ESSLLI 2010: Resource-light Morpho-syntactic Analysis of Highly Inflected Languages

Classical Approaches to Tagging

Anna Feldman & Jirka Hana

The slides are posted on the web. The url is
`http://chss.montclair.edu/~feldmana/esslli10/`.

Classical tagging techniques

Overview:

- Definition of tagging
- Non-statistical approaches to tagging
- Statistical approaches to tagging:
  - Supervised (HMMs in particular)
  - Unsupervised (only the definition)
- TnT (Brants 2000)
- Evaluation

What is morphological tagging?

- Part-of-speech (POS) tagging is the task of labeling each word in a sentence with its appropriate POS information.
- Morphological tagging is a process of labeling words in a text with their appropriate (in context) detailed morphological information.

An example: English

English:

Linguistics/**NN**	common noun
is/**AUX**	auxiliary
that/**DT**	determiner
branch/**NN**	common noun
of/**IN**	preposition
science/**NN**	common noun
which/**WDT**	wh-determiner
contains/**VBZ**	verb 3sg.
all/**PDT**	predeterminer
empirical/ **JJ**	adjective
investigations/**NNS**	plural common noun
concerning/**VBG**	gerund
languages/**NNS**	plural common noun
./**.**	

An example: Russian

Russian:

Byl/VpMS----R-AA---	be.**Verb.Past.Masc.Sg.Act.Affirm**
xolodnyj/AAMS1----1A----	cold.**Adj.Long.Masc.Sg.Nom.Posit.Affirm**
,/Z:-------------	,
jasnyj/AAMS1---1A----	bright.**Adj.Long.Masc.Sg.Nom.Posit.Affirm**
aprel'skij/AAMS1----1A----	April.**Adj.Long.Masc.Sg.Nom.Posit.Affirm**
den'/NNMS1----A----	day.**Noun.Masc.Sg.Nom**
,/Z:-------------	,
i/J*-------------	and.**coord-conjunction**
chasy/NNXP1-----A----	clocks.**Noun.Masc.Pl.Nom**
probili/VpXP----R-AA---	strike.**Past.Pl.Act.Affirm**
trinadcat'/CrXX4----------	thirteen.**Numeral.Card.Acc**
./Z:-------------	.

It was a bright cold day in April and the clocks were striking thirteen. (from Orwell's *1984*)

The problem of ambiguity

- POS tagging sounds trivial: for each word in the utterance, just look up its POS in the dictionary and append it to the word,
  Can/MD I/PRP book/VB that/DT flight/NN ?/?
  Does/VBZ that/DT flight/NN serve/VB dinner/NN ?/?
- The problem is that many common words are ambiguous
  - *can* can be a modal auxiliary, a noun, or a verb
  - *book* and *serve* can be verbs or nouns,
  - *that* can be a determiner or a complementizer (*I thought that your flight was earlier* vs. *I missed that flight*)

Ambiguous word types in the Brown corpus

- Most English words are unambiguous, but many of the most common words are ambiguous
- Ambiguity in the Brown corpus
  - 40% of word tokens are ambiguous
  - 12% of word types are ambiguous
  - Breakdown of ambiguous word types:

Unambiguous (1 tag)	35,340
Ambiguous (2–7 tags)	4,100
2 tags	3,760
3 tags	264
4 tags	61
5 tags	12
6 tags	2
7 tags	1 (“still”)

How bad is the ambiguity problem?

- Even though 40% of word tokens are ambiguous, one tag is usually much more likely than the others,
    - Example: in the Brown corpus, *race* is a noun 98% of the time, and a verb 2% of the time.
- A tagger **for English** that simply chooses the most likely tag for each word can achieve good performance.
- Any new approach should be compared against the unigram baseline (assigning each token to its most likely tag)

Ambiguity (cont.)

- Problem 1:
  - Mrs./NNP Shaefer/NNP never/RB got/VBD **around/RP** to/TO joining/VBG.
  - All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB **around/IN** the/DT corner/NN.
  - Chateau/NNP Petrus/NNP costs/VBZ **around/RB** 2500/CD.
- Problem 2:
  - cotton/NN sweater/NN; income-tax/JJ return/NN; the/DT Gramm-Rudman/NP Act/NP.
- Problem 3:
  - They were married/VBN by the Justice of the Peace yesterday at 5:00.
  - At the time, she was already married/JJ.

