Understanding Collections of Text

How do we start to talk about language?

In our first class, I talked about language as linguists think about it: through the lens of *levels of linguistic abstraction.*

When we handle raw text, though, those layers aren't always easy to pull apart.

How can we describe text data? How can we start to process it?

What can we find in a corpus of text?

At the most basic level, a corpus is a collection of characters.

Whether or not words are "real", though, they are a useful abstraction. How do we find them?

What are the words?

"I do uh main- mainly business data processing"

- Fragments, filled pauses

"Seuss’s cat in the hat is different from other cats!"

- **Lemma**: same stem, part of speech, rough word sense
  - cat and cats = same lemma
- **Wordform**: the full inflected surface form
  - cat and cats = different wordforms

How many words in a sentence?

$\overset{1}{\text{the}}$ $\overset{2}{\text{the}}$

they lay back on the San Francisco grass and looked at the stars and their

token5 token12

Type: an element of the vocabulary.

Token: an instance of that type in running text.

How many?

15 tokens

13 types

How many words in a corpus?

N = number of tokens

V = vocabulary = set of types, **|*V*|** is size of vocabulary

Heaps Law = Herdan's Law = $|V| = kN^{\beta}$ where often $.67 < \beta < .75$

i.e., vocabulary size grows with > square root of the number of word tokens

	Tokens = N	**Types = |V|**
Switchboard phone conversations	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
COCA	440 million	2 million
Google N-grams	1 trillion	13+ million

Corpora

Words don't appear out of nowhere!

A text is produced by

- a specific writer(s),
- at a specific time,
- in a specific variety,
- of a specific language,
- for a specific function.

Corpora vary along many dimensions:

- **Language**: 7000+ languages in the world
- **Variety**, like African American Language varieties.
  - AAE Twitter posts might include forms like "*iont*" *(I don't)*
- **Code switching**, e.g., Spanish/English, Hindi/English:

  S/E: Por primera vez veo a @username actually being hateful! It was beautiful:)

  [For the first time I get to see @username actually being hateful! it was beautiful:)]

  H/E: dost tha or ra- hega ... dont wory ... but dherya rakhe

  [“he was and will remain a friend ... don’t worry ... but have faith”]
- **Genre:** newswire, fiction, scientific articles, Wikipedia
- **Author Demographics**: writer's age, gender, ethnicity, SES

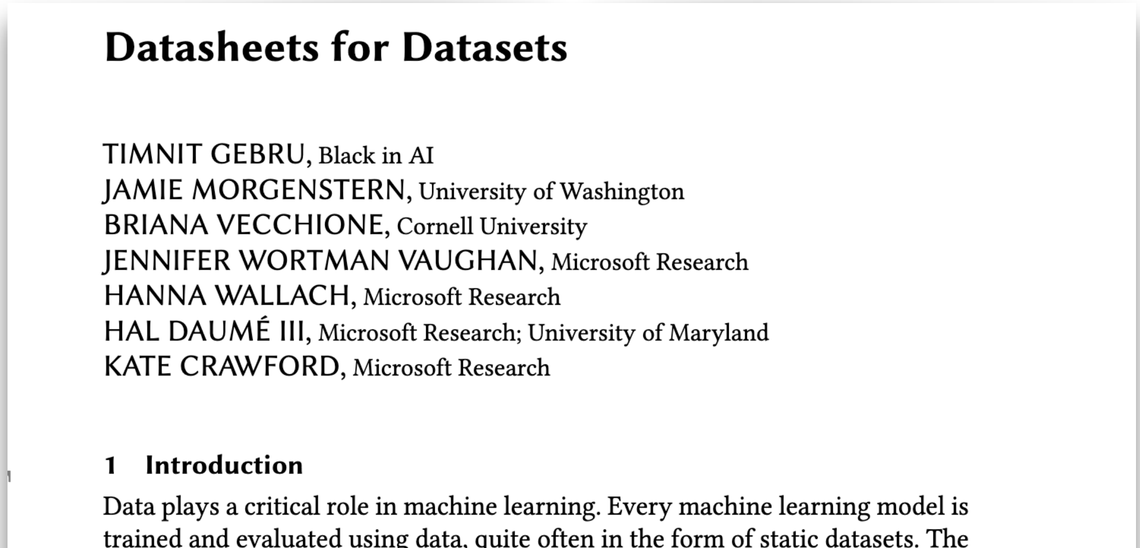
Corpus datasheets

Motivation:
- Why was the corpus collected?
- By whom?
- Who funded it?

