

# Natural Language Processing

---

Tokenization; Morphology

Ana Marasović  
University of Utah

# Announcements

## Gradescope Autograder for Assignment A1

- Unless your implementation crashes, the autograder will award 20 points for Q2 and Q3, but we will manually review your implementation and deduct points if it is significantly incorrect
- Please add a description of your “better” feature extractor in comments to help us understand what you did

*Recap:*

## 4 components of a supervised ML for binary classification

### 1. **A feature representation of the input**

- ✿ Tokenize text
- ✿ Map a list of tokens into a high-dimensional feature vector

### 2. **A classification function that decides which class to apply to an instance**

- ✿ Extra: Probabilistic classifier
- ✿ Logistic regression

### 3. **An objective function that we want to optimize for to learn appropriate model parameters**

- ✿ Minimizing negative log likelihood

### 4. **An algorithm for finding optimal model parameters/weights for the objective function**

- ✿ Gradient descent

## Recap:

### Two phases

1. **Training:** We use a set of *labeled* training examples  $\mathcal{D} = \{(x^{(i)}, y^{(i)})\}_{i=1}^m$  to make decisions, compute the decision error, & change the model parameters/weights to minimize the error
2. **Testing:** We use a held-out labeled set of test instances (not used for training) and measure the model accuracy

#### Notation notes:

- $\{\}$  denotes a set
- Subscript and superscript next  $\}$  shorten writing  $\mathcal{D} = \{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$
- Superscripts in parentheses refer to individual instances
- $(, )$  denotes a pair
- $x$  denotes input
- $y$  denotes the class label

# Goals & Overview of August 21 Lecture

**Goal:** Learn components of a supervised machine learning system for binary text classification

- ✿ Components of a supervised machine learning system for classification
- ✿ Feature representations of the input
- ✿ The sigmoid function
- ✿ Negative log-likelihood / cross-entropy loss
- ✿ Stochastic gradient descent
- ✿ Accuracy & F1-score

# Accuracy

**Accuracy:** The proportion of true results (both true positives & true negatives)

- $\text{accuracy} = (\text{tp} + \text{tn}) / (\text{tp} + \text{tn} + \text{fp} + \text{fn})$
- Fine when the class distribution is similar
- Fine when the costs of false positives & false negatives are roughly the same

## **A true positive (tp):**

An instance predicted to be a member of a positive class and truly is a member of a positive class

## **A true negative (tn):**

An instance predicted to be a member of a negative class and truly is a member of a negative class

## **A false positive (fp):**

An instance predicted to be a member of a positive class, but is a member of a negative class

## **A false negative (fn):**

An instance predicted to be a member of a negative class, but is a member of a positive class

# F1 Score

**Precision:** The proportion of true positives in the total positive predictions made

- $\text{precision} = \text{tp} / (\text{tp} + \text{fp})$
- E.g. how reliable the test is when it predicts the disease

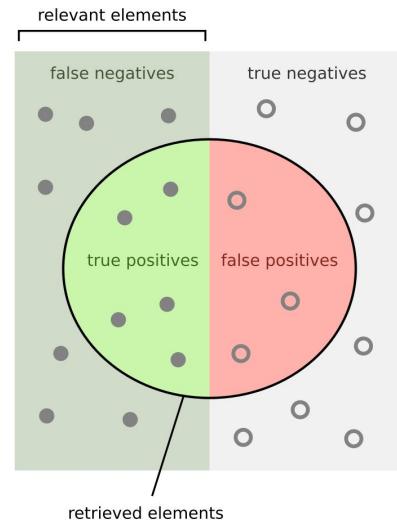
**Recall:** The proportion of true positives that were identified

- $\text{recall} = \text{tp} / (\text{tp} + \text{fn})$
- E.g. how many patients go undiagnosed and untreated

**F1 Score** =  $2 * \text{precision} * \text{recall} / (\text{precision} + \text{recall})$

**Micro F1:** Averages metrics across all instances

**Macro F1:** Averages metrics across all classes, equally weighting each class

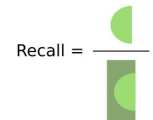


How many retrieved items are relevant?