POS tagging (cont.)

- Input = a string of words and a specified tagset.
- Output = a single best tag for each word.
- We can say that POS tagging is a **disambiguation** task.

Tokenization

- Tokenization is required before tagging is performed
- Tokenization is the the task of converting a text from a single string to a list of tokens.
  e.g., *I read the book.* $\rightarrow$ [I, read, the, book, .]
- Tokenization is harder than it seems
  e.g.,
  - I'll see you in New York.
  - The aluminum-export ban.
- The simplest approach is to use "graphic words" (i.e., separate words using whitespace)

Two approaches to POS tagging

1. Rule-based tagging
   - Assign each word in the input a list of potential POS tags, then winnow down this list to a single tag using hand-written disambiguation rules
2. Statistical tagging (can be supervised/unsupervised)
   - Probabilistic: Find the most likely tag t for each word w, based on the prior probability of tag t:

     $\arg\max_t P(t|w) = \arg\max_t P(w|t)P(t)$
   - Transformation-based (Brill) tagging: Get a training corpus of tagged text, and give it to a machine learning algorithm so it will learn its own tagging rules (as in 1).

Supervised vs. Unsupervised tagging

- *Supervised* taggers
  - rely on pretagged corpora
- *Unsupervised* models
  - do not require a pretagged corpus,
  - use sophisticated computational methods to automatically induce word groupings (i.e., tagsets)
  - based on those automatic groupings calculate the probabilistic information needed by stochastic taggers or to induce the context rules needed by rule-based systems.

Pros and Cons

- Supervised taggers
  - tend to perform best when both trained and used on the same genre of text,
  - pretagged corpora are not readily available for the many language and genres which one might wish to tag.
- Unsupervised taggers
  - addresses the need to tag previously untagged genres and languages in light of the fact that hand tagging of training data is a costly and time-consuming process;
  - However, the word clusterings (i.e., automatically derived tagsets) which tend to result from these methods are very coarse, i.e., one loses the fine distinctions found in the carefully designed tag sets used in the supervised methods.

Rule-based POS tagging

The earliest algorithms for automatically assigning POS tags were based on a two-stage architecture:

1. Use a dictionary to assign each word a list of potential POS tags;
2. Use large lists of hand-written disambiguation rules to winnow down this list to a single POS for each word.

Rule-based POS tagging (cont.)

English Constraint Grammar approach (e.g., Karlsson et al. 1995) and EngCG tagger (Voutilainen, 1995,1999).
The rule-based tagger contains the following modules:

1. Tokenization
2. Morphological analysis
    - Lexical component
    - Rule-based guesser for unknown words
3. Resolution of morphological ambiguities

Rule-based approaches (cont.)

- There are thousands of rules, that are applied in steps (from basic to more advanced levels of analysis).
- Each rule either *adds*, *removes*, *selects* or *replaces* a tag or a set of grammatical tags in a given sentence context.
- Context conditions are included, both local (defined distances) or global (undefined distances)
- Context conditions in the same rule may be linked, i.e. conditioned upon each other, negated or blocked by interfering words or tags.

An Example

Pavlov had shown that salivation...

- Stage 1:
  - Pavlov **PAVLOV N NOM SG PROPER**
  - had **HAVE V PAST VFIN SVO** / HAVE PCP2 SVO
  - shown **SHOW PCP2 SVOO SVO SV**
  - that ADV / PRON DEM SG/ DET CENTRAL DEM SG / **CS**
  - salivation **N NOM SG**
- Stage 2: Apply constraints (3,744) (used in a negative way to eliminate tags that inconsistent with the context):

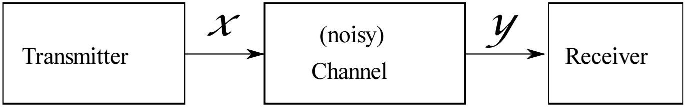
```
ADVERBIAL-THAT RULE
Given input: "that"
if
        (+1 A/ADV/QANT); if next word is adj, adverb, or quantifier
        (+2 SENT-LIM); and following which is a sentence boundary
        (NOT -1 SVOC/A); and the previous word is not a verb like
        "consider" which allows adjectives as object complements
then eliminate non-ADV tags
else eliminate ADV-tags
```

Q: How should "that" be analyzed in *I consider that odd.* based on the algorithm?