Situation: In what situation was the text written?

Collection process: Was there consent? Pre-processing?

Annotation process, language variety, demographics, etc.



Datasheets for Datasets

TIMNIT GEBRU, Black in AI
JAMIE MORGENSTERN, University of Washington
BRIANA VECCHIONE, Cornell University
JENNIFER WORTMAN VAUGHAN, Microsoft Research
HANNA WALLACH, Microsoft Research
HAL DAUMÉ III, Microsoft Research; University of Maryland
KATE CRAWFORD, Microsoft Research

1 Introduction

Data plays a critical role in machine learning. Every machine learning model is trained and evaluated using data, quite often in the form of static datasets. The

Basic Text Processing

Word tokenization

Text Normalization

Every NLP task requires text normalization:

1. Tokenizing (segmenting) words
2. Normalizing word formats
3. Segmenting sentences

Space-based tokenization

A very simple way to tokenize: split on spaces

Caveat: this only works for languages that use space characters between words!

"I write this sitting in the kitchen sink" --->

"I" "write" "this" "sitting" "in" "the" "kitchen" "sink"

Issues in Tokenization

Can't just blindly remove punctuation:

- m.p.h., Ph.D., AT&T, cap’n
- prices ($45.55)
- dates (01/02/06)
- URLs (http://www.stanford.edu)
- hashtags (#nlproc)
- email addresses (someone@cs.colorado.edu)

Clitic: a word that doesn't stand on its own

- "are" in we're, French "je" in j'ai, "le" in l'honneur

When should multiword expressions (MWE) be words?

- New York, rock ’n’ roll

Tokenization in languages without spaces

Even bigger problem: many languages (like Chinese, Japanese, Thai) don't use spaces to separate words.

How can we figure out word boundaries in these languages?

Word tokenization in Chinese

Chinese words are composed of characters called "**hanzi"** (or sometimes just "**zi**")

Each character represents a meaning unit called a ***morpheme.***

A ***morpheme*** is the **smallest meaning-bearing unit** of a language.

- *unlikeliest* has 3 morphemes *un-*, *likely*, and *-es.*

Chinese words have, on average, 2.4 characters. But deciding what counts as a word is complex and not agreed upon.

How to do word tokenization in Chinese?

姚明进入总决赛 “Yao Ming reaches the finals”

How to do word tokenization in Chinese?

姚明进入总决赛 “Yao Ming reaches the finals”

3 words?

姚明	进入	总决赛
YaoMing	reaches	finals

How to do word tokenization in Chinese?

姚明进入总决赛 “Yao Ming reaches the finals”

3 words?

姚明	进入	总决赛
YaoMing	reaches	finals

5 words?

姚	明	进入	总	决赛
Yao	Ming	reaches	overall	finals

How to do word tokenization in Chinese?

姚明进入总决赛 “Yao Ming reaches the finals”

3 words?

姚明	进入	总决赛
YaoMing	reaches	finals

5 words?

姚	明	进入	总	决赛
Yao	Ming	reaches	overall	finals

7 characters? (don't use words at all):

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

this is what linguists do, with a standardized set of glossing conventions (Leipzig rules)

Word tokenization / segmentation

In Chinese it's common to just treat each character as a token. This makes the **segmentation** step is simple.

In other languages (like Thai and Japanese), more complex word segmentation is required.

- The standard algorithms are neural sequence models trained by supervised machine learning.

Words and Frequency

Zipf's Law

Zipfian hypotheses

Last class:

Zipf's hypothesis:
Shorter words are more frequent because languages maximize efficiency: they assign common meanings to words that take less effort to produce.

