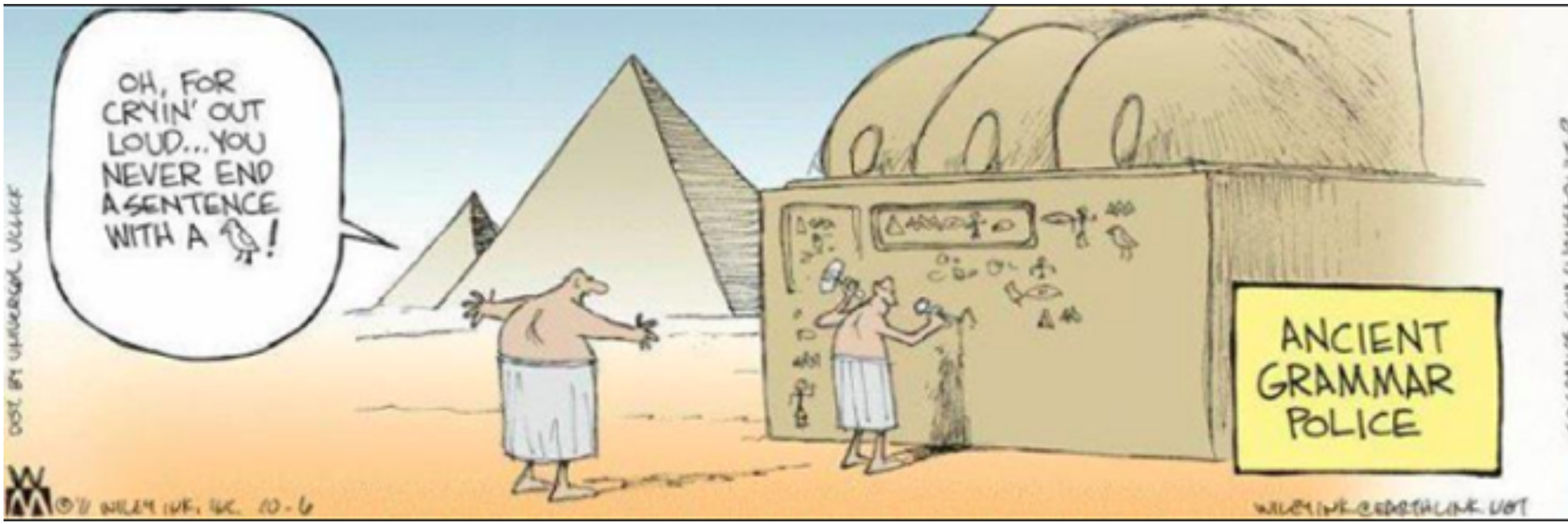
I256:
Applied Natural Language Processing

Marti Hearst
Week 5


OH, FOR CRYIN' OUT LOUD... YOU NEVER END A SENTENCE WITH A !
ANCIENT GRAMMAR POLICE

COURSE CONCEPT MAP

	Word-Level	Phrase-Level	Sentence-Level	Discourse-Level
Segment-ation				
Syntax	Part of Speech Tagging			
Semant-ics				

Statistics

There are MANY Algorithms

- Rule-based POS tagging
- N-gram count-based
- Transformation-based learning (Brill tagger)
- Hidden Markov Models
- Maximum Entropy Models
- Conditional Random Fields
- Multiplicative weight updating learning algorithms for linear functions (used at http://cogcomp.cs.illinois.edu/page/demo_view/POS)

Features for POS Tagging

- **Syntagmatic**: tags of the other words
  - AT JJ NN is common
  - AT JJ VBP is impossible (or unlikely)
- **Lexical**: look at the words
  - The → AT
  - Flour → more likely a noun than a verb
  - A tagger that always chooses the most common tag is 90% correct
    - (often used as the baseline)
- Most taggers use both