Noisy Channel

- Tags and words transferred over the noisy channel get corrupted into words
- We want to reconstruct the original message, but how?
- Possible solution: Markov model: we move between items of the original message (i.e. tags) and emit the items of the corrupted message (i.e. words).

http://upload.wikimedia.org/wikipedia/commons/4/48/Comm_Channel.svg



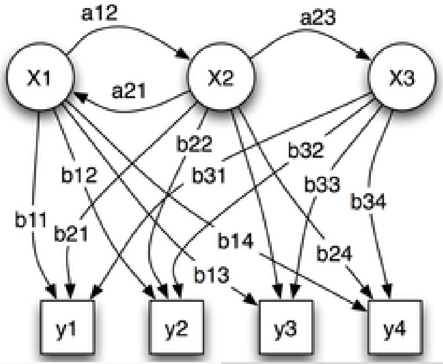
Hidden Markov Models (HMMs) and POS tagging

Problem definition: we are given some observation(s) and our job is to determine which of a set of classes it belongs to.
POS tagging is generally treated as a sequence classification task:

- observation = a sequence of words (e.g. sentence)
- task = assign the words a sequence of POS tags.
- I.e., find $\arg\max_t \; P(t|w)$

Probabilistic parameters of a hidden Markov model (example)

- x — states
- y — possible observations
- a — state transition probabilities
- b — output probabilities



n-grams

n-grams are sequences of probabilities based on a limited number of previous categories.

- The bigram model uses $P(t_i|t_i - 1)$ ("first order model")
- The trigram model uses $P(t_i|t_i - 2)$ ("second order model")

Exercise: How many *n*-grams does a corpus of *N* tokens have?

Example text: *a screaming comes across the sky (N = 6)*

Unigrams	Bigrams	Trigrams
a		
screaming	a screaming	
comes	screaming comes	a screaming comes
across	comes across	screaming comes across
the	across the	comes across the
sky	the sky	across the sky

A simple bigram tagger

- We want to find the most likely tag t for each word w
- We can use the Bayes' rule to calculate $\arg\max_t P(t|w)$: $P(B|A) = \frac{P(A|B)P(B)}{P(A)}$
- A bigram tagger makes the Markov assumption that $P(t)$ depends only on the previous tag t_{i-1}; $\arg\max_t P(t_i|w_i) = \arg\max_t P(w_i|t_i)P(t_i|t_{i-1})$
- The optimal $t_{1,n}$ then is calculated as $$\arg\max_{t_{1,n}} P(t_{1,n}|w_{1,n}) = \arg\max_{t_{1,n}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$
- We can train the tagger on a tagged corpus using Maximum Likelihood Estimate (MLE):

$$P(t|t_{i-1}) = \frac{C(t_{i-1},t)}{C(t_{i-1})}$$
$$P(w|t) = \frac{C(w,t)}{C(t)}$$

Transitions and emissions

- There are two sets of probabilities involved.
  - *Transition probabilities* control the movement from state to state (i.e., $p(t|p)$)
  - *Emission probabilities* control the emission of output symbols (=words) from the hidden states, i.e., $p(w_k|t_k)$

An Example: a toy corpus (taken from Allen (1995))

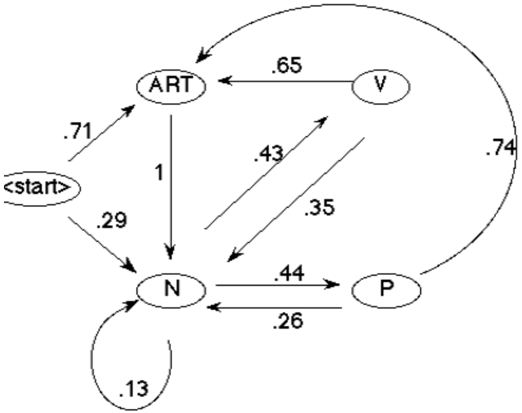
- The corpus consisted of 300 sentences and had words in only four categories: N, V, ART, and P, including 833 nouns, 300 verbs, 558 articles, and 307 prepositions for a total of 1998 words.

