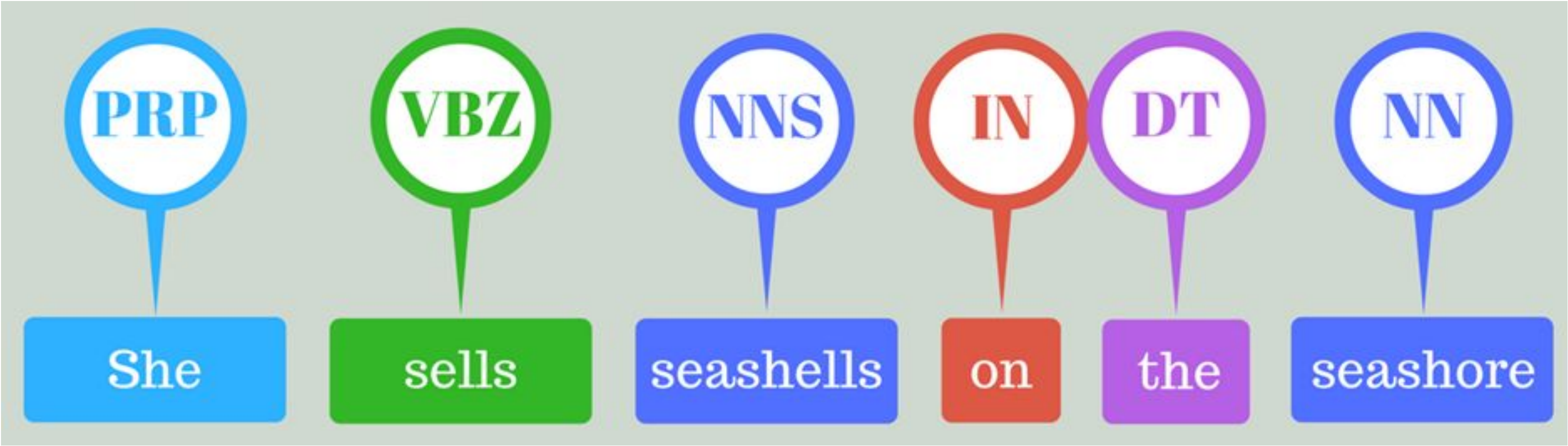
Part-of-Speech Tagging



Word Classes

- Words are traditionally grouped into equivalence classes:
  - Part-of-speech (POS)
  - Word classes
  - Lexical tags

Part-of-Speech (POS)

- The part of speech for a word gives a significant amount of information about the word and its neighbours
- Knowing the part of speech of a word tell us what words are likely to occur within its vicinity (i.e., context).
  - Example : in<PREP> the<DET> house<NOUN>
- Useful in a language model for speech recognition and word sense disambiguation

Part-of-Speech (POS)

- Can be divided into two broad categories:
  - **closed** class type
  - **open** class type

Closed Class

- Composed of a small, fixed set of grammatical function words (fixed membership). Examples:
  - **Prepositions (in, at, on, for, from, with, …)**
    - Fixed set of prepositions (new ones are rarely introduced)
  - **Function words (of, it, and, or, …)**
    - Grammatical words which tend to be very short, occur frequently, and play an important role in grammar

Open Class

- Have large number of words (expanding membership) and new ones are easily invented.
  - **Content words**
    - **Verbs (google, teach, study, …)**
    - **Nouns (googler, teacher, student, …)**
    - **Adjectives (large, small, easy, difficult, …)**
    - **Adverbs (surprisingly, happily, sadly, …)**
  - New nouns, verbs, adjectives and adverbs are continually introduced

English Tagsets

- Most commonly used in NLP today is the Penn Treebank set of 45 tags.
  - Texts from the Brown corpus
- The C5 tagset used for the British National Corpus (BNC) has 61 tags.
- Example of a tagged sentence from the Penn Treebank

