cross-lingual Considerations: Vocabulary Allocation

The vocabulary allocation problem:

- With a fixed vocabulary budget (e.g., 32K tokens), how should tokens be allocated across languages?
- Naive approach: train BPE on all data $\rightarrow$ high-resource languages dominate.
- Temperature-based sampling: sample training data from language ℓ with probability:

$$p_{\ell} \propto \left(\frac{n_{\ell}}{\sum_{\ell'} n_{\ell'}} \right)^{1/T}$$

where n_{ℓ} is the data size and T controls the smoothing.

$T = 1$: proportional to data size. $T \rightarrow \infty$: uniform across languages.

- Used in mBERT, XLM-R to balance representation across scripts.

Cross-lingual Considerations: Low-Resource Scripts

Challenges for low-resource scripts:

- Scripts with small training data end up with character-level or even byte-level tokenization → very long sequences.
- Unique scripts (e.g., Burmese, Khmer, Tibetan) may not share any subword tokens with high-resource languages.
- This creates a representation bottleneck.

Mitigation strategies:

- Oversample low-resource language data during tokenizer training.
- Train script-specific tokenizers and combine vocabularies.
- Use byte-level BPE as a universal fallback.
- “Vocabulary extension”: add tokens for a new language to a pretrained model’s vocabulary.

Summary

- What is a word? The definition depends on context, and there is no single universal answer.
- Morphology: Words are built from morphemes (stems + affixes). Languages vary in morphological typology (isolating, agglutinative, fusional).
- Tokenization: Converting text to tokens; rule-based methods are language-specific and inconsistent.
- Subword tokenization: Data-driven methods that learn reusable pieces.
 - BPE: greedy bottom-up merging of frequent pairs.
 - WordPiece: merges based on likelihood / PMI.
 - Unigram LM: top-down pruning to optimize likelihood.
- Cross-lingual: vocabulary sharing, fertility, and the challenges of low-resource scripts.

Readings

Required:

- Sennrich, R., Haddow, B., & Birch, A. (2016). [Neural machine translation of rare words with subword units](#). In Proc. ACL.
- Kudo, T. (2018). [Subword regularization: Improving neural network translation models with multiple subword candidates](#). In Proc. ACL.

Optional background:

- Jurafsky & Martin, SLP 3, Chapter 2 (Words and Tokens).
- Schuster, M. & Nakajima, K. (2012). Japanese and Korean voice search. ICASSP.

Next Lecture

Distributional Semantics and Word Embeddings

- Distributional hypothesis.
- Count-based methods (PMI, SVD).
- Word2Vec (skip-gram, CBOW, negative sampling).

Reading: [SLP 3, Chapter 5 \(Embeddings\)](#).