This class:

Zipf's law: The frequency of a word is inversely proportional to its frequency ranking.

Zipf's Law

The frequency of a word is proportional to the inverse of its rank.

$$f(r) \propto \frac{1}{r^{\alpha}}$$

where r is the frequency rank of the word

The most frequent word in a corpus has rank 1

1. the
2. a
3. is
4. in

$$f(in) = \frac{1}{4^{\alpha}}$$

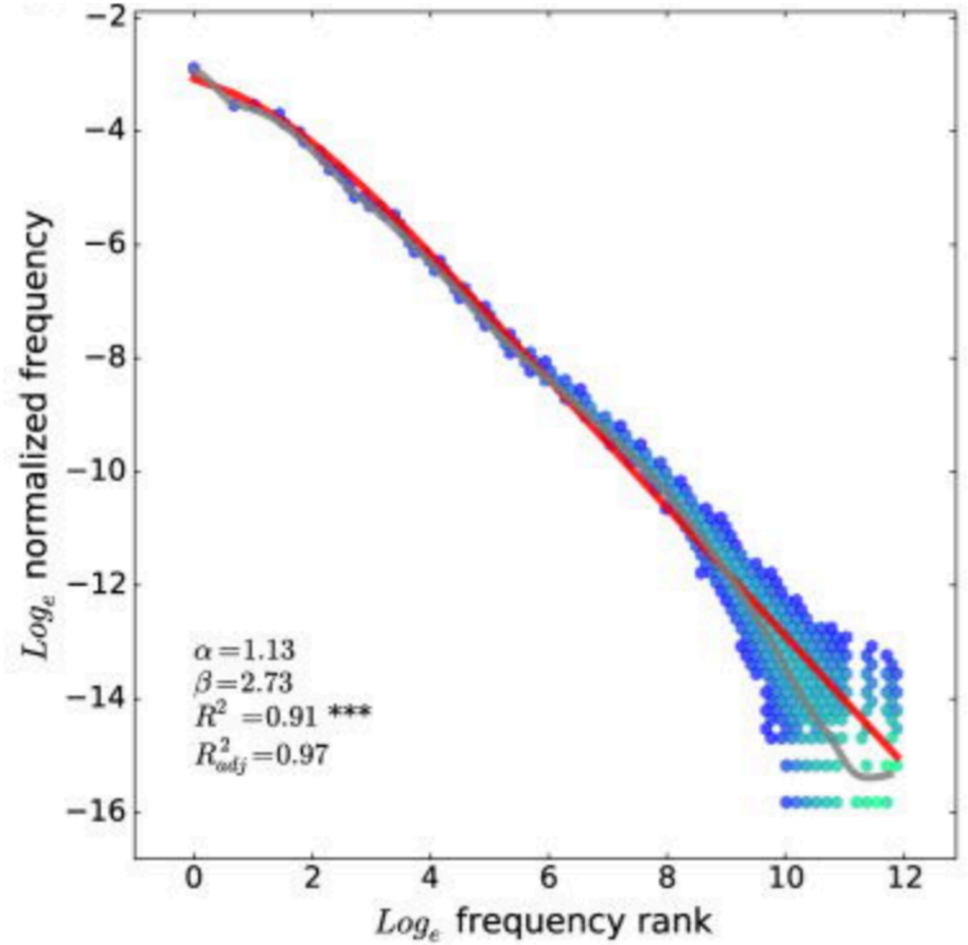
$$f(the) = \frac{1}{1^{\alpha}}$$

$\alpha = 1$

Zipf's Law

Piantadosi (2014)

Here, frequency and frequency rank are calculated on two separate halves of the American National Corpus.

