Basic Text Processing

- Words and Corpora

How many words?

- "I do uh main- mainly business data processing"
  - Fragments, filled pauses ⍰ count or not?
- "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 ⍰ count as one or two?
  - **Wordform**: the full inflected surface form
    - cat and cats = different wordforms

How many words?

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 < β < .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 (a folder of text, e.g. Google?)

- 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 dimension like

- **Language**: 7097 languages in the world
- **Variety**, like African American Language varieties.
  - AAL 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, non-fiction, scientific articles, Wikipedia
- **Author Demographics**: writer's age, gender, race, socioeconomic status, etc.

Basic Text Processing

- Word tokenization

Text Normalization

- Every NLP task requires text normalization:
  1. Tokenzing (segmenting) words
  2. Normalizing word formats
  3. Segmenting sentences

Simple Tokenization in UNIX

- (Inspired by Ken Church's UNIX for Poets.)
- Given a text file, output the word tokens and their frequencies

```
tr -sc 'A-Za-z' '\n' < shakes.txt
     | sort
     | uniq -c
```

Change all non-alpha to newlines

Sort in alphabetical order

Merge and count each type

```
1945 A
  72 AARON
  19 ABBESS
   5 ABBOT
 ... ...
```

```
25 Aaron
 6 Abate
 1 Abates
 5 Abbess
 6 Abbey
 3 Abbot
.... …
```

The first step: tokenizing

```
tr -sc ’A-Za-z’ ’\n’ < shakes.txt | head
```

```
THE
SONNETS
by
William
Shakespeare
From
fairest
creatures
We
...
```

The second step: sorting

```
tr -sc ’A-Za-z’ ’\n’ < shakes.txt | sort | head
```

```
A
A
A
A
A
A
A
A
A
...
```

More counting

- Merging upper and lower case

```
tr 'A-Z' 'a-z' < shakes.txt | tr -sc 'A-Za-z' '\n' | sort | uniq -c
```

- Sorting the counts

```
tr 'A-Z' 'a-z' < shakes.txt | tr -sc 'A-Za-z' '\n' | sort | uniq -c | sort -n
-r
```

```
23243 the
22225 i
18618 and
16339 to
15687 of
12780 a
12163 you
10839 my
10005 in
8954  d
```

What happened here?

Tokenization without spaces

- Chinese, Japanese, Thai, don't use spaces to separate words

Word tokenization in Chinese

- Chinese words are composed of characters called **hanzi**
- Each one represents a meaning unit called a morpheme.
- Each word has on average 2.4 of them.
- 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”

- 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
-
-

Basic Text Processing

- Word tokenization

Basic Text Processing

- Word Normalization and other issues

Word Normalization

- Putting words/tokens in a standard format
    - U.S.A. or USA
    - uhhuh or uh-huh
    - Fed or fed
    - am, is be, are

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)

Lemmatization

- Represent all words as their shared root, = dictionary headword form:
  - *am, are, is → be*
  - *car, cars, car's, cars' → car*
  - Spanish quiero (‘I want’), quieres (‘you want’) → querer ‘want'
- *He is reading detective stories → He be read detective story*

Lemmatization is done by Morphological Parsing

- Morphemes:
  - The small meaningful units that make up words
  - **Stems**: The core meaning-bearing units
  - **Affixes**: Parts that adhere to stems, often with grammatical functions
- Morphological Parsers:
  - Parse *cats* into two morphemes *cat* and *s*
  - Parse Spanish *amaren* ('if in the future they would love') into morpheme *amar* 'to love', and the morphological features *3PL* and *future subjunctive*.

Stemming

- Reduce terms to stems, chopping off affixes crudely

This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written notes.

Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with the singl except of the red cross and the written note
.

Porter Stemmer

- Based on a series of rewrite rules run in series
  - A cascade, in which output of each pass fed to next pass
- Some sample rules:

ATIONAL	$\rightarrow$	ATE	(e.g., relational $\rightarrow$ relate)
ING	$\rightarrow$	ϵ	if stem contains vowel (e.g., motoring $\rightarrow$ motor)
SSES	$\rightarrow$	SS	(e.g., grasses $\rightarrow$ grass)

Basic Text Processing

- Byte Pair Encoding tokenization

A third option for word segmentation

- Use the data to tell us how to tokenize.
- **Subword tokenization** (because tokens are often parts of words)
- Can include common morphemes like *-est* or *-er*.
  - (A morpheme is the smallest meaning-bearing unit of a language; *unlikeliest* has morphemes *un-*, *likely*, and *-est*.)

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)

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 training corpus (say ‘A’, ‘B’),
  - adds a new merged symbol ‘AB’ to the vocabulary
  - replace every adjacent ’A’ ’B’ in 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)

- Most subword algorithms are run inside white-space separated tokens.
- So first add a special end-of-word symbol '__' before whitespace 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 and segment:

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 _	

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

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 _	

BPE

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

Merge er _ to er_

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_

BPE

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_

Merge n e to ne

corpus

5	l o w _
2	l o w e s t _
6	ne w er_
3	w i d er_
2	ne w _

vocabulary

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

BPE

The next merges are:

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_