INF5830 – 2017 FALL
NATURAL LANGUAGE PROCESSING

Jan Tore Lønning, Lecture 3, 4.9

Today

- Recap
- Naive Bayes
  - Bernoulli
  - Multinomial for text classification
- scikit representations
- Smoothing
- Tagged text

Recap

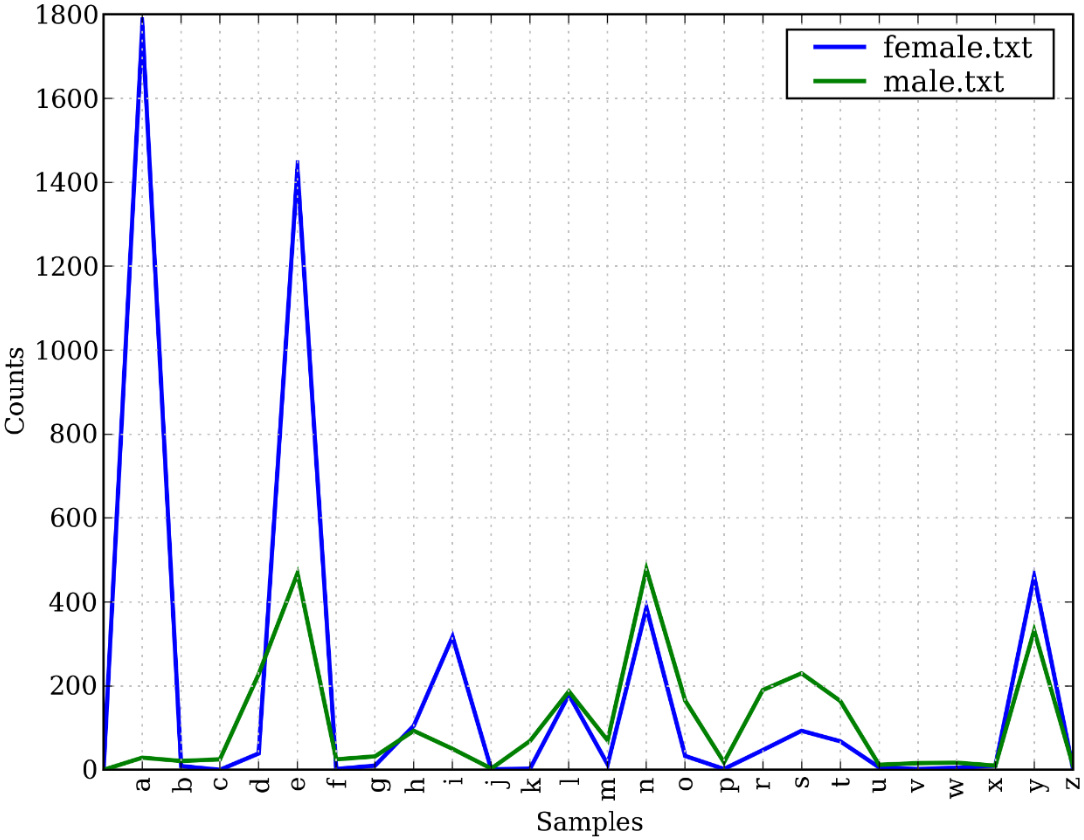
- Classification
- Experimental set-up
- Evaluation

Supervised classification

- A given set of classes, $S=\{s_1, s_2, \ldots, s_k\}$
- A well defined class of objects, O

- Some features $f_1, f_2, \ldots, f_n$
- For each feature: a set of possible values $V_1, V_2, \ldots, V_n$
- The set of feature vectors: $V= V_1\times V_2\times\ldots\times V_n$
- Each object in O is represented by some member of V:
  - Written $(v_1, v_2, \ldots, v_n)$, or
  - $(f_1=v_1, f_2=v_2, \ldots, f_n=v_n)$
- A classifier, γ, can be considered a mapping from V to S