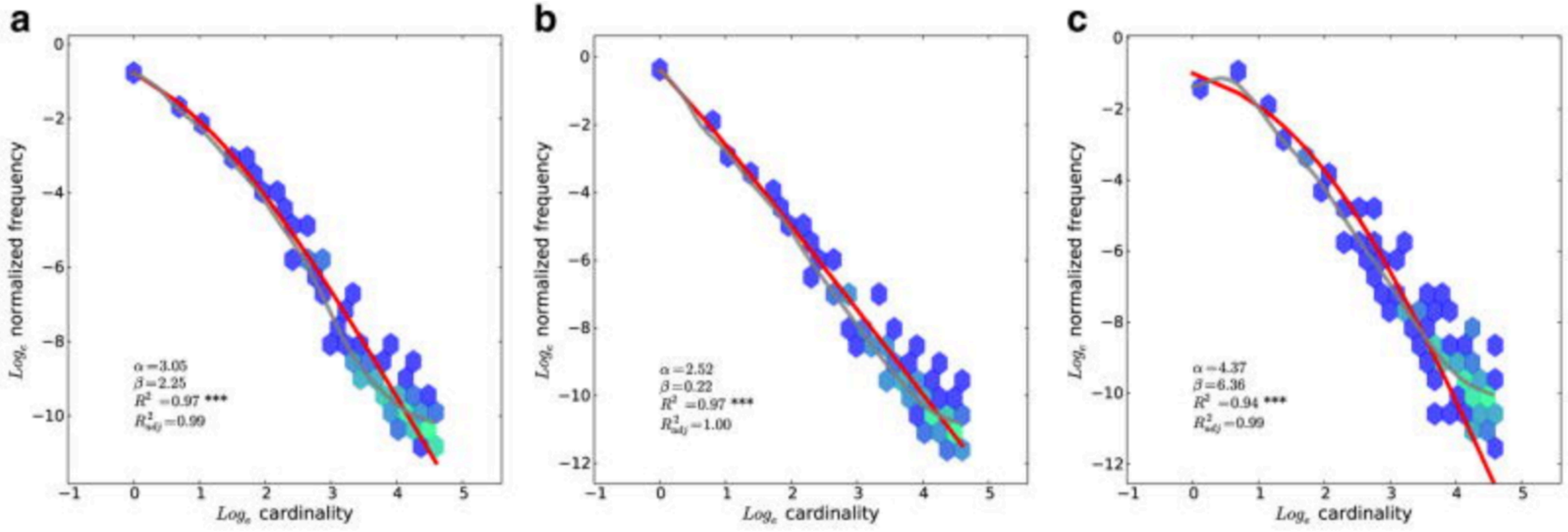


Proposed explanations for Zipf's Law

- Semantics: there are cross-linguistically stable relationships between frequency and meaning.

water
hand
sun
moon

Counter: Zipf's Law holds within domains



Power law frequencies for number words (“one,” “two,” “three,” etc.) in English (a), Russian (b), and Italian (c), taken from Piantadosi (2014)

Plus: Zipf's Law holds in artificial languages

Proposed explanations for Zipf's Law

- Memory: since Zipf's Law holds in artificial language learning experiments, maybe it is due to constraints on human memory