Source: [Wiki](#)

How many relevant items are retrieved?



Today...



# 4 components of a supervised ML for binary classification

## 1. A feature representation of the input

- ✿ Tokenize text

- ✿ Map a list of tokens into a high-dimensional feature vector

## 2. A classification function that decides which class to apply to an instance

- ✿ Extra: Probabilistic classifier

- ✿ Logistic regression

## 3. An objective function that we want to optimize for to learn appropriate model parameters

- ✿ Minimizing negative log likelihood

## 4. An algorithm for finding optimal model parameters/weights for the objective function

- ✿ Gradient descent

# Reminder: What's tokenization

**Splitting a string into a sequence of *tokens***

**Tokens: “basic units which need not be decomposed in a subsequent processing”**

[\[Webster and Kit, 1992\]](#)

# Reminder: Whitespace tokenizer

Tokens are implied to be *words*

Example:

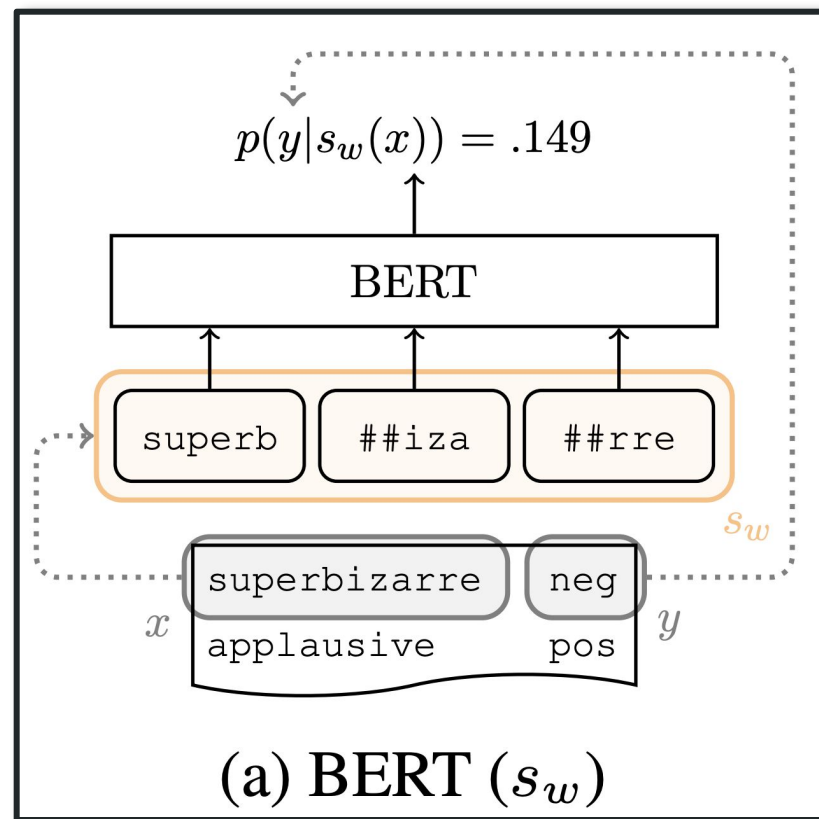
- **Text:** “This film is interesting as an experiment but tells no cogent story.”
- ⇒ **Tokenized text:** [“This“, “film“, “is“, “interesting“, “as“, “an“, “experiment“, “but“, “tells“, “no“, “cogent“, “story.“]

Whitespace tokenizer failures

- *conjunctions:* isn’t ⇒ is, n’t
- *hyphenated phrases:* prize-winning ⇒ prize, -, winning
- *punctuation:* great movie! ⇒ great, movie, !