Here are some bigram probabilities estimated based on this corpus:

$$P(t|t_{i-1}) = \frac{C(t_{i-1},t)}{C(t_{i-1})}$$

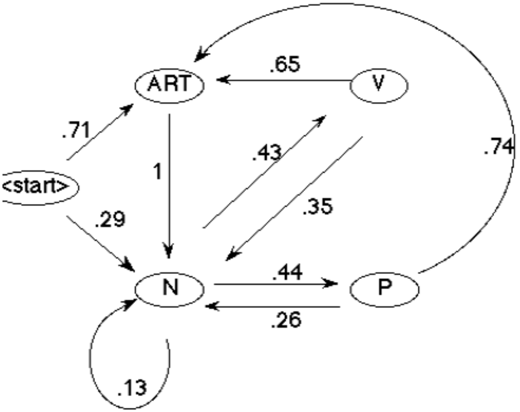
Category	Count at i	Pair	Count at i,$i+1$	Bigram	Estimate
<start>	300	<start>,ART	213	P(Art|<start>)	.71
<start>	300	<start>,N	87	P(N|<start>)	.29
ART	558	ART,N	558	P(N|ART)	1
N	833	N,V	358	P(V|N)	.43
N	833	N,N	108	P(N|N)	.13
N	833	N,P	366	P(P|N)	.44
V	300	V,N	75	P(N|V)	.35
V	300	V,ART	194	P(ART|V)	.65
P	307	P,ART	226	P(ART|P)	.74
P	307	P,N	81	P(N|P)	.26

A network capturing the bigram probabilities



How would you calculate the probability of *ART N V N*?

A network capturing the bigram probabilities



How would you calculate the probability of *ART N V N*?
$0.71 \times 1 \times 0.43 \times 0.35 = 0.107$.

Some corpus word counts

word	N	V	ART	P	TOTAL
flies	21	23	0	0	44
fruit	49	5	1	0	55
like	10	30	0	31	61
a	1	0	201	0	202
the	1	0	300	2	303
flower	53	15	0	0	68
flowers	42	16	0	0	58
birds	64	1	0	0	65
others	592	210	56	284	1142
TOTAL	833	300	558	307	1998

The emission probabilities $P(W_i|T_i)$ can be estimated simply by counting the number of occurrences of each word by category, i.e., $P(w|t) = \frac{C(w,t)}{C(t)}$
So, what is the P(flies|N)?

Some corpus word counts

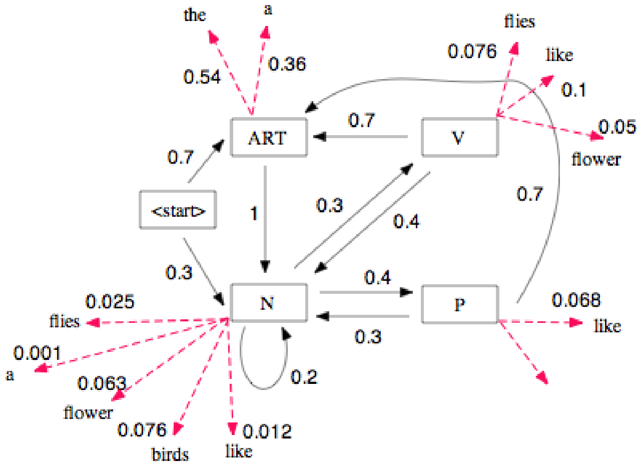
word	N	V	ART	P	TOTAL
flies	21	23	0	0	44
fruit	49	5	1	0	55
like	10	30	0	31	61
a	1	0	201	0	202
the	1	0	300	2	303
flower	53	15	0	0	68
flowers	42	16	0	0	58
birds	64	1	0	0	65
others	592	210	56	284	1142
TOTAL	833	300	558	307	1998