NLTK: names



1800
1600
1400
1200
1000
800
600
400
200
0
Counts
a b c d e f g h i j k l m n o p r s t u v w x y z
Samples
female.txt
male.txt

NLTK-example 2, names

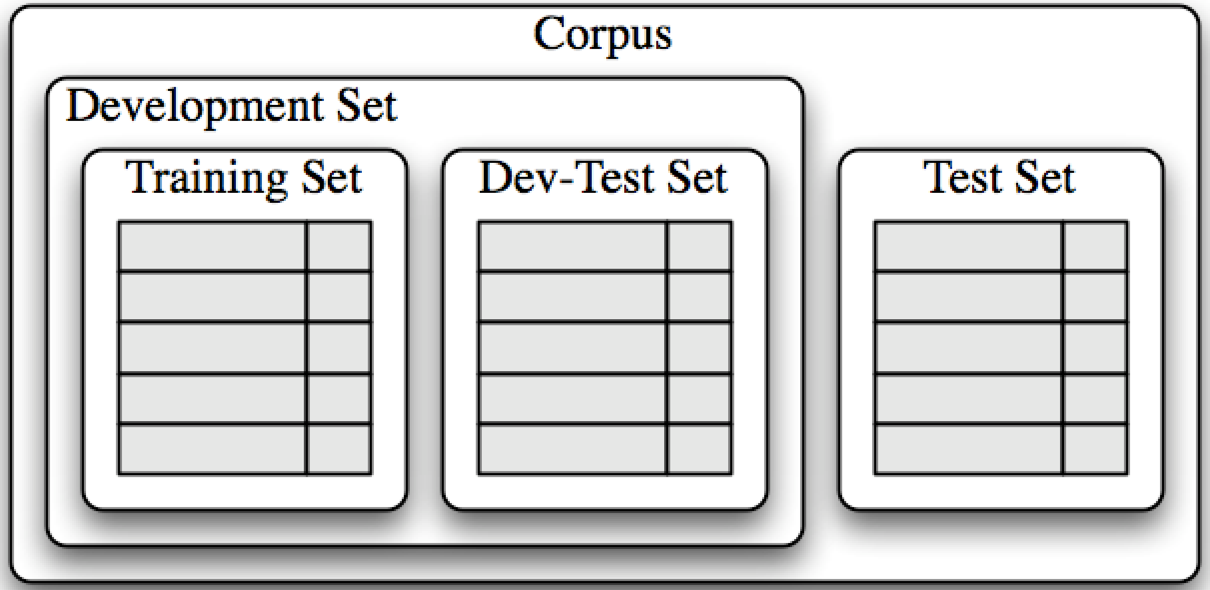

```
In [56]: def gender_features2(name):
   ...:     features = {}
   ...:     features["first_letter"] = name[0].lower()
   ...:     features["last_letter"] = name[-1].lower()
   ...:     for letter in 'abcdefghijklmnopqrstuvwxyz':
   ...:         features["count({})".format(letter)] = name.lower().count(letter)
   ...:         features["has({})".format(letter)] = (letter in name.lower())
```

	first letter	last letter	count(X) X: a-z	has(X) X: a-z	total
Number of features	1	1	26,	26	54
	a-z	a-z	0, 1, 2, …	True, False	
Possible values for each feat.	26	26	infinite	2	

