Natural Language Processing

Jim Martin -- Lecture 2

CSCI 5832

NLP@CU

Shocking

NYTIMES.COM

Electric Eels Hunt in Packs, Shocking Prey and Scientists

The behavior, used by wolves and orcas to run down fast prey, is rarely seen i…

Today

- Words
- Corpora
- Vocabularies
- Text normalization
- Subword units

Complex Morphology

Many languages require complex morphological segmentation/analysis

- Turkish
- Uygarlastiramadiklarimizdanmissinizcasina
- `(behaving) as if you are among those whom we could not civilize'
- Uygar `civilized' + las `become'

  + tir `cause' + ama `not able'

  + dik `past' + lar 'plural'

  + imiz 'p1pl' + dan 'abl'

  + mis 'past' + siniz '2pl' + casina 'as if'

Complex Morphology

- Morphological analysis takes a surface form and returns the stem with a set of morphological features.
  - Cats → Cat+PL   Ate → eat+PAST
- Generation goes in the other direction.
- Capturing the underlying morphology of a language is very very hard. Typically, involving a combination of finite state transducers and probabilities
- For many languages, like English and Chinese, you can get away with simpler approaches.

HW 1 Part 1: 50 Points

- How many words do you know in your native language?
  - Due Monday 8/30 by 11:59PM
  - Submit via Canvas
  - Your answer and a writeup explaining your answer.
    - No longer than necessary. 2-3 pages should suffice.
    - Long enough to say something interesting
    - PDF; follow the naming convention specified on the Canvas assignment page

HW 1 Part 1

- How many words do you know in your native language?
  - In your answer clearly address
    - “how many”
    - “words”
    - “know”

Corpora and Vocabularies

Vocabulary

- What is a vocabulary (or lexicon)?
  - All the words in a language?
    - All the words in a comprehensive dictionary?
  - Or some subset?
    - The words needed for a particular application?
      - How many?
      - How do we determine the right list?
      - Is it fixed or will we be adding new words?

Corpora

- Words don't appear out of nowhere.
- We typically generate vocabularies from collections of representative text for a possible domain or application.
- <u>But</u> texts are produced by specific writer(s), at a specific times, in a variety of a specific languages, for specific functions. All of which combine combinatorially.

Dimensions of Corpora

- Language
  - 7097 languages in the world (not all have written forms)
- Language variety
  - Dialects, creoles, pidgins, etc.
- Code switching
  - Mixing language in the same utterance

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
- Demographics
  - writer's age, gender, race, socioeconomic status, etc.
- Medium
  - Spoken, written, captioned, w/ video