  The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS

English Part-of-Speech

- **Noun (person, place or thing)**
  - Singular **(NN)**: cat, mouse, house
  - Plural **(NNS)**: cats, mice, houses
  - Proper **(NNP, NNPS)**: Yaseer, Malaysia
  - Personal pronoun **(PRP)**: I, you, he, she, it
  - Wh-pronoun **(WP)**: who, what
- **Verb (actions and processes)**
  - Base, infinitive **(VB)**: eat, drink, go
  - Past tense **(VBD)**: ate, drank, went
  - Gerund **(VBG)**: eating, drinking, going
  - Past participle **(VBN)**: eaten, drunk, gone
  - **Non 3rd person (e.g., I, you) singular** present tense **(VBP)**: eat
  - 3rd person (**e.g., he, she, it**) **singular** present tense: **(VBZ)**: eats
  - Modal **(MD)**, auxiliary verbs: shall, should, can, could, will, must
  - To **(TO)**: to (to eat)

English Part-of-Speech

- **Adjective (modify nouns)**
  - Basic **(JJ)**: red, tall
  - Comparative **(JJR)**: redder, taller
  - Superlative **(JJS)**: reddest, tallest
- **Adverb (modify verbs)**
  - Basic **(RB)**: quickly, slowly
  - Comparative **(RBR)**: quicker, slower
  - Superlative **(RBS)**: quickest, slowest
- **Preposition (IN)**: on, in, by, to, with
- **Determiner:**
  - Basic **(DT)** a, an, the
  - WH-determiner **(WDT)**: which, that
- **Coordinating Conjunction (CC)**: and, but, or,…
- **Particle (RP)**: off (took off), up (put up)

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	$
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	“	Left quote	(‘ or “)
POS	Possessive ending	*'s*	”	Right quote	(’ or ”)
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	Possessive pronoun	*your, one's*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Tagsets from Penn Treebank

Part-of-Speech (POS) Tagging

- The process of assigning(i.e., labelling) an appropriate part-of-speech or other lexical class marker to each word in a corpus
- Used in many **disambiguation** tasks
- Input to tagging algorithm:
  - **a string of words** and a specified **tagset**
- Output from algorithm:
  - **a single best tag** for each word

Example (POS) Tagging

VB DT NN .

Book that flight .

VBZ DT NN VB NN ?

Does that flight serve dinner ?

Ambiguity in (POS) Tagging

- "Book" is ambiguous, can be a **noun (NN)** or **verb (VB)**
  - The English book **(NN) – object**
  - I book**(VBP)** that flight **- reserve**
- "Like" can be a verb or a preposition
  - She likes**/VBZ** candy.
  - Time flies like**/IN** an arrow.
- POS-tagging resolve ambiguities by choosing the proper tag for the context.

Ambiguities in POS Tagging

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")

The number of word types in Brown corpus by degree of ambiguity

POS Tagging Approaches

- Rule-based tagging
  - Human crafted rules based on lexical and other linguistic knowledge
- Learning-based tagging
  - Trained on human annotated(i.e., labeled) corpora like the Penn Treebank
- **Learning-based** approaches have been found to be **more effective** overall considering the total amount of human expertise and effort involved

Classification Learning

- Use machine learning techniques to address the problem of classifying a **feature-vector description** into a fixed number of classes
- Some standard learning methods for this task:
  - Decision Trees and Rule Learning
  - Naïve Bayes and Bayesian Networks
  - Logistic Regression / Maximum Entropy (MaxEnt)
  - Perceptron and Neural Networks
  - Support Vector Machines (SVMs)
  - Nearest-Neighbor / Instance-Based

Sequence Labeling Problem

- Many NLP problems can viewed as sequence labeling.
- Each token in a sequence is assigned a label.
- Labels of tokens are dependent on the labels of other tokens in the sequence, particularly their neighbors

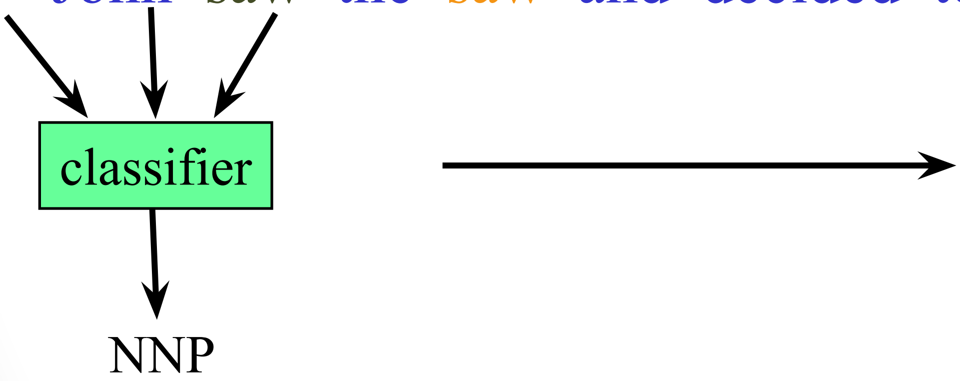
I want to fly a kite

(PP) (VB) (TO) (VB) (DT) (NN)