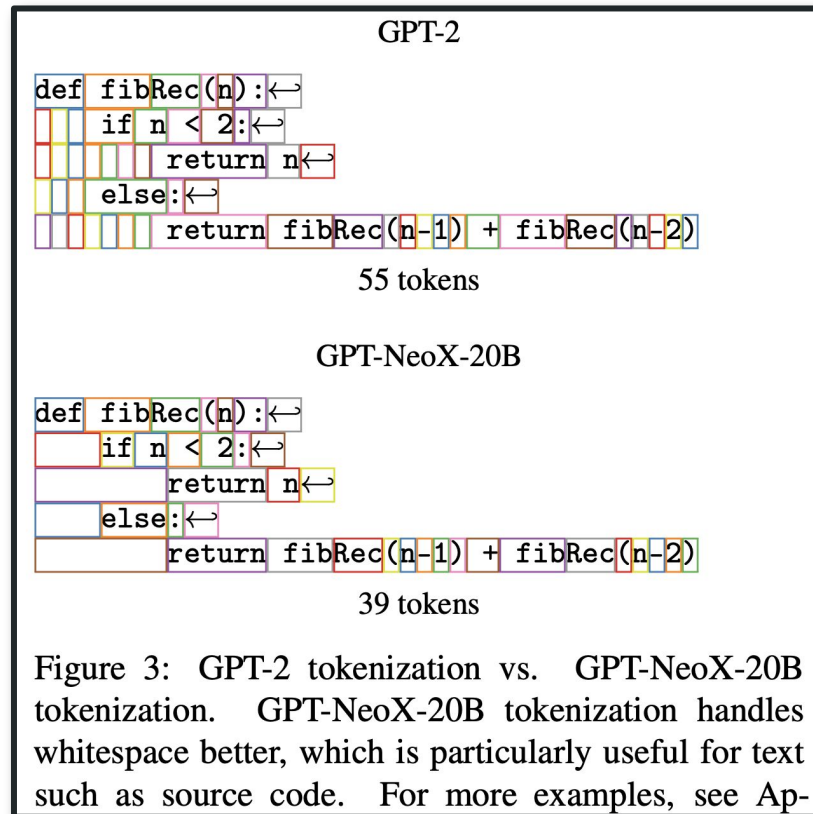
BERT thinks the sentiment of "superbizarre" is positive because its tokenization contains the token "superb"



Natural phenomena like diacritics or a little easily human-readable noise lead to unexpected BPE sequences and catastrophic translation failure

| Arabic–English      |   |
|---------------------|---|
| src                 | أنا كنديّة، وأنا أصغر أخواني السبعة                       |
| diacritics 1.0      | أنا كَنَدِيَّةٌ ، وأنا أَصْغَرُ إِخْوَانِي السَّبْعَةِ    |
| ref                 | I'm Canadian, and I'm the youngest of seven kids.         |
| in <sub>vis</sub>   | أنا كنديّة كنديّة كنديّة ، وأنا أصغر أخواني السبعة        |
| out <sub>vis</sub>  | I'm a Canadian, and I'm the youngest of my seven sisters. |
| COMET               | 0.764   |
| in <sub>text</sub>  | ..أنا كنديّة كنديّة كنديّة ، وأنا أصغر أخواني السبعة      |
| out <sub>text</sub> | We grew up as a teacher, and we gave me a hug.            |
| COMET               | -1.387  |

Manually adding spaces sequences  
up to 24 to better handle code



[[Black et al., 2022](#)]

Tokenization is at the heart of much weirdness of LLMs. Do not brush it off.

- Why can't LLM spell words? **Tokenization.**
- Why can't LLM do super simple string processing tasks like reversing a string? **Tokenization.**
- Why is LLM worse at non-English languages (e.g. Japanese)? **Tokenization.**
- Why is LLM bad at simple arithmetic? **Tokenization.**
- Why did GPT-2 have more than necessary trouble coding in Python? **Tokenization.**
- Why did my LLM abruptly halt when it sees the string "<|endoftext|>"? **Tokenization.**
- What is this weird warning I get about a "trailing whitespace"? **Tokenization.**
- Why the LLM break if I ask it about "SolidGoldMagikarp"? **Tokenization.**
- Why should I prefer to use YAML over JSON with LLMs? **Tokenization.**
- Why is LLM not actually end-to-end language modeling? **Tokenization.**
- What is the real root of suffering? **Tokenization.**