The emission probabilities $P(W_i|T_i)$ can be estimated simply by counting the number of occurrences of each word by category, i.e., $P(w|t) = \frac{C(w,t)}{C(t)}$
So, what is the P(flies|N)?
$21/833 = 0.025$

The emission probabilities

$P(the \mid ART)$	0.54	$P(a \mid ART)$	0.360
$P(flies \mid N)$	0.025	$P(a \mid N)$	0.001
$P(flies \mid V)$	0.076	$P(flower \mid N)$	0.063
$P(like \mid V)$	0.1	$P(flower \mid V)$	0.05
$P(like \mid P)$	0.068	$P(birds \mid N)$	0.076
$P(like \mid N)$	0.012		

Emissions and transitions together



Emissions and transitions (cont.)

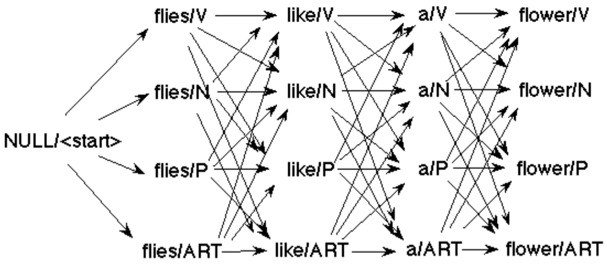
- The probability that the sequence *N V ART N* generates the output *Flies like a flower* is computed as follows:
  - The probability of the path *N V ART N* is $.29 \times .43 \times .65 \times 1 = 0.081$.
  - The probability of the output being *Flies like a flower* is computed from the output probabilities:

    $P(flies|N) \times P(like|V) \times P(a|ART) \times P(flower|N) =$
    $0.025 \times .1 \times .36 \times .063 = 5.4 \times 10^-5$

Back to Most-likely Tag Sequences

- So, how to find the most likely sequence of tags for a sequence of words?
- The key insight is that because of the Markov assumption, you do not have to enumerate all the possible sequences. Sequences that end in the same category can be collapsed together since the next category only depends on the previous one in the sequence.
- So if you just keep track of the most likely sequence found so far for each possible ending category, you can ignore all the other less likely sequences.

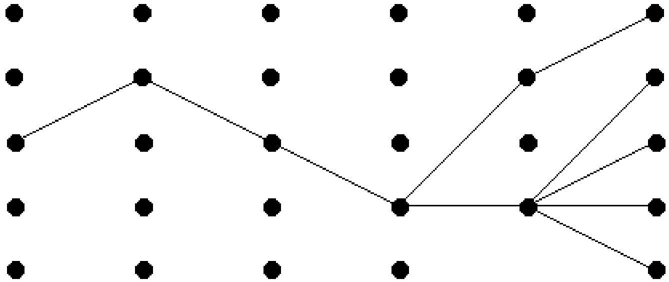
Encoding 256 possible sequences, using the Markov assumption



Finding the Most Likely Tag Sequence

- To find the most likely sequence, sweep forward through the words one at a time finding the most likely sequence for each ending category.
- In other words, you find the four best sequences for the two words *Flies like*: the best ending with *like* as a V, the best as an N, the best as a P and the best as an ART.
- You then use this information to find the four best sequences for the the words *flies like a*, each one ending in a different category.
- This process is repeated until all the words are accounted for.
- Very costly.
- A more efficient method is is the Viterbi algorithm (Viterbi 1967) .

Viterbi



Viterbi (cont.)

- The Viterbi algorithm finds the best possible path through the Markov model of states and transitions (the most likely sequence of hidden states), based on the transition and emission probabilities.
- The output is the sequence of observed events (words)
- The main observations:
  - For any state, there is only one most likely path to that state.
  - Therefore, if several paths converge at particular state, instead of recalculating them all, less likely path can be discarded.