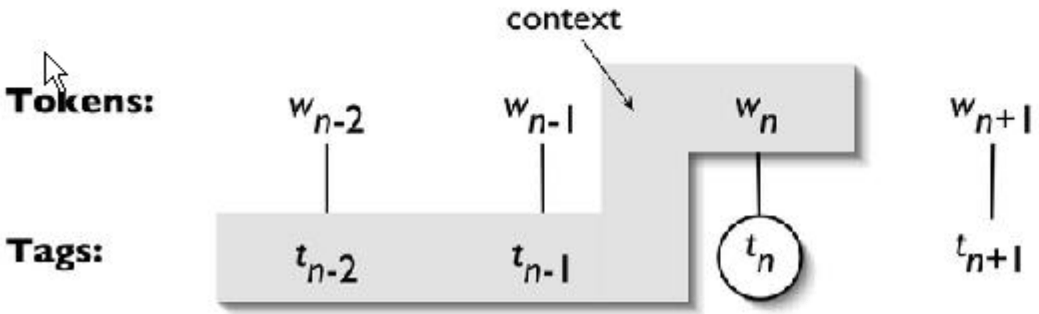
Unigram Tagger

- **Train:** by providing tagged sentence data:
  - Inspect the tag of each word
  - Store the most likely tag in a dictionary for that position in the sentence

```
>>> size = int(len(brown_tagged_sents) * 0.9)
>>> size
4160
>>> train_sents = brown_tagged_sents[:size]
>>> test_sents = brown_tagged_sents[size:]
>>> unigram_tagger = nltk.UnigramTagger(train_sents)
>>> unigram_tagger.evaluate(test_sents)
0.81202033290142528
```

NLTK N-gram Tagging

- An **n-gram tagger** is a generalization of a unigram tagger
  - The context is the current word together with the part-of-speech tags of the *n-1* preceding tokens
- A 1-gram tagger is another term for a unigram tagger:
  - The context used to tag a token is just the text of the token itself.
- 2-gram taggers are also called *bigram taggers*, and 3-gram taggers are called *trigram taggers*.



trigram tagger

NLTK Ngram Tagger Training

__init__(self, n, train=None, model=None, backoff=None, cutoff=0, verbose=False)
(Constructor) [source code]

Train a new `NgramTagger` using the given training data or the supplied model. In particular, construct a new tagger whose table maps from each context `(tag[i-n:i-1], word[i])` to the most frequent tag for that context. But exclude any contexts that are already tagged perfectly by the backoff tagger.

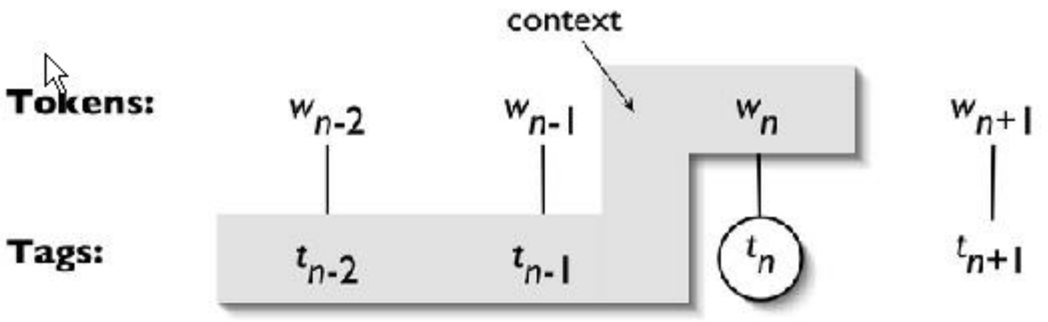
Parameters:

- **`train`** - A tagged corpus consisting of a `list` of tagged sentences, where each sentence is a `list` of `(word, tag)` tuples.
- **`backoff`** - A backoff tagger, to be used by the new tagger if it encounters an unknown context.
- **`cutoff`** - If the most likely tag for a context occurs fewer than `cutoff` times, then exclude it from the context-to-tag table for the new tagger.