Corpora Metrics

N = number of instances or tokens in a corpus

V = vocabulary = set of unique types

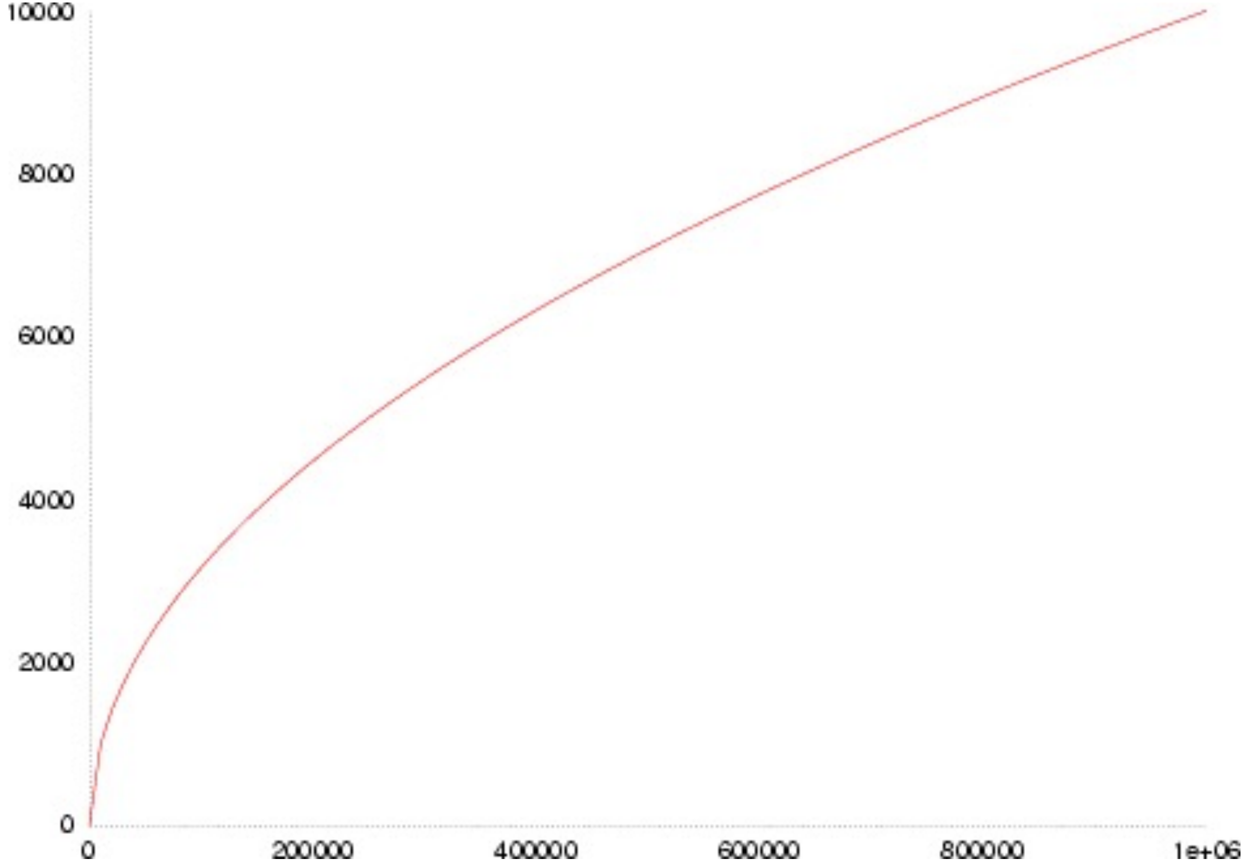
$|V|$ is the size of the vocabulary

	Tokens = N	Types = $|V|$
Switchboard corpus (phone conversations)	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

Heaps' Law

- Heaps Law says that the size of the vocabulary grows somewhat faster than the square root of the corpus size $$|V| = kN^\beta$$
- Where *k* and $\boldsymbol{\beta}$ are free variables. Usually k is between 10 and 100 and $\boldsymbol{\beta}$ is between .6 and .7
- Which really means that the rate of growth of the vocabulary tails off as the corpus grows but never completely flattens out.

Heaps' Law



10000
8000
6000
4000
2000
0
0
200000
400000
600000
800000
1e+06

|V|

N

Building a Vocabulary

- Let’s say we’re handed some large text collection representative of some application domain.
    - Like building a Shakespeare generator
- How do we pull the words out from the text to form a vocabulary?

Simple Tokenization in UNIX

- Given a text file, output all the word types and their associated frequencies in a given text corpus
  - Inspired by Ken Church's UNIX for Poets.

- Unix has many commands to deal with basic text processing operations
  - Original Unix designers <u>cared a lot</u> about text processing

Notebook

Issues in Tokenization

Of course, it is never really that easy. There are lots of complications.

- `Finland's capital` → `Finland Finlands Finland' s`
- `what're, I'm, isn't` → `What are, I am, is not`
- `Hewlett-Packard` → `Hewlett Packard`
- `state-of-the-art` → `state of the art`
- `Lowercase` → `lower-case lowercase lower case`
- `San Francisco` → one token or two?
- m.p.g., PhD. → ??