Sparsity problem

- Standard *n*-gram models must be trained from some corpus
- any training corpus is finite
- some perfectly acceptable *n*-grams are bound to be missing from it
- Thus we have a very large number of cases of putative zero-probability *n*-grams that should really have some non-zero probability.
- Solution:
  - Smoothing (e.g., (Chen and Goodman 1996)): Assign a non-zero (small) probability to unseen possibilities

TnT tagger (Brants 2000)

- Trigrams'n'Tags (TnT) is a statistical Markov model tagging approach, developed by (Brants 2000).
- Performs very well
- States are tags; outputs are words; transition probabilities depend on the pairs of tags.
- Transitions and output probabilities are estimated from a tagged corpus, using maximum likelihood probabilities, derived from the relative frequencies.

TnT (cont.)

- Special features:
  - *Suffix analysis* for handling unknown words: Tag probabilities are set according to the word's ending because suffixes are word predictors for word classes (e.g., 98% of the words in the Penn Treebank corpus ending in *-able* are adjectives and the rest are nouns).
  - *Capitalization*: probability distributions of tags around capitalized words are different from those not capitalized
  - *Reducing the processing time*
    The processing time of the Viterbi algorithm is reduced by introducing a beam search. While the Viterbi algorithm is guaranteed to find the sequence of states with the highest probability, this is no longer true when beam search is added.

Evaluating POS taggers

- Taggers are evaluated by comparing them with a 'gold standard' (human-labeled) test set, based on percent correct: the percentage of all tags in the test set where the tagger and the gold standard agree
- Most current taggers get about 96% correct (for English)
- Note, however, that human experts don't always agree on the correct tag, which means the 'gold standard' is likely to have errors and 100% accuracy is impossible

Measures of success

The following measures are typically used for evaluating the performance of a tagger:

- Precision $= \frac{\text{Correctly-Tagged-Tokens}}{\text{Tags-generated}}$
  - Precision measures the percentage of system-provided tags that were correct.
- Recall $= \frac{\text{Correctly-Tagged-Tokens}}{\text{Tokens-in-data}}$
  - Recall measures the percentage of tags actually present in the input that were correctly identified by the system.
- F-measure $= 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$
  - The F-measure provides a way to combine these two measures into a single metric.

Exercise

- Imagine these 24 words in your Gold Standard corpus.
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. Chateau/NNP Petrus/NNP costs/VBZ around/RB 2500/CD ./.
- And this is the result that your tagger produced (**two words have multiple tags**):
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. **Chateau/NNP,NN Petrus/NNP,NC** costs/VBZ around/RB 2500/CD ./.
- What is the recall and precision of your tagger in this case?

Exercise

- Imagine these 24 words in your Gold Standard corpus.
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. Chateau/NNP Petrus/NNP costs/VBZ around/RB 2500/CD ./.
- And this is the result that your tagger produced (**two words have multiple tags**):
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. **Chateau/NNP,NN Petrus/NNP,NC** costs/VBZ around/RB 2500/CD ./.
- What is the recall and precision of your tagger in this case?

Precision: 24/26 = 0.92; Recall: 24/24 = 1; F-measure: 2*1*0.92/(0.92+1) = 1.84/1.92 = 0.96

Exercise (cont.)

- Imagine these 24 words in your Gold Standard corpus.
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. Chateau/NNP Petrus/NNP costs/VBZ around/RB 2500/CD ./.
- And this is the result that your tagger produced (**two words are left untagged**):
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. **Chateau/? Petrus/?** costs/VBZ around/RB 2500/CD ./.
- What is the recall and precision of your tagger in this case?

Exercise (cont.)

- Imagine these 24 words in your Gold Standard corpus.
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. Chateau/NNP Petrus/NNP costs/VBZ around/RB 2500/CD ./.
- And this is the result that your tagger produced (**two words are left untagged**):
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. **Chateau/? Petrus/?** costs/VBZ around/RB 2500/CD ./.
- What is the recall and precision of your tagger in this case?

Precision: 22/22 = 1; Recall: 22/24 = 0.92; F-measure: 2*1*0.92/(0.92+1) = 1.84/1.92 = 0.96

Exercise (cont.)