Overrides: SequentialBackoffTagger.__init__

Computing the Context

- Unigram
  - Say i=5, n=1
  - Context(tag[i-n:i-1], word[i]) is (tag[4:4],word[5])
- Bigram
  - Say i=5, n=2
  - Context(tag[i-n:i-1], word[i]) is (tag[3:4], word[5])
- Trigram
  - Say i=5, n=3
  - Context(tag[i-n:i-1], word[i]) is (tag[2:4], word[5])



trigram tagger

Why not a 10gram tagger?

Answer: data sparseness

Key Topic: Evaluation

KEY TECHNIQUES

- Looking at Data / Error Analysis
- Evaluating with Training / Development / Test Sets
- Predicting with Sequences (Ngrams / Language models)
- Machine Learning with Feature Creation and Selection
- Word Windows and Context Vectors

Supervised Machine Learning

- Start with annotated corpus
  - Desired input/output behavior
- Training phase:
  - Represent the training data in some manner
  - Apply learning algorithm to produce a system (tagger)
- Testing phase:
  - Apply system to unseen test data
  - Evaluate output

Three Pillars of Statistical NLP

- Corpora (training data)
- Representations (features)
- Learning approach (models and algorithms)

Components of a Proper Evaluation

- Figures(s) of merit
  - The determination of correctness or accuracy.
- Baseline
  - The score achieved by a straightforward, simple algorithm.
- Upper bound
  - The best score possible, given human agreement.
- Tests of statistical significance

Three Laws of Evaluating with Data

Thou shalt not mingle training data with test data

Thou shalt not mingle training data with test data

Thou shalt not mingle training data with test data

Evaluation Metric for Tagging

- Accuracy (used in the tagger evaluation)
  - What percent of the items was correct?
    - In nltk TaggerI class, evaluate() imports nltk.metrics.accuracy

Function Details

[hide private]

accuracy(reference, test)

source code

Given a list of reference values and a corresponding list of test values, return the fraction of corresponding values that are equal. In particular, return the fraction of indices 0<i<=len(test) such that test[i] == reference[i].

Parameters:

- **reference** (list) - An ordered list of reference values.
- **test** (list) - A list of values to compare against the corresponding reference values.

In-Class Exercise

- Train a Back-off Tagger

Evaluation Metrics

- Binary condition (correct/incorrect):
  - Accuracy
- Set-based metrics (illustrated with document retrieval):

	Relevant	Not relevant
Retrieved	A	B
Not retrieved	C	D

  - Precision = A / (A+B)
  - Recall = A / (A+C)
  - F-measure:

$$F = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

Collection size = A+B+C+D
Relevant = A+C
Retrieved = A+B

Tagging accuracies

- Taggers are pretty good on WSJ journal text.
- Other collections can be harder.
- Performance depends on several factors
  - The amount of training data
  - The tag set (the larger, the harder the task)
  - Difference between training and testing corpus
  - Unknown words
    - For example, technical domains

Confusion Matrix

- Useful for looking at Common Errors
- Plot the hypothesized score vs the true (gold) score.

Use a Confusion Matrix to Look for Common Errors

- Common errors [from Toutanova & Manning 00]

	JJ	NN	NNP	NNPS	RB	RP	IN	VB	VBD	VBN	VBP	Total
JJ	0	**177**	**56**	0	**61**	2	5	10	15	**108**	0	488
NN	**244**	0	**103**	0	12	1	1	29	5	6	19	525
NNP	**107**	**106**	0	**132**	5	0	7	5	1	2	0	427
NNPS	1	0	**110**	0	0	0	0	0	0	0	0	142
RB	**72**	21	7	0	0	16	**138**	1	0	0	0	295
RP	0	0	0	0	**39**	0	**65**	0	0	0	0	104
IN	11	0	1	0	**169**	**103**	0	1	0	0	0	323
VB	17	**64**	9	0	2	0	1	0	4	7	**85**	189
VBD	10	5	3	0	0	0	0	3	0	**143**	2	166
VBN	**101**	3	3	0	0	0	0	3	**108**	0	1	221
VBP	5	34	3	1	1	0	2	**49**	6	3	0	104
Total	626	536	348	144	317	122	279	102	140	269	108	3651

NN/JJ NN

official knowledge

VBD RP/IN DT NN

made up the story

RB VBD/VBN NNS

recently sold shares

Confusion Matrix for Bigram Tagger on Brown (train news, test editorial; simplified tags)

Exercise: which tags are most often confused?