Movie reviews, eample 3

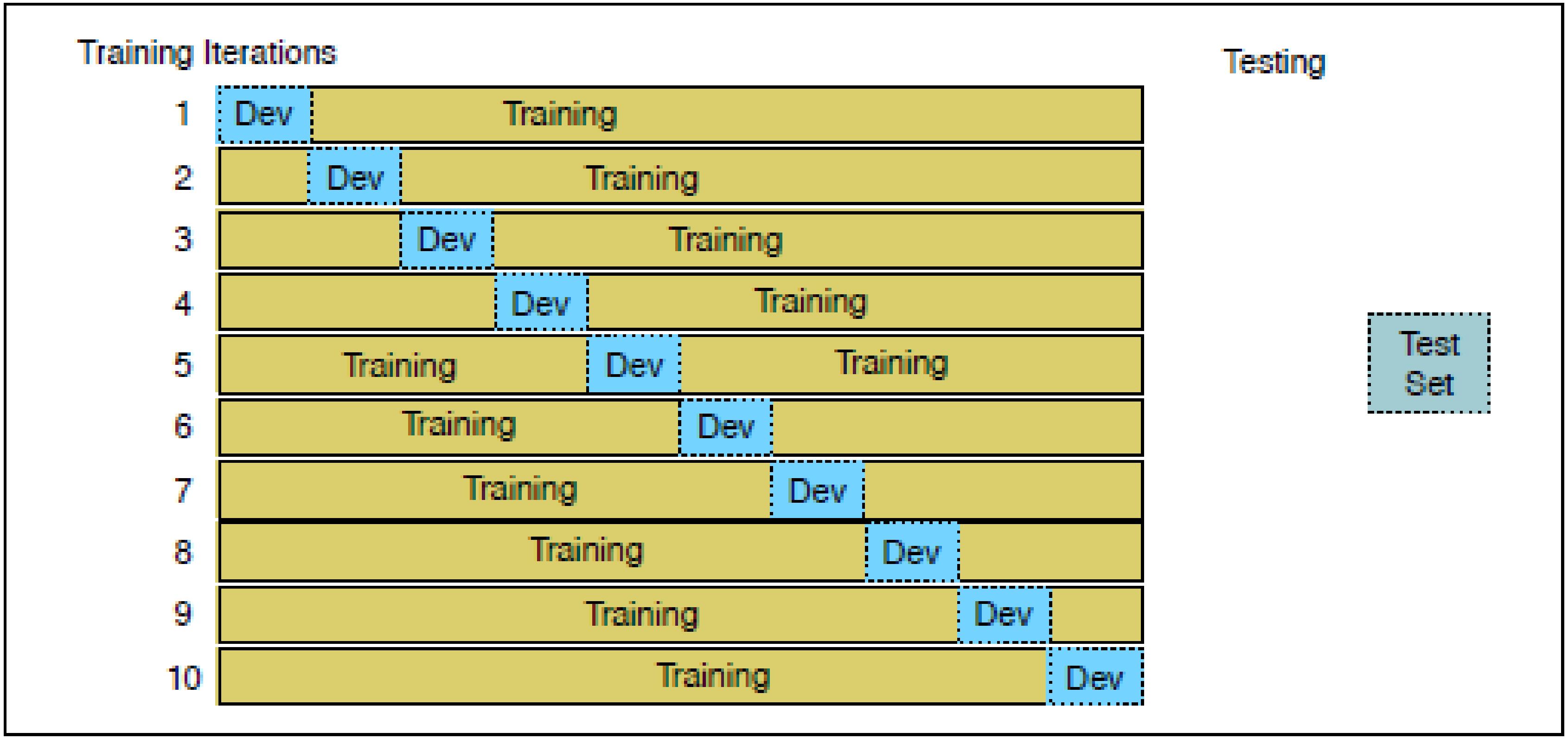
- Two classes: ‘neg’, ‘pos’
- Features’:
  - 2000 most frequent words in corpus
- Values: True/False
  - Don't count number of occs in each document
  - All features (words) not in document gets value “’False”

Set-up for experiments



- Before you start: split into development set and test set.
- Hide the test set
- Split development set into Training and Development-Test set
- Use training set for training a learner
- Use Dev(-Test) for repeated evaluation in the test phase
- Finally test on the test set!

Crossvalidation



- But take away a final test set first!