- Imagine these 24 words in your Gold Standard corpus.
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. Chateau/NNP Petrus/NNP costs/VBZ around/RB 2500/CD ./.
- And this is the result that your tagger produced (**4 words are mistagged; 2 words have no tags**):
  - **Mrs./IN Shaefer/NN** never/RB got/VBD **around/RB** to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ **go/VBD** around/IN the/DT corner/NN ./. **Chateau/? Petrus/?** costs/VBZ around/RB 2500/CD ./.
- What is the recall and precision of your tagger?

Exercise (cont.)

- Imagine these 24 words in your Gold Standard corpus.
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. Chateau/NNP Petrus/NNP costs/VBZ around/RB 2500/CD ./.
- And this is the result that your tagger produced (**4 words are mistagged; 2 words have no tags**):
  - **Mrs./IN Shaefer/NN** never/RB got/VBD **around/RB** to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ **go/VBD** around/IN the/DT corner/NN ./. **Chateau/? Petrus/?** costs/VBZ around/RB 2500/CD ./.
- What is the recall and precision of your tagger?

Precision: 18/22 = 0.82; Recall: 18/24 = 0.75; F-measure: 2*0.75*0.82/(0.75+0.82) = 1.23/1.57=0.78

Exercise (cont.)

- Imagine these 24 words in your Gold Standard corpus.
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. Chateau/NNP Petrus/NNP costs/VBZ around/RB 2500/CD ./.
- And this is the result that your tagger produced (**two words have multiple tags + four words are mistagged**):
  - **Mrs./IN Shaefer/NN** never/RB got/VBD **around/RB** to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ **go/VBD** around/IN the/DT corner/NN ./. **Chateau/NNP,NN Petrus/NNP,NC** costs/VBZ around/RB 2500/CD ./.
- What is the recall and precision of your tagger in this case?

Exercise (cont.)

- Imagine these 24 words in your Gold Standard corpus.
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. Chateau/NNP Petrus/NNP costs/VBZ around/RB 2500/CD ./.
- And this is the result that your tagger produced (**two words have multiple tags + four words are mistagged**):
  - **Mrs./IN Shaefer/NN** never/RB got/VBD **around/RB** to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ **go/VBD** around/IN the/DT corner/NN ./. **Chateau/NNP,NN Petrus/NNP,NC** costs/VBZ around/RB 2500/CD ./.
- What is the recall and precision of your tagger in this case?

Precision: 20/26 = 0.77; Recall: 20/24 = 0.83; F-measure: 2*0.83*0.77/ (0.83+0.77) = 1.2782/1.6 = 0.8

Exercise (cont.)

- Imagine these 24 words in your Gold Standard corpus.
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. Chateau/NNP Petrus/NNP costs/VBZ around/RB 2500/CD ./.
- And this is the result that your tagger produced (**1 word with two tags, 4 mistagged words, one word wasn't tagged**):
  - **Mrs./IN Shaefer/NN** never/RB got/VBD **around/RB** to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ **go/VBD** around/IN the/DT corner/NN ./. **Chateau/? Petrus/NNP,NC** costs/VBZ around/RB 2500/CD ./.
- What is the recall and precision of your tagger in this case?

Exercise (cont.)

- Imagine these 24 words in your Gold Standard corpus.
  - Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN ./. Chateau/NNP Petrus/NNP costs/VBZ around/RB 2500/CD ./.
- And this is the result that your tagger produced (**1 word with two tags, 4 mistagged words, one word wasn't tagged**):
  - **Mrs./IN Shaefer/NN** never/RB got/VBD **around/RB** to/TO joining/VBG ./. All/DT we/PRP gotta/VBN do/VB is/VBZ **go/VBD** around/IN the/DT corner/NN ./. **Chateau/? Petrus/NNP,NC** costs/VBZ around/RB 2500/CD ./.
- What is the recall and precision of your tagger in this case?

Precision: 19/24 =0.79; Recall: 19/24 = 0.79; F-measure: 2*0.79*0.79/(0.79+0.79) = 1.25/1.58 = 0.79