INF5830 – 2015 FALL
NATURAL LANGUAGE PROCESSING

Jan Tore Lønning, Lecture 4, 10.9

2

Working with texts

From bits to meaningful units

Today:

- Reading in texts
- Character encodings and Unicode
- Word tokenization and regular expressions
- Sentence segmentation
- Tagged text (and taggers)

NLTK

Chapter 1

- from nltk.book import *
- Loads a set of objects of type nltk.text.Text
- These objects come with many NLTK-defined methods

Chapter 2

- NLTK corpora
- Preprocessed:
  - Word tokenized
  - Sentence segmented
  - Etc.
- Associated corpus reader:
  - Various methods for accessing preprocessed text

Python

From file

- infile = open(<filepath>,'r')
- file= infile.read()
  - or
- line = infile.readline()
- infile.close()
- Simple methods
- Raw text

From URL

- from urlib import urlopen
- url =``http://... ´´
- raw = urlopen(url).read()
- type(raw) is string

Today:

- Reading in texts
- Character encodings and Unicode
- Word tokenization and regular expressions
- Sentence segmentation
- Tagged text (and taggers)

Character encodings

- All we compute are binary numbers 0s and 1s
- How can that be text?

- ``In the beginning there was ASCII´´
  - (Not really, about 1963)
- 7 bits – 128 characters
- What about ``Æ, Ø, Å´´ – and the rest?

USASCII code chart

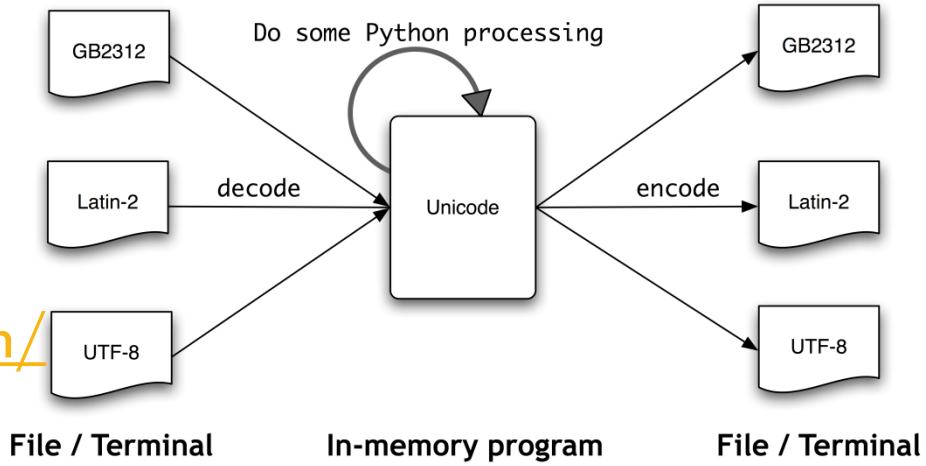
b_7 b_6 b_5 →					0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
Bits b_4	b_3	b_2	b_1	Column → / Row ↓	0	1	2	3	4	5	6	7
0	0	0	0	0	NUL	DLE	SP	0	@	P	`	p
0	0	0	1	1	SOH	DC1	!	1	A	Q	a	q
0	0	1	0	2	STX	DC2	"	2	B	R	b	r
0	0	1	1	3	ETX	DC3	#	3	C	S	c	s
0	1	0	0	4	EOT	DC4	$	4	D	T	d	t
0	1	0	1	5	ENQ	NAK	%	5	E	U	e	u
0	1	1	0	6	ACK	SYN	&	6	F	V	f	v
0	1	1	1	7	BEL	ETB	'	7	G	W	g	w
1	0	0	0	8	BS	CAN	(	8	H	X	h	x
1	0	0	1	9	HT	EM	)	9	I	Y	i	y
1	0	1	0	10	LF	SUB	*	:	J	Z	j	z
1	0	1	1	11	VT	ESC	+	;	K	[	k	{
1	1	0	0	12	FF	FS	,	<	L	\	l	|
1	1	0	1	13	CR	GS	–	=	M	]	m	}
1	1	1	0	14	SO	RS	.	>	N	^	n	~
1	1	1	1	15	SI	US	/	?	O	_	o	DEL

ASCII-Extensions

- Latin1:
  - 8 bit, 256 characters
  - room for: Æ, Ø, Å
  - default on IFI Linux until 2012
- Shortcomings:
  - Different encodings for different languages
    - Choose the correct one
    - Problems for comparing languages
  - Some languages have more than 256 characters
  - Mathematical symbols?
  - Vendor dependent choices