# Goals & Overview of Today's Lecture

**Goal:** Learn one modern algorithm for tokenizing text

- ✿ Which units for tokens?
- ✿ BPE tokenizer
- ✿ Few additional notes

# Goals & Overview of Today's Lecture

**Goal:** Learn one modern algorithm for tokenizing text

- ✿ Which units for tokens?
- ✿ BPE tokenizer
- ✿ Few additional notes

# Reminder: What's tokenization

**Splitting a string into a sequence of *tokens***

**Tokens: “basic units which need not be decomposed in a subsequent processing”**

[\[Webster and Kit, 1992\]](#)

# What if a new (or infrequent) word appears?

## UNK tokens:

- ✿ Historically rare word types were replaced with a new word type UNK (unknown) at train time
- ✿ At test time, any token that wasn't part of the model's vocabulary could be replaced by UNK

## Issues with UNK tokens:

- ✿ You should not generate UNK when generating text (imagine ChatGPT doing this 🤖)
- ✿ UNKs don't give features for novel words that are useful anchors of meaning
- ✿ In languages with more productive morphology [not English], removing rare words is infeasible

**Out-of-vocabulary (OOV):** Words that were seen very rarely during training or not even at all

**Closed-vocabulary models:** Unable to produce word forms unseen in training data

# Maximal decomposition into **characters**

But deciding what counts as a word in Chinese is complex. For example, consider the following sentence:

(2.4) 姚明进入总决赛  
“Yao Ming reaches the finals”

As [Chen et al. \(2017\)](#) point out, this could be treated as 3 words (‘Chinese Treebank’ segmentation):

(2.5) 姚明 进入 总决赛  
YaoMing reaches finals

or as 5 words (‘Peking University’ segmentation):

(2.6) 姚 明 进入 总 决赛  
Yao Ming reaches overall finals

Finally, it is possible in Chinese simply to ignore words altogether and use characters as the basic elements, treating the sentence as a series of 7 characters:

(2.7) 姚 明 进 入 总 决 赛  
Yao Ming enter enter overall decision game

In fact, for most Chinese NLP tasks it turns out to work better to take characters rather than words as input, since characters are at a reasonable semantic level for most applications, and since most word standards, by contrast, result in a huge vocabulary with large numbers of very rare words ([Li et al., 2019](#)).

But character-level models applied to languages like English must discover that words exist and are delimited by spaces—basic linguistic facts that are built in to the structure of word-based models

# Linguistically motivated units?

- ✦ The choice of units is important:
  - These units often receive annotations
- ✦ A large range of phenomena (*conjunctions, hyphenated phrases, punctuation, etc.*) ⇒  
Hard to identify and to consistently define linguistic units
- ✦ Typographic units have been used as an approximation for linguistically motivated units
- ✦ Whitespace is a typographic separator that's now universally used with the Latin script

# A redefinition of the notion of tokenization

Due to:

- Scientific results: The impact of sub-word segmentation on machine translation performance in 2016
- Technical requirements: A fixed-size vocabulary for neural language models & a reasonable vocabulary size

...in current NLP, the notion of *token* and *tokenization* changed

**“Tokenization”** is now the task of segmenting a sentence into non-typographically (& non-linguistically) motivated units, which are often smaller than classical tokens, and therefore often called **sub-words**

Typographic units (the “old” tokens) are now often called **“pre-tokens”**, and what used to be called “tokenization” is therefore called nowadays **“pre-tokenization”**

- [https://github.com/huggingface/tokenizers/tree/main/tokenizers/src/pre\\_tokenizers](https://github.com/huggingface/tokenizers/tree/main/tokenizers/src/pre_tokenizers)

**Unseen word can be represented by some sequence of known subwords; “lower”=“low”+ “er”**

# “Digression”: Some related terminology

- ✿ A **morpheme** is the smallest meaning-bearing unit of a language
  - “unlikeliest” has the morphemes {un-, likely, -est}
- ✿ **Morphology** is the study of the way words are built up from morphemes
- ✿ **Word forms** are the variations of a word that express different grammatical categories
  - **Tense** (when something happened; past, present, future)
  - **Case** (inflecting nouns/pronoun and their modifiers to express their relationship with other words; English has largely lost its inflected case system)
  - **Number** (singular/plural)
  - **Gender** (masculine, feminine, neuter; not extensively used in English)

and thus help convey the specific meaning and function of the word in a sentence

**Subwords** can be arbitrary substrings, but also meaning-bearing units like the morphemes -est or -er



# Goals & Overview of Today's Lecture

**Goal:** Learn one modern algorithm for tokenizing text

- 📌 Which units for tokens?
- 📌 BPE tokenizer
- 📌 Few additional notes

# Byte-Pair-Encoding (BPE)

[coined by [Gage et al., 1994](#); adapted to the task of word segmentation by [Sennrich et al., 2016](#); see [Gallé \(2019\)](#) for more]