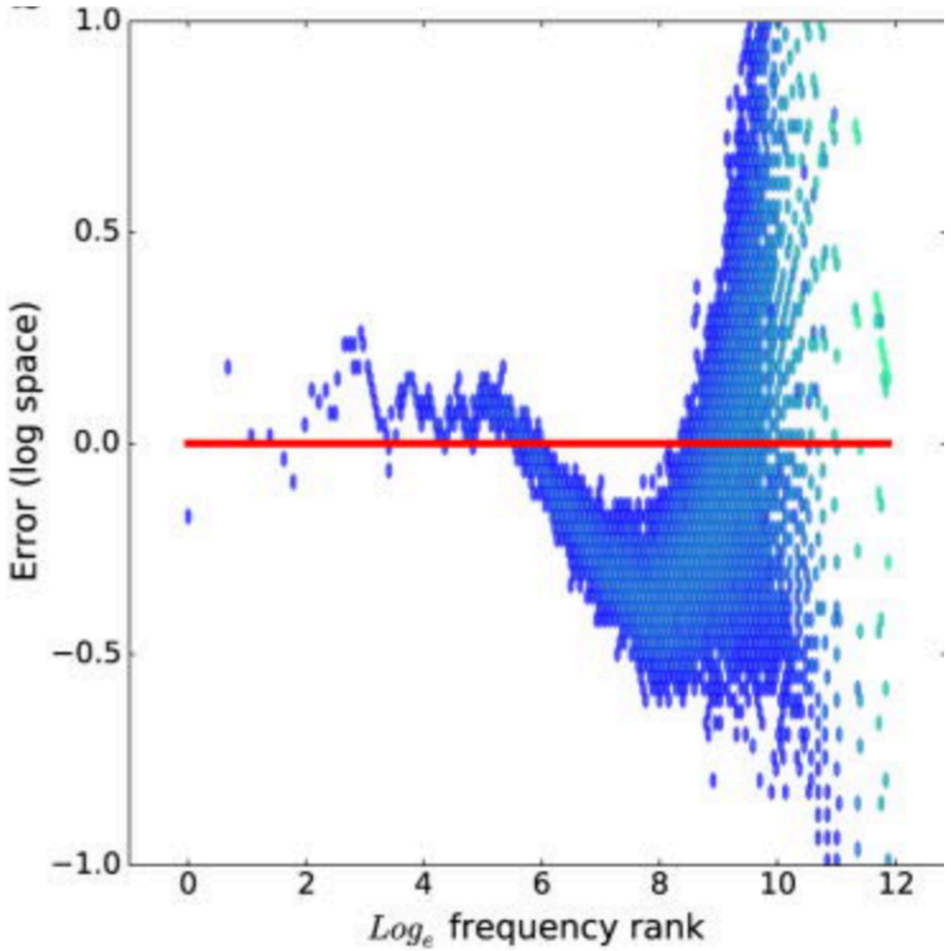
Zipf's Law holds for other human systems

- Distribution of instructions in computer architecture
- Token sequences in programming languages
- Music

Complications

Piantadosi (2014)

Some regularities in differences between predicted frequency and actual frequency.



Basic Text Processing

Byte Pair Encoding

Another option for text tokenization

Instead of

- white-space segmentation
- single-character segmentation

Use the data to tell us how to tokenize.

Subword tokenization (because tokens can be parts of words as well as whole words)

Subword tokenization

Three common algorithms:

- **Byte-Pair Encoding (BPE)** (Sennrich et al., 2016)
- **Unigram language modeling tokenization** (Kudo, 2018)
- **WordPiece** (Schuster and Nakajima, 2012)

All have 2 parts:

- A token **learner** that takes a raw training corpus and induces a vocabulary (a set of tokens).
- A token **segmenter** that takes a raw test sentence and tokenizes it according to that vocabulary