Unicode

- Goal: a universal character set for all languages
- A unique code point (number) for each character
  - (across languages)
- Room for more than 1 mill. different code points
- Injective mappings from known character encodings into unicode (and back again)
- http://unicode-table.com/en/



Unicode in Python

Python 2

- In [18]: a='Bodø‘ (utf-8)
- In [19]: a (Python string)
- Out[19]: 'Bod\xc3\xb8'
- In [20]: len(a)
- Out[20]: 5
- In [21]: print a
- Bodø
- In [22]: b=a.decode('utf-8')
- In [23]: b (Python unicode string)
- Out[23]: u'Bod\xf8'
- In [24]: print(b)
- Bodø
- In [25:] len(b)
- Out[25]: 4

Python 3

- In [18]: a='Bodø‘
- In [19]: a (
- Out[19]: 'Bodø'
- In [20]: len(a)
- Out[20]: 4
- In [21]: print(a)
- Bodø

All strings are unicode

Unicode in Python

Python 2

- `In [30:] import codecs`
- `In [31]: f = codecs.open(path, encoding='latin2')`

Python 3

- `In [31]: f = open(path, encoding='latin2')`

NLTK and unicode

- Newer NLTK uses unicode for strings (behind the curtain)
- For Python 3, this is seamless
- For Python 2, there are some additional u's
- One might have to consult both ed.1 and ed.2 of the NLTK book

Today:

- Reading in texts
- Character encodings and Unicode
- Word tokenization and regular expressions
- Sentence segmentation
- Tagged text (and taggers)

Tokenization

- After reading in one gets a string of characters, e.g.
  - 'For example, this isn't a well-formed example.'
- We want to split it into (a list of) words
- What should the result be?
  1. |For|example|,|this|is|n't|a|well-formed| example|.|
  2. |For example,|this|isn't|a|well- |formed| example.|
  3. |for|example|this|is|not|a|well-formed|example|
     - (1) is Penn TreeBank-style (PTB)
     - (2) is English Resource Grammar-style (ERG)

Tokenization - alternatives

1. |For|example|,|this|is|n't|a|well-formed| example|.|
2. |For example,|this|isn't|a|well- |formed| example.|
3. |for|example|this|is|not|a|well-formed|example|

- Punctuation:
  (1) separate tokens, (2) part of words, (3) remove
- isn't, doesn't etc.: (1) split, (2) keep, (3) normalize
- Multiword expressions: (2) one token, (1,3) one token per word
- Hyphens: when to split? How?
- Case folding (lowercasing) or not?
- In addition, there are special constructions like decimal numbers, urls, etc.

Tokenization - alternatives

- Which alternative depends partly of what the text should be used for.
- An alternative to lowercasing is true-casing:
  - Train a classifier to take decisions, cf.
  - Brown sugar isn't healthy but it is better than white sugar.
  - Brown lives next to Jones.

How to tokenize

- The cheapest way in Python:
  - words = s.split()
- If we prefer ‘example’ to ‘example.’ we could proceed
  - clean_words = [w.strip(‘.,:;?!’) for w in words]
- To keep ‘.’ as a token, you must be more refined.
- In NLTK for English, we can use the word_tokenize
  - words = nltk.word_tokenize(s)
  - How does this tokenize the ``for example”-sentence?

Kleene regular expressions

Kleene regular expressions:

- Corresponds to finite-state automata

Regular expression	Describes the language
$\varnothing$	$L(\varnothing) = \varnothing$
ε	$L(\varepsilon) = \{ \varepsilon \}$
a, for alle $a \in A$	$L(a) = \{ a \}$
If R and S are regular expressions:	
(R | S)	$L(R \mid S) = L(R) \cup L(S)$
(R T)	$L(R\ T) = L(R)L(T)$
(R*)	$L(R^*) = L(R)^*$ concatenation of 0 or more expressions in L(R)

Applied regular expressions

In addition:

Regular expression	Interpretation
^	Beggining of line
$	End of line
\b	Word boundary
\B	Word non-boundary
.	Any character
\t \n \r	tab – newline – carriage return
[...]	Any of, e.g. [abc]
[... – ...]	All characters in the span, e.g. [a-zA-Z]
[^...]	Any character not in [...]

and more

Applied regular expressions

- Typical use: searching for strings where a part matches the reg.ex
  - (This is not the same as describing a language and may yield some unexpected results)
- In python
  - import re
  - re.search(´^[ghi][mno][jlk][def]$´, text)

Regular expressions for tokenization

- re.findall(r'\w+|\Sw*, raw)
- What does the reg.ex say?