Evaluation

gold standard labels

		gold positive	gold negative	
system output labels	system positive	**true positive**	**false positive**	**precision** $= \frac{tp}{tp+fp}$
	system negative	**false negative**	**true negative**	
		recall $= \frac{tp}{tp+fn}$		**accuracy** $= \frac{tp+tn}{tp+fp+tn+fn}$

Figure 6.4 Contingency table

- $F_1 = \frac{2PR}{P+R}$

Naive Bayes

Naive Bayes: Decision

- Given an object
  - $\langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle$
- Consider
  - $P(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle)$ for each class s_m
- Choose the class with the largest value, in symbols

$$\operatorname*{arg\,max}_{s_m \in S} P(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle)$$

- i.e. choose the class for which the observations are most likely

Naive Bayes: Model

- Bayes formula
  - $P\left(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle\right) = \dfrac{P\left(\langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle \mid s_m\right) P(s_m)}{P\left(\langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle\right)}$
- Sparse data, we may not even have seen
  - $\langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle$
- We assume (wrongly) independence
  - $P\left(\langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle \mid s_m\right) \approx \prod_{i=1}^{n} P\left(f_i = v_i \mid s_m\right)$
- Putting together
  - $\underset{s_m \in S}{\arg\max}\, P\left(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle\right) \approx \underset{s_m \in S}{\arg\max}\, P(s_m) \prod_{i=1}^{n} P\left(f_i = v_i \mid s_m\right)$

Naive Bayes: Calculation

- $$\arg\max_{s_m \in S} P\left(s_m \mid \left\langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \right\rangle\right) \approx \arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P\left(f_i = v_i \mid s_m\right)$$

- For calculations
  - avoid underflow, use logarithms
  - $$\arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P\left(f_i = v_i \mid s_m\right) =$$
    $$\arg\max_{s_m \in S} \left(\log \left(P(s_m) \prod_{i=1}^{n} P\left(f_i = v_i \mid s_m\right) \right) \right)$$
    $$= \arg\max_{s_m \in S} \left(\log(P(s_m)) + \sum_{i=1}^{n} \log(P\left(f_i = v_i \mid s_m\right)) \right)$$

Naive Bayes, Training 1

- Maximum Likelihood
- $$\hat{P}(s_m) = \frac{C(s_m, o)}{C(o)}$$
  - where $C(s_m, o)$ are the number of occurrences of objects o in class s_m
- Observe what we are doing in statistical terms:
  - We want to estimate the true probability $P(s_m)$ from a set of observations
  - This is similar to estimating properties (parameters) of a population from a sample.

Naive Bayes (Bernoulli): Training 2

- Maximum Likelihood
- $$\hat{P}(f_i = v_i \mid s_m) = \frac{C(f_i = v_i, s_m)}{C(s_m)}$$
  - where $C(f_i=v_i, s_m)$ is the number of occurrences of objects o
    - where the object o belongs to class s_m
    - and the feature f_i takes the value v_i
  - $C(s_m)$ is the number of occurrences belonging to class s_m

The two models

- Bernoulli
  - the standard form of NB
  - NLTK book, Sec. 6.1, 6.2, 6.5
  - Jurafsky and Martin, 2.ed, sec. 20.2, WSD
- Multinomial model
  - For text classification
  - Related to n-gram models
  - Jurafsky and Martin, 3.ed, sec. 7.1, Sentiment analysis
- Both
  - Manning, Raghavan, Schütze, *Introduction to Information Retrieval*, Sec. 13.0-13.3

Multinomial text classification

- Build a language model for each class
- Score the document according to the different classes
- Choose the class with the best score

Multinomial NB: Decision

$$\arg\max_{s_m \in S} P\left(s_m \mid \left\langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \right\rangle\right) \approx \arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P\left(f_i = v_i \mid s_m\right)$$

- In the multinomial model
  - f_i refers to position i in the text
  - v_i refers to the word occurring in this position
- We model the probability of the full texts given the class s_m
- Then we make a simplifying assumption:
  - We assume a word to be equally likely in all positions:

$$\arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P\left(f_i = v_i \mid s_m\right) = \arg\max_{s_m \in S} P(s_m) \prod_{i=1}^{n} P\left(v_i \mid s_m\right)$$