Tokenization: Language Issues

- French
  - *L'ensemble* → one token or two?
    - *L* ? *L’* ? *Le* ?
- German noun compounds
  - *Plastikwasserflaschenhalter*
  - ‘plastic water bottle holder’

Tokenization: language issues

- Chinese has no spaces between words
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida
- Japanese allows intermingled alphabets

Case Study:
Word Segmentation in Chinese

- Chinese words are composed of characters
  - Characters are generally 1 syllable and 1 morpheme
  - Average word length is 2.4 characters

姚明进入总决赛

Option 1

Syntax/Semantics driven segmentation

姚明进入总决赛

姚明　　进入　　总决赛

YaoMing reaches finals

Option 2

More fine-grained segmentation

姚明进入总决赛

姚 明 进入 总 决赛

Yao Ming reaches overall finals

Option 3

Since all the characters have meanings just use them.

姚明进入总决赛

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

Large Vocabularies

- How do we make sure we’re dealing with all the high frequency words that Zipf’s law predicts
- While still providing a way to deal with out-of-vocabulary (OOV) terms
- And still keep the vocabulary size reasonable?

Subword Tokenization

- Use **subword tokenization** to find words and common subwords empirically
  - Let the data tell us what the words are
- 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)
  - **WordPieces** (Schuster and Nakajima, 2012)
- All have 2 parts:
  - A token **learner** that takes a raw training corpus and induces a vocabulary.
  - A token **segmenter** that takes an input and tokenizes it according to a vocabulary.
    - Words present the vocabulary are left alone (unsegmented)
    - OOV words are broken into optimal sequences of words and subwords.

Byte Pair Encoding (BPE)

Let initial vocabulary be the set of all individual characters

= {A, B, C, D,…,a, b, c, d…., 0-9, etc.}

- Repeat:
  - choose the two symbols that are most frequently adjacent in training corpus (say ‘A’, ‘B’),
  - add 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 with initial white-space separated tokens.
- So first add a special end-of-word symbol '__' before whitespace in training corpus
- Next, separate tokens into letters

BPE Token Learner

Original "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

Original "corpus"

low _

lowest _

*lowest_*

*newer_*

*newer_*

*newer_*

*wider_*

BPE Token Learner

Original “corpus”

l o w _

l o w e s t _

l o w e s t _

n e w e r _

n e w e r _

n e w e r _

w i d e r _

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 _	

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		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_

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 _	

BPE

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

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

BPE

Continuning 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_

BPE token segmentation algorithm

- On test data
  - White-space separate
  - If character sequence is in the vocab just leave it alone
  - If its OOV
    - Run each merge learned from the training data:
    - Greedily
    - In the order we learned them
- 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_"

BPE

- With BPE (and other subword approaches) there are no out of vocabulary words. Every word can be decomposed into a sequence of known vocabulary items (sequences of words and subwords, or worst case, characters).

BERT Vocabulary

- All modern language models (BERT, GPT, T5 and their variants) and MT systems make use of relatively small vocabularies derived using one of the popular subword unit algorithms.
- Typically reported at around 30k entries. This is largely chosen for computational efficiency reasons.
- The original BERT vocabulary was generated from the “Books” corpus and an English Wikipedia dump.

HW 1 Part 2

- In this part of the HW you'll explore a generic BERT vocabulary, with a particular focus on that 30k size. We're interested in how many words does BERT really know?
- How does BERT's vocabulary stack up against the kind of considerations we've been discussing for people?

Notebook

HW 1 Part 2: 50 Points

- How many words does BERT really know?
  - Due Wednesday 9/1 by 11:59PM
  - Submit via Canvas
  - Your answer and a writeup explaining your answer.
    - No longer than necessary. 2-3 pages should suffice.
    - Long enough to say something interesting
    - PDF; follow the naming convention specified on the Canvas assignment page

Next Time

- Chapter 3. N-Gram language modeling