Today:

- Reading in texts
- Character encodings and Unicode
- Word tokenization and regular expressions
- Sentence segmentation
- Tagged text (and taggers)

Sentence segmentation

- Split a text into sentences.
- ``How difficult could that be?'':
  - ``Split at: . ! ?''
- What about e.g. abbreviations?
  - ``Okay, not after abbreviations''
- What about abbreviations at the end of a sentence, etc.
- What about embedded sentences like ``what about embedded sentences?''
- A non-trivial problem

Sentence segmentation

- Is normally done with some sort of machine learning.
- We will not study this at this point
- How well is *nltk.sentence_tokenize()* handling the examples from last slide?

Today:

- Reading in texts
- Character encodings and Unicode
- Word tokenization and regular expressions
- Sentence segmentation
- Tagged text (and taggers)

Tagged text

- [('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'), ('completely', 'RB'), ('different', 'JJ')]
- Each token in the text is assigned a part of speech (POS) tag
- There is a finite defined set of tags
- A tagger is a process which assigns tags to the words in the text

Universal POS tag set (NLTK)

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Various POS tag set

- NLTK:
  - Universal POS Tagset, 12 tags, (see 2.ed of the book)
  - Simplified POS tagset, 19 tags, (1.ed, defunct)
- Brown tagset:
  - Original: 87 tags
  - Versions with extended tags <original>-<more>
- Penn treebank tags: 35+9 punctuation tags

Nouns

NN	Noun, sing. or mass	*llama*
NNS	Noun, plural	*llamas*
NNP	Proper noun, singular	*IBM*
NNPS	Proper noun, plural	*Carolinas*

Penn treebank

NN	(common) singular or mass noun	time, world, work, school, family, door
NN$	possessive singular common noun	father's, year's, city's, earth's
NNS	plural common noun	years, people, things, children, problems
NNS$	possessive plural noun	children's, artist's parent's years'
NP	singular proper noun	Kennedy, England, Rachel, Congress
NP$	possessive singular proper noun	Plato's Faulkner's Viola's
NPS	plural proper noun	Americans Democrats Belgians Chinese Sox
NPS$	possessive plural proper noun	Yankees', Gershwins' Earthmen's
NR	adverbial noun	home, west, tomorrow, Friday, North,
NR$	possessive adverbial noun	today's, yesterday's, Sunday's, South's
NRS	plural adverbial noun	Sundays Fridays

Brown

How do these two tokenize

``the queen's castle'' ?

NN	Noun, sing. or mass	*llama*
NNS	Noun, plural	*llamas*
NNP	Proper noun, singular	*IBM*
NNPS	Proper noun, plural	*Carolinas*

Penn treebank

NN	(common) singular or mass noun	time, world, work, school, family, door
NN$	possessive singular common noun	father's, year's, city's, earth's
NNS	plural common noun	years, people, things, children, problems
NNS$	possessive plural noun	children's, artist's parent's years'
NP	singular proper noun	Kennedy, England, Rachel, Congress
NP$	possessive singular proper noun	Plato's Faulkner's Viola's
NPS	plural proper noun	Americans Democrats Belgians Chinese Sox
NPS$	possessive plural proper noun	Yankees', Gershwins' Earthmen's
NR	adverbial noun	home, west, tomorrow, Friday, North,
NR$	possessive adverbial noun	today's, yesterday's, Sunday's, South's
NRS	plural adverbial noun	Sundays Fridays

Brown

Verbs

VB	Verb, base form	*eat*
VBD	Verb, past tense	*ate*
VBG	Verb, gerund	*eating*
VBN	Verb, past participle	*eaten*
VBP	Verb, non-3sg pres	*eat*
VBZ	Verb, 3sg pres	*eats*

Penn treebank

VB	verb, base form	make, understand, try, determine, drop
VBD	verb, past tense	said, went, looked, brought, reached kept
VBG	verb, present participle, gerund	getting, writing, increasing
VBN	verb, past participle	made, given, found, called, required
VBZ	verb, 3rd singular present	says, follows, requires, transcends