Multinomial NB: Training

- $$\hat{P}(s_m) = \frac{C(s_m, o)}{C(o)}$$
  - where $C(s_m, o)$ is the number of occurrences of objects o in class s_m
- $$\hat{P}(w_i \mid s_m) = \frac{C(w_i, s_m)}{\sum_j C(w_j, s_m)}$$
  - where $C(w_i, s_m)$ is the number of occurrences of word w_i in all texts in class s_m
  - $\sum_j C(w_j, s_m)$ is the total number of words in all texts in class s_m
- Bernoulli counts the number of objects/texts where w_i occurs
- Multinomial counts the number of occurrences of w_i.

Comparison

Bernoulli

- Registers whether a term is present or not
- Considers both
  - The present terms
  - The absent terms
- Suitable for various tasks

Multinomial

- Counts how many times a term is present
- Considers
  - only the present terms
  - Ignores absent terms
- Tailor-made for text classification

Implementation

Doing it ourselves

- Possible to implement Naive Bayes classifiers ourselves
  - (That's not the case for all classifiers)
- Efficiency (and memory space) may be challenging
- Many available implementations. More efficient.
  - E.g. scikit-learn

Available learners

NLTK

- Bernoulli NB
- Decision trees
- (Python inefficient)

Scikit-learn

- Bernoulli NB
- Multinomial NB
- and many, many more
- Much more efficient

Data-representation

NLTK

```
[({'f1': 'a', 'f2': 'z', 'f3': True, 'f4': 5}, 'class_1'),
 ({'f1': 'b', 'f2': 'z', 'f3': False, 'f4': 2}, 'class_2'),
 ({'f1': 'c', 'f2': 'x', 'f3': False, 'f4': 4}, 'class_1')]
```

3 training instances

4 features

class

scikit

```
X_train:
array([[ 1.,  0.,  0.,  0.,  1.,  1.,  5.],
       [ 0.,  1.,  0.,  0.,  1.,  0.,  2.],
       [ 0.,  0.,  1.,  1.,  0.,  0.,  4.]])
```

7 features

3 training instances

```
train_target: ['class_1', 'class_2', 'class_1']
```

classes

One-hot encoding

feature 1				feature 2	
a	b	c		x	y
(1,0,0)	(0,1,0)	(0,0,1)		(1,0)	(0,1)

scikit

X_train:

```
array([[ 1.,  0.,  0.,  0.,  1.,  1.,  5.],
       [ 0.,  1.,  0.,  0.,  1.,  0.,  2.],
       [ 0.,  0.,  1.,  1.,  0.,  0.,  4.]])
```

3 training instances

7 features

train_target: ['class_1', 'class_2', 'class_1']

classes

Converting dictionary

- We can construct the data to scikit directly
- Scikit has methods for converting Python-dictionaries/NLTK-format to arrays

```
train_data = [inst[0] for inst in train]
train_target = [inst[1] for inst in train]
v = DictVectorizer()
X_train=v.fit_transform(train_data)

X_test=v.transform(test_data)
```

1. Constructs (=fit) repr. format
2. Transform

Transform
Use same v as for train

Multinomial NB in scikit

- We can construct the data to scikit directly
- Scikit has methods for converting text to bag of words arrays

```
» train_data=["en rose er en rose",
              "anta en rose er en fiol"]
» v = CountVectorizer()
» X_train=v.fit_transform(train_data)

» print(X_train.toarray())
  [[0 2 1 0 2]
  [1 2 1 1 1]]
```

29

Smoothing

Naive Bayes: Calculation

- When using maximum likelihood estimation

$$\hat{P}(f_i = v_i \mid s_m) = \frac{C(f_i = v_i, s_m)}{C(s_m)}$$

- may become 0
- Then the whole

$$\underset{s_m \in S}{\arg\max}\, P\left(s_m \mid \langle f_1 = v_1, f_2 = v_2, \ldots, f_n = v_n \rangle\right) \approx \underset{s_m \in S}{\arg\max}\, P(s_m) \prod_{i=1}^{n} P\left(f_i = v_i \mid s_m\right)$$