Sequence Labeling as Classification

- Classify each token independently but use as input features, information about the surrounding tokens (sliding window).

John saw the saw and decided to take it to the table.



Sequence Labeling as Classification Using Outputs as Inputs

- Better input features are usually the categories of the surrounding tokens, but these are not available yet.
- Can use category of either the preceding or succeeding tokens by going forward or back and using previous output

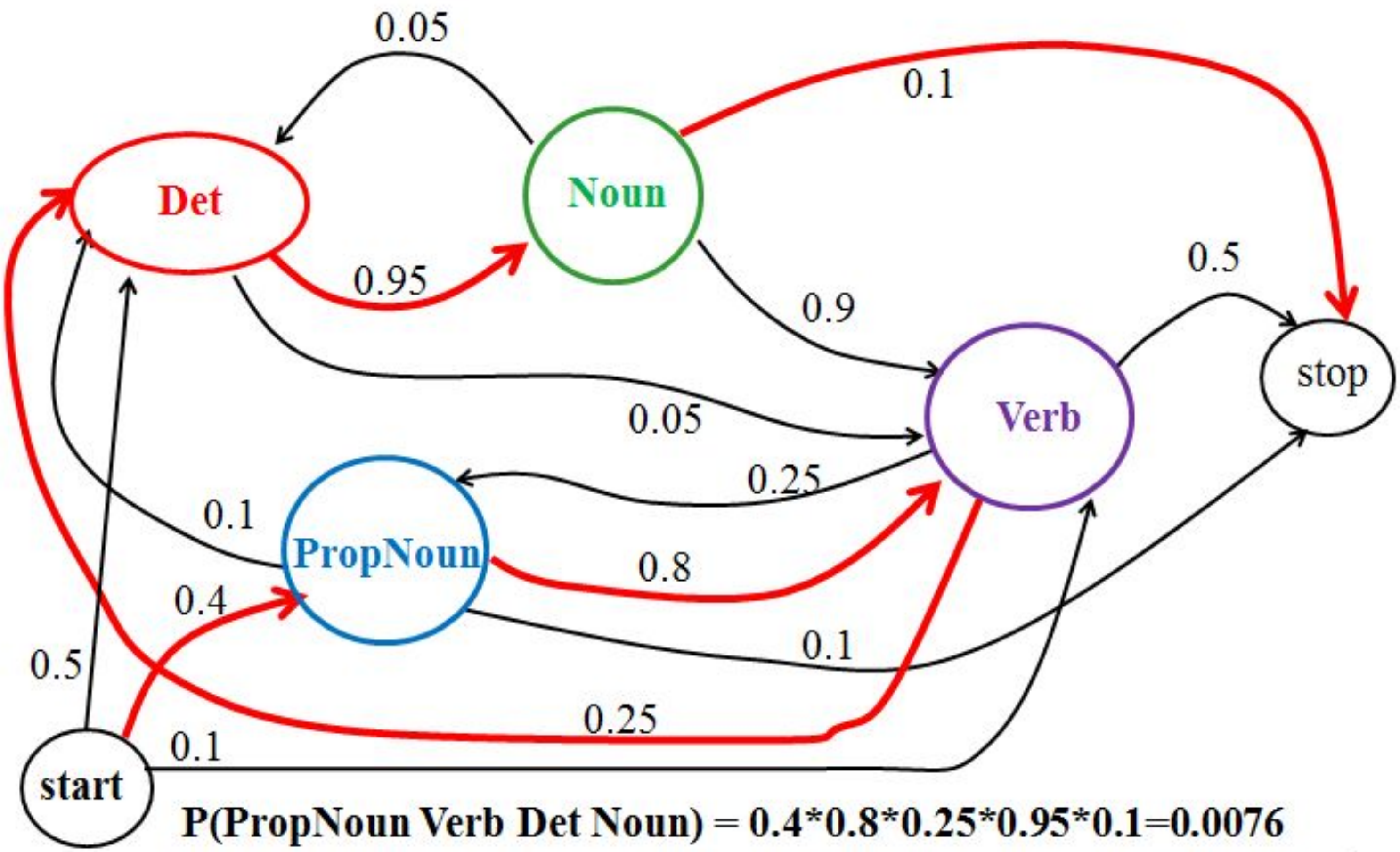
Probabilistic Sequence Model

- Probabilistic sequence models allow integrating uncertainty over multiple, interdependent classifications and collectively determine the most likely global assignment.
- Two standard models
  - Hidden Markov Model (HMM)
  - Conditional Random Field (CRF)

Markov Model/Markov Chain

- A finite state machine with probabilistic state transitions
- Makes Markov assumption that the next state only depends on the current state and independent of previous history

Sample Markov Model for POS



P(PropNoun Verb Det Noun) = 0.4*0.8*0.25*0.95*0.1=0.0076

HMM POS Tagging

- Also referred to as ***n*-gram tagging**
- Sequence classification task: **given a sequence** of ***N* words**, return a sequence of ***N* tags**
- Basic idea: consider all possible sequences of tags and choose the most probable given the sequence of words.
