UNIVERSITÀ DEGLI STUDI DI NAPOLI
PARTHENOPE

Natural Language Processing

Corpora and text processing

LESSON 4

prof. Antonino Staiano

M.Sc. In ''Machine Learning e Big Data'' - University Parthenope of Naples

What is a Corpus?

- NLP algorithms are most useful when they apply across many languages
- *A corpus (plural corpora), is a body of utterances, as words or sentences, assumed to be* ***representative of*** *and used for lexical, grammatical, or other linguistic analysis*

Corpora in NLP

- To understand and model how language works, we need empirical evidence
  - Ideally, natural-occurring corpora serve as realistic samples of a language
- Aside from linguistic utterances, corpus datasets include metadata
  - Collateral information about where the language comes from, such as author, date, topic, publication
- Of particular interest are corpora with linguistic annotations, where humans have read the text and marked categories or structures describing their syntax and/or meaning

Example of corpora

- Focusing on English; most released by the Linguistic Data Consortium (LDC):
  - **Brown**: 500 texts, 1M words in 15 genres. POS-tagged. SemCor subset (234K words) labelled with WordNet word senses
  - **WSJ**: 6 years of Wall Street Journal; subsequently used to create Penn Treebank, PropBank, and more! Translated into Czech for the Prague Czech-English Dependency Treebank
  - **ECI**: European Corpus Initiative, multilingual
  - **BNC**: 100M words; balanced selection of written and spoken genres.
  - **Redwoods**: Treebank aligned to wide-coverage grammar; several genres.
  - **Gigaword**: 1B words of news text.
  - **AMI**: Multimedia (video, audio, synchronized transcripts).
  - **Google Books N-grams**: 5M books, 500B words (361B English).
  - **Flickr 8K**: images with NL captions
  - **English Visual Genome**: Images, bounding boxes ⇒ NL descriptions

Markup

- There are several common markup formats for structuring linguistic data, including XML, JSON, CoNLL-style (one token per line, annotations in tab-separated columns)
- Some datasets, such as WordNet and PropBank, use custom file formats
  - Many libraries (such as NLTK) provides friendly Python APIs for reading many corpora so one doesn't have to worry about this

Corpus size

- How large the corpus should be?
  - There is no specific answer to this question
  - The size of the corpus depends upon the purpose for which it is intended as well as on some practical considerations as follows
    - Kind of query anticipated from the user
    - The methodology used by the users to study the data
    - Availability of the source of data
    - With the advancement in technology, the corpus size also increases

Year	Name of the Corpus	Size (in words)
1960s - 70s	Brown and LOB	1 Million words
1980s	The Birmingham corpora	20 Million words
1990s	The British National corpus	100 Million words
Early 21st century	The Bank of English corpus	650 Million words

Sources of variability in corpora

- **Language**: 7097 languages in the world
- **Variety**, like African American Language varieties
  - AAE Twitter posts might include forms like "*iont*" *(I don't)*
- **Genre**
  - *Newswire, fiction, scientific articles, Wikipedia*
- **Author Demographics**: writer's age, gender, ethnicity, SES
- **Code switching**
  - E.g., Spanish/English
    - 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:)]

Text Normalization

Text Normalization

- Text normalization is the process of transforming a text into some predefined standard form, and could consist of several tasks
- There is no all-purpose normalization procedure rather, it depends on
  - What type of text is being normalized
  - What type of NLP task needs to be carried out afterwards
- Text normalization is also important for applications other than NLP, such as text mining and WEB search engines

Text normalization

- Tokenization
  - Separating out (tokenizing) words from running text
- Lemmatization
  - Determining that two words have the same root, despite their *surface differences*
- Stemming
  - A simpler lemmatization in which we just strips suffixes from the end of the words
- Sentence segmentation
  - Breaking up a text into individual sentences

Words

- Before any text processing takes place, one must agree on what counts as a word
- Let’s consider the following sentence from Brown Corpus
  - “*He stepped out into the hall, was delighted to encounter a water brother.*”
    - 13 words not considering punctuation marks as words, 15 otherwise
    - It depends on the task. Punctuation is critical for finding boundary of things (commas, period, colons) and for identifying some aspects of meaning (?, !, “ ”)
    - For *part-of-speech tagging*, *parsing* or *speech synthesis*, punctuation marks treated as separate words
- Other corpora (spoken language) don’t have punctuation but further complicate defining what is a word …

How many words in a sentence?

- Let's look at one sample **utterance** (the spoken correlate of a sentence)
  - "*I do uh main- mainly business data processing*"
  - Fragments (*main-*), filled pauses (*uh*) (**disfluencies**)
    - Words or not?
- *They* and *they* are the same word?
  - Yes, in some applications (speech recognition) no in others (e.g., POS tagging, NER)
- "*Seuss's cat in the hat is different from other cats!*"
  - cat and cats same lemma (cat) but different wordforms
  - Lemma: set of lexical forms with same stem, part of speech, word sense
  - Wordform: the full inflected or derived surface form of a word

How many words in a corpus?