**Main idea:** Use our data to automatically tell us what the tokens should be

**Token learner:**

Raw train corpus  $\Rightarrow$  Vocabulary (a set of tokens)

**Token segmenter:**

Raw sentences  $\Rightarrow$  Tokens in the vocabulary

**Types** are distinct tokens in a corpus

**Corpus** vs. **Dataset**

# Byte-Pair-Encoding (BPE) – Token learner

[coined by [Gage et al., 1994](#); adapted to the task of word segmentation by [Sennrich et al., 2016](#); see [Gallé \(2019\)](#) for more]

Raw train corpus  $\Rightarrow$  Vocabulary (a set of tokens)

- ✦ Pre-tokenize the corpus in words & append a special end-of-word symbol \_ to each word
- ✦ Initialize vocabulary with the set of all individual characters
- ✦ Choose 2 tokens that are most frequently adjacent (“A”, “B”)
  - Respect word boundaries: Run the algorithm inside words
- ✦ Add a new merged symbol (“AB”) to the vocabulary
- ✦ Change the occurrence of the 2 selected tokens with the new merged token in the corpus
- ✦ Continues doing this until **k** merges are done

All **k** new symbols and initial characters are the final vocabulary

**What’s k?** Open research question, see [Mielke et al., 2021](#) Sec 6.6; 30K is seen frequently

# Byte-Pair-Encoding (BPE) – Token learner *Example*

**corpus**

end-of-word symbol

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 5 | l | o | w | — |   |   |   |
| 2 | l | o | w | e | s | t | — |
| 6 | n | e | w | e | r | — |   |
| 3 | w | i | d | e | r | — |   |
| 2 | n | e | w | — |   |   |   |

**vocabulary**

—, d, e, i, l, n, o, r, s, t, w

each word is split into characters

word occurrence  
count in the corpus

# Byte-Pair-Encoding (BPE) – Token learner *Example*

- ✎ Counts all pairs of adjacent symbols
- ✎ The most frequent is the pair **e r** [a total of 9 occurrences]
- ✎ Merge these symbols, treating **er** as one symbol, & add the new symbol to the vocabulary

## corpus

5 l o w \_  
2 l o w e s t \_  
6 n e w er \_  
3 w i d er \_  
2 n e w \_

## vocabulary

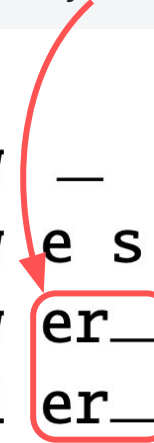
\_, d, e, i, l, n, o, r, s, t, w, er

# Byte-Pair-Encoding (BPE) – Token learner *Example*

- ✎ Counts all pairs of adjacent symbols
- ✎ The most frequent is the pair **er** \_
- ✎ Merge these symbols, treating **er\_** as one symbol, & add the new symbol to the vocabulary

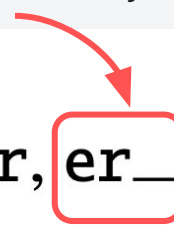
## corpus

5    l o w \_  
2    l o w e s t \_  
6    n e w **er\_**  
3    w i d **er\_**  
2    n e w \_



## vocabulary

\_, d, e, i, l, n, o, r, s, t, w, er, **er\_**



# Byte-Pair-Encoding (BPE) – Token learner *Example*

- ✎ Counts all pairs of adjacent symbols
- ✎ The most frequent is the pair **n e**
- ✎ Merge these symbols, treating **ne** as one symbol, & add the new symbol to the vocabulary