		'	''	*	,	.	ADJ	ADV	CNJ	DET	EX	FW	MOD	N	NN	NP	NUM	P	PRO	TO	UH	V	VB+PPO	VBZ	VD	VG	VN	WH	``
	<192>	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
'	.	<19>	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
''	.	.	<382>	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
*	.	.	.	<307>	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
,	.	.	.	.	<2766>	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	<3001>	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
ADJ	.	.	.	.	.	.	<2941>	36	1	29	.	.	.	142	1024	44	.	.	.	.	.	13	.	.	3	1	1	.	.
ADV	.	.	.	.	.	.	45	<1504>	23	136	20	.	.	3	265	.	12	97	.	.	.	1	.	.	.	.	.	2	.
CNJ	.	.	.	.	.	.	2	18	<2866>	31	.	.	.	.	8	.	.	125	.	.	.	.	.	.	.	.	.	9	.
DET	.	.	.	.	.	.	25	65	109	<7387>	.	.	.	15	36	.	.	10	1	.	.	.	.	.	.	.	.	9	.
EX	.	.	.	.	.	.	.	5	.	.	<96>	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
FW	.	.	.	.	.	.	1	.	.	.	.	<8>	.	2	30	3	.	.	.	.	.	.	.	.	.	.	.	.	.
MOD	.	.	.	.	.	.	.	.	.	.	.	.	<929>	4	1	.	.	.	.	.	.	1	.	.	.	.	.	.	.
N	.	.	.	.	.	.	114	6	.	1	.	2	9	<9107>	2887	26	.	4	.	.	1	182	.	59	2	68	7	.	.
NN	.	.	.	.	.	.	.	.	.	.	.	.	.	.	<.>	.	.	.	.	.	.	.	.	.	.	.	.	.	.
NP	.	.	.	.	.	.	35	.	.	2	.	.	.	12	897	<1639>	.	.	.	.	.	.	.	.	.	.	.	.	.
NUM	.	.	.	.	.	.	.	6	.	1	.	.	.	.	95	.	<733>	.	.	.	.	.	.	.	.	.	.	.	.
P	.	.	.	.	.	.	.	54	33	4	.	.	.	3	32	.	.	<5719>	3	565	.	.	.	.	.	3	.	.	.
PRO	.	.	.	.	.	.	.	.	.	.	.	.	.	.	17	.	23	.	<2910>	.	.	.	.	.	.	.	.	.	.
TO	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	38	.	<916>	.	.	.	.	.	.	.	.	.
UH	.	.	.	.	.	.	.	.	.	.	.	.	.	.	8	.	.	.	.	.	<.>	3	.	.	.	.	.	1	.
V	.	.	.	.	.	.	20	2	5	3	.	.	2	221	378	.	.	.	.	.	.	<4544>	.	.	4	.	22	.	.
VB+PPO	.	.	.	.	.	.	.	.	.	.	.	.	.	.	1	.	.	.	.	.	.	.	<.>	.	.	.	.	.	.
VBZ	.	.	.	.	.	.	.	.	.	.	.	.	.	71	175	.	.	.	.	.	.	.	.	<318>	.	.	.	.	.
VD	.	.	.	.	.	.	2	.	.	.	.	.	.	2	74	.	.	.	.	.	.	4	.	.	<470>	.	149	.	.
VG	.	.	.	.	.	.	4	.	.	.	.	.	.	32	304	1	.	.	.	.	.	.	.	.	.	<565>	.	.	.
VN	.	.	.	.	.	.	.	.	.	.	.	.	.	5	332	.	.	.	.	.	.	6	.	.	190	.	<1055>	.	.
WH	.	.	.	.	.	.	.	.	93	2	.	.	.	.	12	.	.	.	.	.	.	.	.	.	.	.	.	<763>	.
``	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	<396>