- N = number of tokens (i.e., number of running words)
- V = vocabulary = set of types, $|V|$ is size of vocabulary
  - Types = number of different words in a corpus
- Heaps Law = Herdan's Law = $|V| = kN^{\beta}$ where often .67 < **β** < .75
  - vocabulary size grows faster than 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

Text Normalization

Word Tokenization

- Space-based tokenization is a very simple way to tokenize
  - For languages that use space characters between words
    - Arabic, Cyrillic, Greek, Latin, etc., based writing systems
  - Segment off a token between instances of spaces
- Unix tools for space-based tokenization
  - The "tr" command
  - Inspired by Ken Church's UNIX for Poets
  - Given a text file, output the word tokens and their frequencies

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

- Many languages (like Chinese, Japanese, Thai) don't use spaces to separate words!

- How do we decide where the token boundaries should be?

Word tokenization in Chinese

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

- Example
  - Consider the following Chinese sentence and possible tokenizations

姚明进入总决赛
“Yao Ming reaches the finals”

姚 明 进入 总 决赛
Yao Ming reaches overall finals

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

Character tokenization

So, in Chinese it's common to just treat each character (zi) as a token

- the **segmentation** step is very simple
- For Japanese and Thai the character is too small a unit, and algorithms for word segmentation are required
- Standard segmentation algorithms for these languages use neural sequence models
  - This is related to sequence labelling

Another option for text tokenization

- **Use the data** to tell us how to tokenize instead of
  - white-space segmentation
  - single-character segmentation
- **Subword tokenization** (because tokens can be parts of words as well as whole words)

Subword tokenization

- Subword tokenization schemes have two parts
  - the **token learner** takes a raw training corpus and induces a set of tokens, called vocabulary
  - the **token segmenter** takes a raw test sentence and segments it into the tokens in the vocabulary
- Three algorithms are widely used
  - byte-pair encoding (BPE) tokenization
  - unigram tokenization
  - WordPiece tokenization

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
  - k is a hyperparameter

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
  - a special end-of-word symbol '__' before space in training corpus is first added
  - Next, separate into letters

BPE token learner

- A very simple 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 the following vocabulary:

corpus representation		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 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_

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 token learner

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 token learner

- 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 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)
- E.g., 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 includes frequent words
- And frequent subwords
  - Which are often morphemes like *-est* or *–er*
- A **morpheme** is the smallest meaning-bearing unit of a language
  - *unlikeliest* has 3 morphemes *un-*, *likely*, and *-est*

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*
- Sometimes it might be useful to keep both versions of the text data
  - Case is helpful (US versus us is important) for sentiment analysis, MT, Information extraction

Stop words

- **Stop words removal** includes getting rid of
  - common articles
  - pronouns
  - prepositions
  - Coordinations (and, or, but)
- Stop word removal heavily depends on the task at hand, since it can wipe out relevant information

Lemmatization

- Lemmatization has the objective of reducing a word to its base form, also called lemma, therefore grouping together different forms of the same word
- Example
  - Am, are, is -> be
  - Car, cars, car’s, cars’ -> car
  - Italian *voglio* (‘*I want*’), *vuoi* (‘*you want*’) -> *volere* (‘*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'

Stemming

- Stemming refers to the process of slicing a word with the intention of removing affixes
- Stemming is problematic in the linguistic perspective, since it sometimes produces words that are not in the language, or else words that have a different meaning
  - Much more commonly used in IR than NLP. Porter and Snowball stemmers very popular (rule based)
  - For low-resource languages statistical stemmers are also an option
- Example
  - Arguing -> argu, flies -> fli
  - Playing -> play, caring-> car
  - News -> new

Stemming

- Reduce the terms to stems, coarsely cutting the affixes

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:

$$
\begin{array}{rcll}
\text{ATIONAL} & \rightarrow & \text{ATE} & \text{(e.g., relational} \rightarrow \text{relate)} \\
\text{ING} & \rightarrow & \epsilon & \text{if stem contains vowel (e.g., motoring} \rightarrow \text{motor)} \\
\text{SSES} & \rightarrow & \text{SS} & \text{(e.g., grasses} \rightarrow \text{grass)}
\end{array}
$$

Languages' complex morphology

- Handling complex morphology is necessary for many languages
  - e.g., the Turkish word:
    - 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'

Sentence segmentation

- Text normalization also includes **sentence segmentation**: breaking up a text into individual sentences
- This can be done using cues like periods, question marks, or exclamation points
- Period is very ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Common algorithms tokenize first using 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