Byte Pair Encoding (BPE) token learner

Let vocabulary be the set of all individual characters

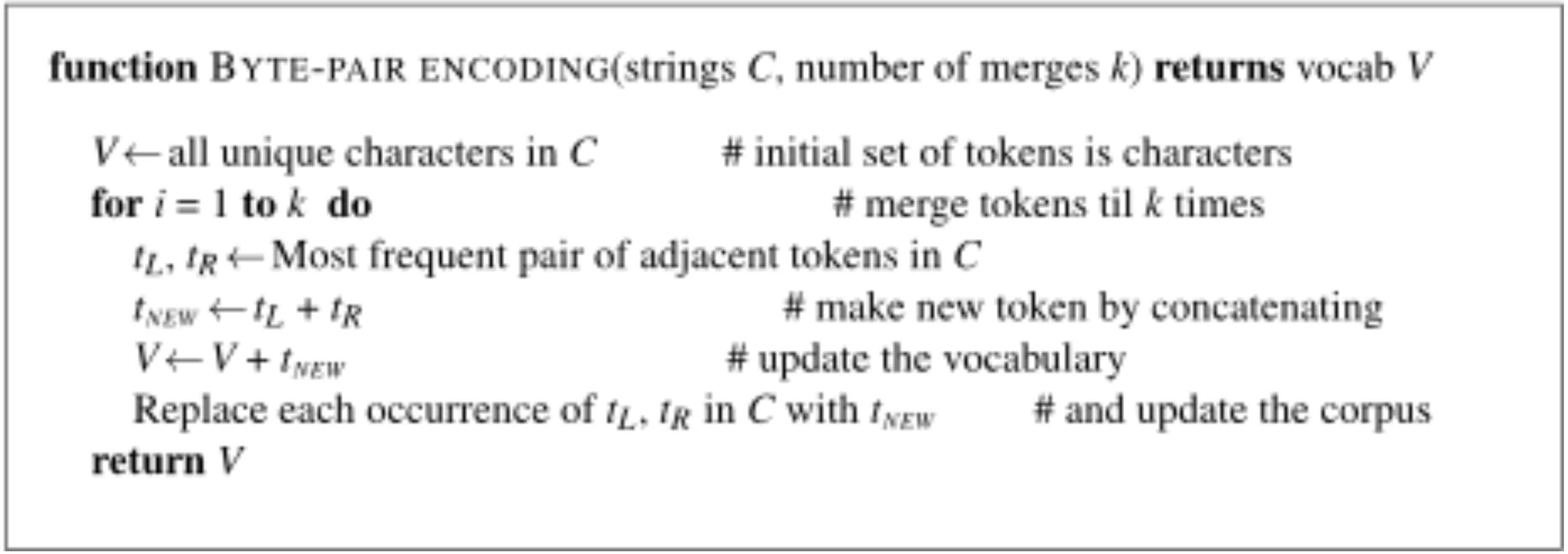
= {A, B, C, D,…, a, b, c, d….}

Repeat:

- Choose the two symbols that are most frequently adjacent in the training corpus (say 'A', 'B')
- Add a new merged symbol 'AB' to the vocabulary
- Replace every adjacent 'A' 'B' in the corpus with 'AB'.

Until *k* merges have been done.

BPE token learner algorithm



```
function BYTE-PAIR ENCODING(strings $C$, number of merges $k$) returns vocab $V$

    $V \leftarrow$ all unique characters in $C$            # initial set of tokens is characters
    for $i = 1$ to $k$ do                                  # merge tokens til $k$ times
        $t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in $C$
        $t_{NEW} \leftarrow t_L + t_R$                     # make new token by concatenating
        $V \leftarrow V + t_{NEW}$                         # update the vocabulary
        Replace each occurrence of $t_L, t_R$ in $C$ with $t_{NEW}$    # and update the corpus
    return $V$
```

Byte Pair Encoding (BPE) Addendum

Most subword algorithms are run inside space-separated tokens.

So we commonly first add a special end-of-word symbol '__' before space in training corpus

Next, separate into letters.

BPE token learner

Original (very fascinating🙄) corpus:

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

Add end-of-word tokens, resulting in this vocabulary:

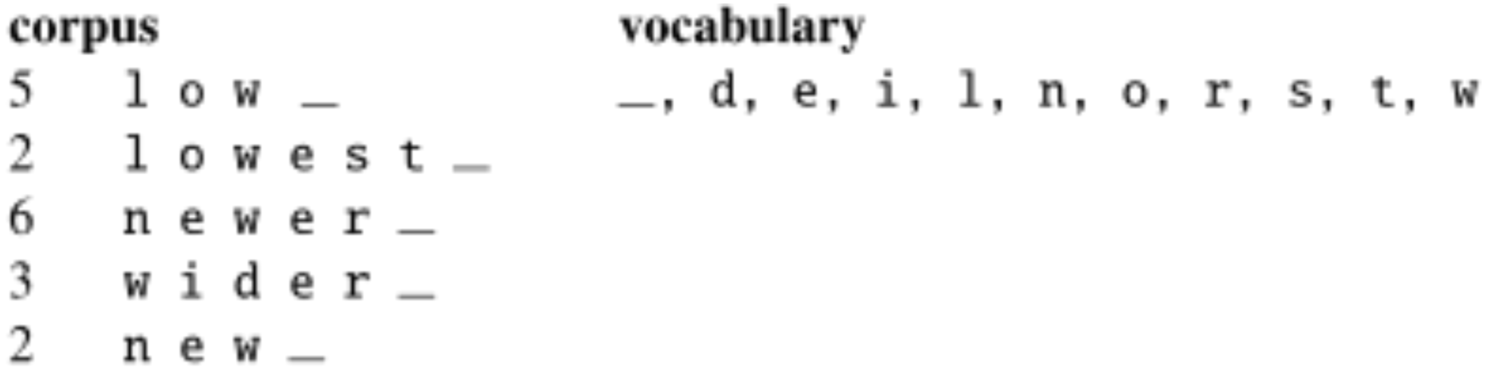
lo : 7	er_ : 9
w_ : 7	wer : 6
ow : 7	
ne : 8	