(row = reference; col = test)

<BigramTagger: size=1830>

Simplified Brown POS Tagset (From version 2.0 of NLTK)

Table 5.1:

Simplified Part-of-Speech Tagset

Tag	Meaning	Examples
ADJ	adjective	*new, good, high, special, big, local*
ADV	adverb	*really, already, still, early, now*
CNJ	conjunction	*and, or, but, if, while, although*
DET	determiner	*the, a, some, most, every, no*
EX	existential	*there, there's*
FW	foreign word	*dolce, ersatz, esprit, quo, maitre*
MOD	modal verb	*will, can, would, may, must, should*
N	noun	*year, home, costs, time, education*
NP	proper noun	*Alison, Africa, April, Washington*
NUM	number	*twenty-four, fourth, 1991, 14:24*
PRO	pronoun	*he, their, her, its, my, I, us*
P	preposition	*on, of, at, with, by, into, under*
TO	the word *to*	*to*
UH	interjection	*ah, bang, ha, whee, hmpf, oops*
V	verb	*is, has, get, do, make, see, run*
VD	past tense	*said, took, told, made, asked*
VG	present participle	*making, going, playing, working*
VN	past participle	*given, taken, begun, sung*
WH	*wh* determiner	*who, which, when, what, where, how*

How To Improve a Tagger?

- Look at errors
  - Combine tags
  - Get more training data
  - Make better rules
  - Create better algorithms

How to Improve a Tagger

- From the Confusion Matrix analysis:
  - We saw a word being confused frequently (“to”)
  - What happens if we look at the context it occurs in?
- Try the code on the next slide:
  - It shows how to view the (word, tag) that FOLLOWS the tag produced
    - – Like the “often/have” example in the homework
  - Try it on the word “TO” and its tags
    - – Use whatever tagset your tagger is trained on
    - – Use your current best tagger
  - See which words and tags follow “to” frequently

Exercise: Inspect the Tags that follow “to”

Evaluating a Tagger by Looking at Tags that Follow Tags

The word **to** is frequently confused; it can be helpful to inspect the context it occurs in. This code shows how to view the frequency of the tag that *follows* the word.

```
def examine_tag_contexts(tagger, target_word, target_tag):
    test_sents = [tagger.tag(sent) for sent in brown.sents(categories='editorial')]
    tags = [b[1] for test_sent in test_sents
            for (a,b) in nltk.bigrams(test_sent)
            if a[0] == target_word and a[1] == target_tag]
    fd = nltk.FreqDist(tags)
    print "Tags that follow the target word and tag"
    fd.tabulate()
examine_tag_contexts(ngram_tagger, 'to', 'TO')
examine_tag_contexts(ngram_tagger, 'to', 'IN')
```

```
Tags that follow the target word and tag
  VB    NN    AT    BE    DO    JJ    NP   PP$   PPO   NNS    HV    CS    DT    CD JJ-TL NN-TL    ``   VBG   WDT    DT
S  ABN    RB     . NP-TL    AP   DTI NNS-TL    PN  PPSS   WRB    IN     ,   BEG   BEZ   HVG JJR-TL    MD   PPL   PPL
S  PPS    QL   VBN   WPO
 541   265   229   106    31    28    27    26    23    18    15    10     9     8     8     8     8     7     7     6
   5     5     4     4     3     3     3     3     3     3     2     1     1     1     1     1     1     1     1     1
   1     1     1
Tags that follow the target word and tag
  AT    VB    NN    CD    BE   PP$    NP    JJ   PPO    DO NN-TL     .    CS   DTI    HV   JJR   VBG   VBN
  30    27    19    11     6     6     5     4     3     2     2     1     1     1     1     1     1     1
```

What we covered about tagging

- What are parts of speech
- What is POS tagging
- Methods for automatic POS tagging
  - N-gram (language model) based
  - Rule-based
  - Transformation-based learning
- Starting to see:
  - Evaluation
  - Supervised machine learning