## corpus

5 l o w \_  
2 l o w e s t \_  
6 **ne** w e r\_  
3 w i d e r\_  
2 **ne** w \_

## vocabulary

\_, d, e, i, l, n, o, r, s, t, w, e r, e r\_, **ne**

# Byte-Pair-Encoding (BPE) – Token learner *Example*

## merge

(ne, w)

(l, o)

(lo, w)

(new, er\_)

(low, \_)

## current vocabulary

\_, d, e, i, l, n, o, r, s, t, w, er, er\_, ne, new

\_, d, e, i, l, n, o, r, s, t, w, er, er\_, ne, new, lo

\_, d, e, i, l, n, o, r, s, t, w, er, er\_, ne, new, lo, low

\_, d, e, i, l, n, o, r, s, t, w, er, er\_, ne, new, lo, low, newer\_

\_, d, e, i, l, n, o, r, s, t, w, er, er\_, ne, new, lo, low, newer\_, low\_

Final vocabulary



# Byte-Pair-Encoding (BPE) – Pseudocode

[coined by [Gage et al., 1994](#); adapted to the task of word segmentation by [Sennrich et al., 2016](#); see [Gallé \(2019\)](#) for more]

```
function BYTE-PAIR ENCODING(strings C, number of merges k) returns vocab V
V ← all unique characters in C # initial set of tokens is characters
for i = 1 to k do # merge tokens k times
    tL, tR ← Most frequent pair of adjacent tokens in C
    tNEW ← tL + tR # make new token by concatenating
    V ← V + tNEW # update the vocabulary
    Replace each occurrence of a pair tL, tR in C with the string tNEW # and update the corpus
return V
```

# Byte-Pair-Encoding (BPE) – Token segmenter

[coined by [Gage et al., 1994](#); adapted to the task of word segmentation by [Sennrich et al., 2016](#); see [Gallé \(2019\)](#) for more]

The token segmenter is used to tokenize a test sentence

- ✦ Just runs on the test data the merges we've learned from the training data, greedily, in the order we learned them

First we segment each test sentence word into characters

Then we apply the first merge rule

- ✦ E.g., replace every instance of **e r** in the test corpus with **er**

Then the second merge rule

- ✦ E.g., replace every instance of **er \_** in the test corpus with **er\_**

And so on

## Byte-Pair-Encoding (BPE) – In Python (Demo)

[utah-cs6340-f24\\_tokenization.ipynb](#)

[https://github.com/huggingface/tokenizers](#) +  
[notebooks/examples/tokenizer\\_training.ipynb at main · huggingface/notebooks · GitHub](#)  
+ [https://huggingface.co/learn/nlp-course/en/chapter6/5](#)

[https://github.com/openai/tiktoken](#)

[https://github.com/karpathy/minbpe/tree/master](#)

# Tiktokenizer (Demo)

Tokenization is at the heart of much weirdness of LLMs. Do not brush it off.

- Why can't LLM spell words? **Tokenization.** `.DefaultCellStyle` is a single token
- Why can't LLM do super simple string processing tasks like reversing a string? **Tokenization.**
- Why is LLM worse at non-English languages (e.g. Japanese)? **Tokenization.**
- Why is LLM bad at simple arithmetic? **Tokenization.** Tokenization of numbers [[Integer tokenization is insane](#)]
- Why did GPT-2 have more than necessary trouble coding in Python? **Tokenization.** Whitespaces
- Why did my LLM abruptly halt when it sees the string "<|endoftext|>"? **Tokenization.** Special tokens
- What is this weird warning I get about a "trailing whitespace"? **Tokenization.**
- Why the LLM break if I ask it about "SolidGoldMagikarp"? **Tokenization.** [[SolidGoldMagikarp — LessWrong](#)]
- Why should I prefer to use YAML over JSON with LLMs? **Tokenization.**
- Why is LLM not actually end-to-end language modeling? **Tokenization.**
- What is the real root of suffering? **Tokenization.**

# Goals & Overview of Today's Lecture

**Goal:** Learn one modern algorithm for tokenizing text

- ✿ Which units for tokens?
- ✿ BPE tokenizer
- ✿ Few additional notes [**next time**]