Brown

Adjectives + Prepositions

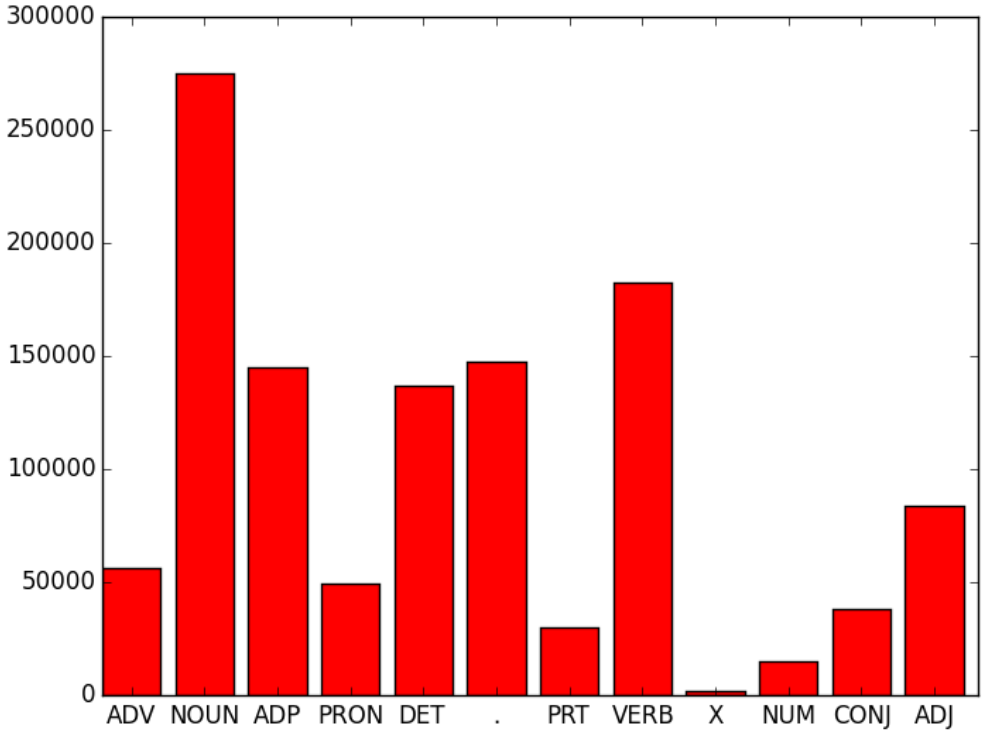
IN	preposition	of in for by to on at
JJ	adjective	
JJR	comparative adjective	better, greater, higher, larger, lower
JJS	semantically superlative adj.	main, top, principal, chief, key, foremost
JJT	morphologically superlative adj.	best, greatest, highest, largest, latest, worst

Brown

Universal POS tagset (NLTK)

Tag	Meaning	English Examples
ADJ	adjective	*new, good, high, special, big, local*
ADP	adposition	*on, of, at, with, by, into, under*
ADV	adverb	*really, already, still, early, now*
CONJ	conjunction	*and, or, but, if, while, although*
DET	determiner, article	*the, a, some, most, every, no, which*
NOUN	noun	*year, home, costs, time, Africa*
NUM	numeral	*twenty-four, fourth, 1991, 14:24*
PRT	particle	*at, on, out, over per, that, up, with*
PRON	pronoun	*he, their, her, its, my, I, us*
VERB	verb	*is, say, told, given, playing, would*
.	punctuation marks	*. , ; !*
X	other	*ersatz, esprit, dunno, gr8, univeristy*

Distribution of universal POS in Brown



``Corrected" from lecture 1

Cat	Freq
ADV	56 239
NOUN	275 244
ADP	144 766
NUM	14 874
DET	137 019
.	147 565
PRT	29 829
VERB	182 750
X	1 700
CONJ	38 151
PRON	49 334
ADJ	83 721

Ambiguity…

- …is what makes natural language processing…
  - …hard/fun
- POS:
  - noun or verb: *eats shoots and leaves*
  - verb or preposition: *like*
- Word sense:
  - *bank, file, …*
- Structural:
  - *She saw a man with binoculars.*
- Sounds

POS ambiguity

- The most frequent word forms are most ambiguous
- Even though most word types are unambiguous, more than 50 % of the tokens in a corpus may be ambiguous.
- The degree of ambiguity depends on the tag set.

Tagged corpora

- In a tagged corpora the word occurrences are disambiguated
- Possible to explore the occurrences of the word with the tag, e.g.
  - How often is ``likes'' used as a noun compared to 20 years ago?
- Explore the frequency and positions of tags:
  - When does a determiner occur in front of a verb?
- Good data for training various machine learning tasks:
  - The tags make useful features

Tagger

- A tagger may be used to make (or extend) a tagged corpus
- To tag text can be a first step towards a deeper analysis (Information extraction).

How does a tagger work?

- Various techniques
- Most common: Hidden Markov Model
  - INF4820
- Lately: Deep learning