- becomes 0
- Goal to avoid 0-probabilities

Laplace Smoothing

- Also called add-one smoothing
- Just add one to all the counts!
- Very simple
- MLE estimate: $P(w_i) = \frac{c_i}{N}$
- Laplace estimate: $P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$
- Lidstone smoothing: add k: $\hat{P}(w_i) = \frac{c_i+k}{N+kV}$
  - NLTK Naïve Bayes: add 0.5

Smoothing contd.

- Example names, suffixes of 3 letters
  - 7944 names
  - 17576 possible suffixes
  - 1538 of them seen
- Trigrams of words, e.g. Brown
  - Words: 1,161,192
  - Vocabulary: 56,057
  - Possible trigrams: 176,152,802,017,193
  - Seen trigrams: 907,494
- Add 1 gives away too much probablity mass

More advanced smoothing

- There are more advanced methods taking the actual distributions into consideration
- Presented in chapter om language models which we will not consider

Working with texts

From bits to meaningful units

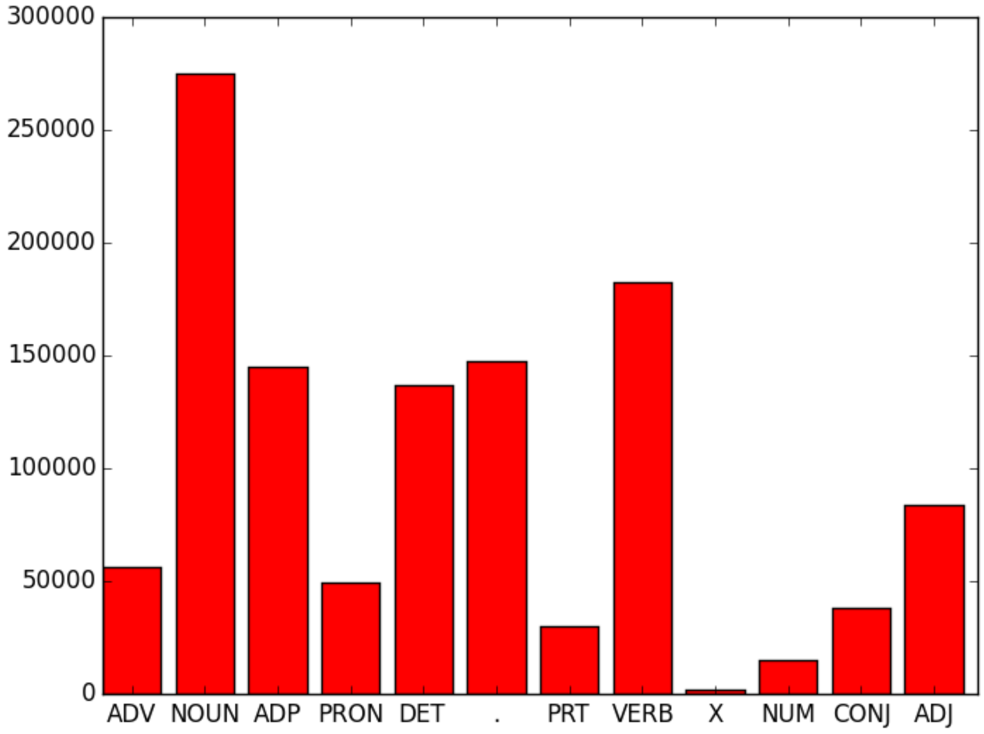
Tagged text

- [('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*

Distribution of universal POS in Brown



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

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

Penn treebank

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

Brown

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

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

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

Summary

- Naive Bayes
  - Bernoulli
  - Multinomial for text classification
- scikit representations
- Smoothing
- Tagged text