vocabulary

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

↑

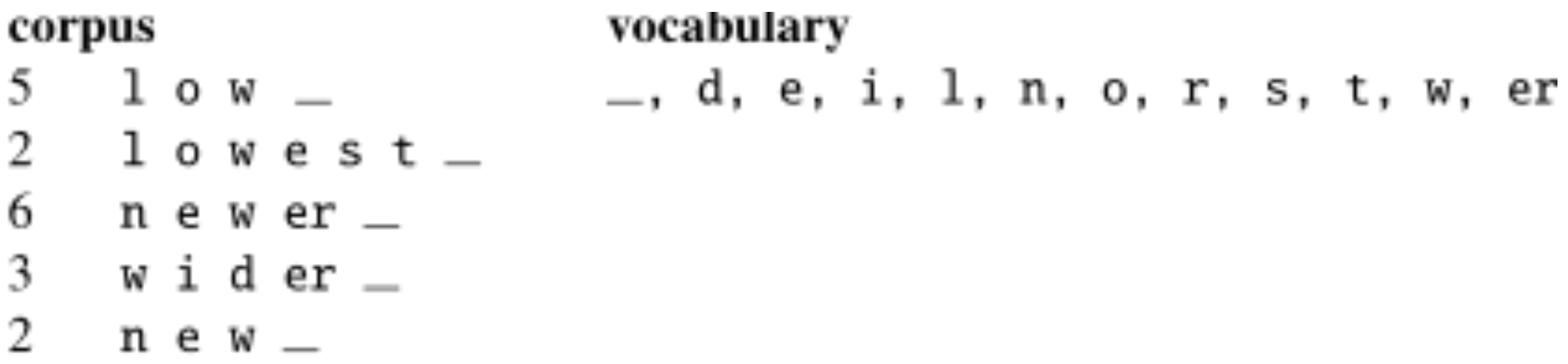
BPE token learner



corpus		vocabulary
5	l o w _	_, d, e, i, l, n, o, r, s, t, w
2	l o w e s t _	
6	n e w e r _	
3	w i d e r _	
2	n e w _	

Merge e r to er:

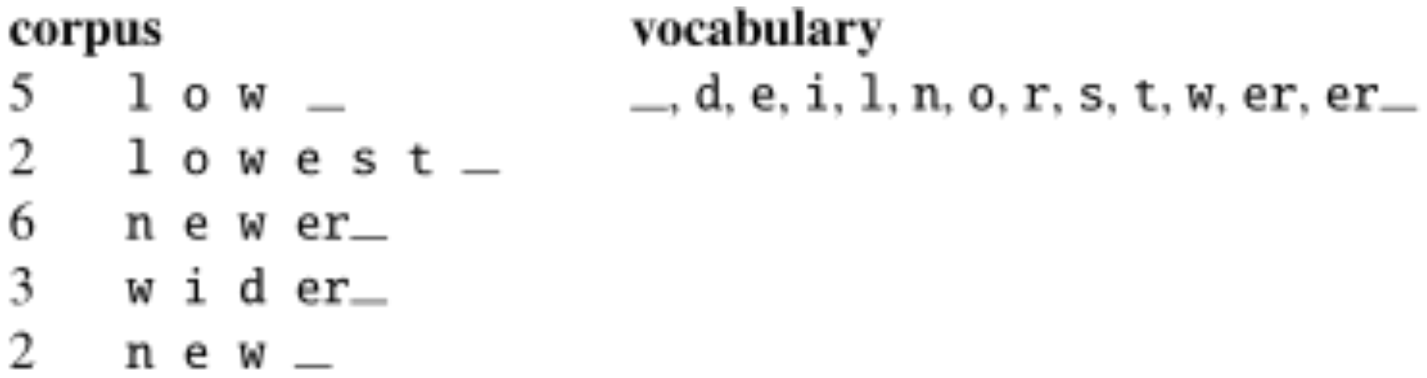
BPE token learner



corpus		vocabulary
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er
2	l o w e s t _	
6	n e w er _	
3	w i d er _	
2	n e w _	

Merge er _ to er_:

BPE token learner



corpus		vocabulary
5	l o w _	_, d, e, i, l, n, o, r, s, t, w, er, er_
2	l o w e s t _	
6	n e w er_	
3	w i d er_	
2	n e w _	

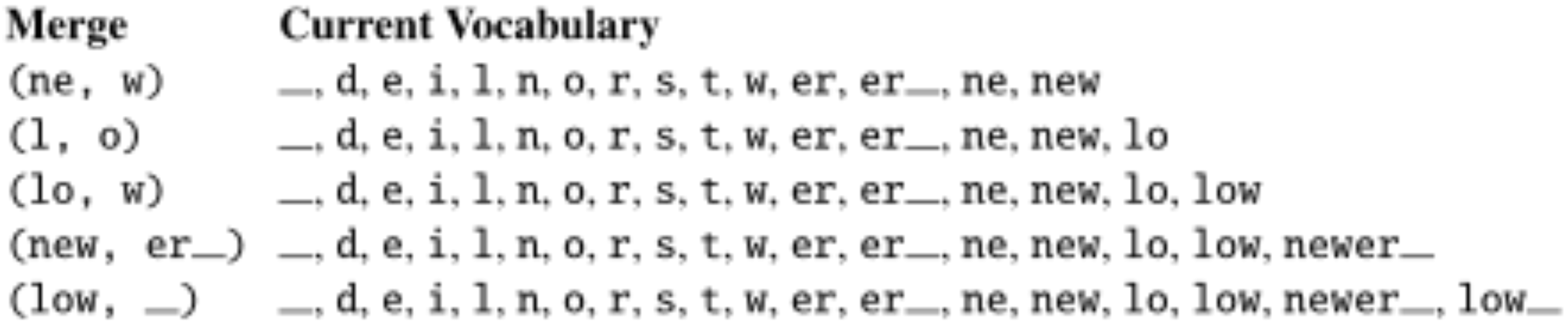
Merge n e to ne:

BPE token learner

What is the next merge?

BPE token learner

Next merges:



Merge	Current Vocabulary
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

BPE token segmenter algorithm

On the test data, run each merge learned from the training data:

- Greedily
- In the order we learned them
- (test frequencies don't play a role)

So: merge every e r to er, then merge er _ to er_, etc.

Result:

- Test set "n e w e r _" would be tokenized as a full word
- Test set "l o w e r _" would be two tokens: "low er_"

Properties of BPE tokens

Usually include frequent word and frequent subwords

- Tokens often morphemes like *-est* or *–er*

Basic Text Processing

Word Normalization and other issues

Case folding

Applications like IR: reduce all letters to lower case

- Since users tend to use lower case
- Possible exception: upper case in mid-sentence?
  - e.g., ***General Motors***
  - ***Fed*** vs. ***fed***
  - ***SAIL*** vs. ***sail***

For sentiment analysis, MT, Information extraction

- Case is helpful (***US*** versus ***us*** is important)

Sentence Segmentation

!, ? mostly unambiguous but **period** "." is very ambiguous

- Sentence boundary
- Abbreviations like Inc. or Dr.
- Numbers like .02% or 4.3

Common algorithm: Tokenize first: use rules or ML to classify a period as either (a) part of the word or (b) a sentence-boundary.

- An abbreviation dictionary can help

Sentence segmentation can then often be done by rules based on this tokenization.