- Notation: let $\boldsymbol{w}_1^N$ be the sequence of N words, and t_1^N the sequence of N tags. Then our best hypothesis of the correct tag sequence, $\hat{t}_1^N$

$$\hat{t}_1^N = \operatorname*{arg\,max}_{t_1^n} P(t_1^N | w_1^N)$$

Bayes Rule

- Bayes rule is a useful way to manipulate conditional probabilities:

$$P(a|b) = \frac{P(b|a)P(a)}{\cancel{P(b)}}$$

- Rewrite the above as

$$\hat{t}_1^N = \arg\max_{t_1^n} \frac{P(w_1^N|t_1^N)P(t_1^N)}{\cancel{P(w_1^N)}}$$

Bayes Rule

- To find the most likely tag sequence involves comparing probabilities, given the same word sequence: hence P(w_1^N) does not change and we can write (denominator eliminated):

$$\hat{t}_1^N = \arg\max_{t_1^n} P(w_1^N | t_1^N) P(t_1^N)$$

likelihood

prior

Prior and Likelihood

$P(w_1^N|t_1^N)$ is called the likelihood
$P(t_1^N)$ is called the prior
We need some simplifying assumptions to estimate these terms
(1) Likelihood: assume that the probability of a word depends only on its tag; independent of surrounding words and their tags:

$$P(w_1^N|t_1^N) \sim \prod_{i=1}^{N} P(w_i|t_i)$$

(2) Prior: use an n-gram assumption (eg bigram):

$$P(t_1^N) \sim \prod_{i=1}^{N} P(t_i|t_{i-1})$$

Prior and Likelihood

- Putting it together:

- Bigram part-of-speech tagger computes the following to estimate the most likely tag sequence:

$$\hat{t}_1^N = \arg\max_{t_1^n} P(t_1^N | w_1^N)$$

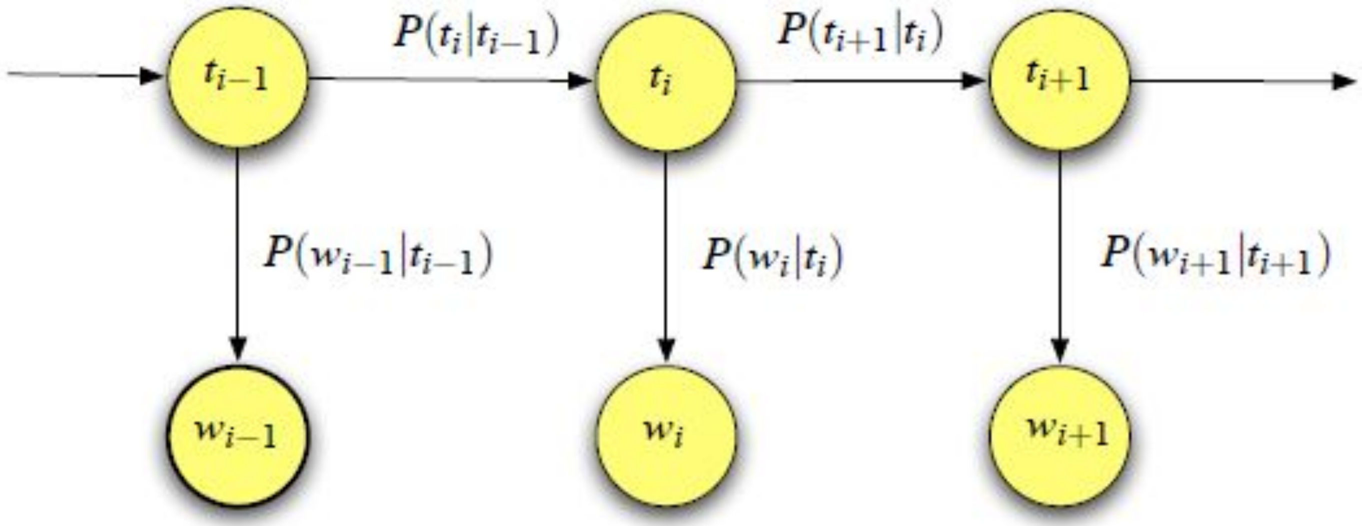
tag transition probability

$$\sim \prod_{i=1}^{N} P(w_i | t_i) P(t_i | t_{i-1})$$

word likelihood

- For each word take the product of the word likelihood (given the tag) and the tag transition probability

HMM Representation



Training and Decoding

- Training phase:
  - Estimate the probability tables
    - $P(w_i | t_i)$ and $P(t_i | t_{i-1})$
- Decoding phase:
  - Given the probability tables and a sequence of words, what is the most likely sequence of tags?

Training: Estimating Probabilities

Use maximum likelihood to estimate the tag transition and word probabilities by computing a ratio of counts:

$$P'(t_i|t_{i-1}) = \frac{c(t_{i-1}, t_i)}{c(t_{i-1})}$$

*Following bigram model

$$P'(w_i|t_i) = \frac{c(t_i, w_i)}{c(t_i)}$$

Example: estimate of $P(NN|DT)$ in the treebank:

*Estimating tag transition

$$P'(NN|DT) = \frac{c(DT, NN)}{c(DT)} = \frac{56\,509}{116\,454} = 0.49$$

Example: estimate of $P(\text{is}|VBZ)$:

*Estimating word prob.

$$P'(\text{is}|VBZ) = \frac{c(VBZ, \text{is})}{c(VBZ)} = \frac{10\,073}{21\,627} = 0.47$$

Unigram Tagging with NLTK

- Unigram taggers are based on a simple statistical algorithm: for each token, assign the tag that is most likely for that particular token
- Train a unigram tagger, use it to tag a sentence:

```
>>> from nltk.corpus import brown
>>> brown_tagged_sents = brown.tagged_sents(categories='news')
>>> brown_sents = brown.sents(categories='news')
>>> unigram_tagger = nltk.UnigramTagger(brown_tagged_sents)
>>> unigram_tagger.tag(brown_sents[2007])
[('Various', 'JJ'), ('of', 'IN'), ('the', 'AT'), ('apartments', 'NNS'),
('are', 'BER'), ('of', 'IN'), ('the', 'AT'), ('terrace', 'NN'), ('type', 'NN'),
(',', ','), ('being', 'BEG'), ('on', 'IN'), ('the', 'AT'), ('ground', 'NN'),
('floor', 'NN'), ('so', 'QL'), ('that', 'CS'), ('entrance', 'NN'), ('is', 'BEZ'),
('direct', 'JJ'), ('.', '.')]
>>> unigram_tagger.evaluate(brown_tagged_sents)
0.9349006503968017
```

Creating Training and Testing Data

- A tagger that simply memorized its training data and made no attempt to construct a general model would get a perfect score, which will be useless for tagging new text
- We need to split the data, training on 90% and testing on the remaining 10% (or 80%/20%, or 70%/30%)
- Observe that the performance score is slightly lower than the previous, when we separate train & test data

```
>>> 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)
0.811721...
```

Bigram Tagging with NLTK

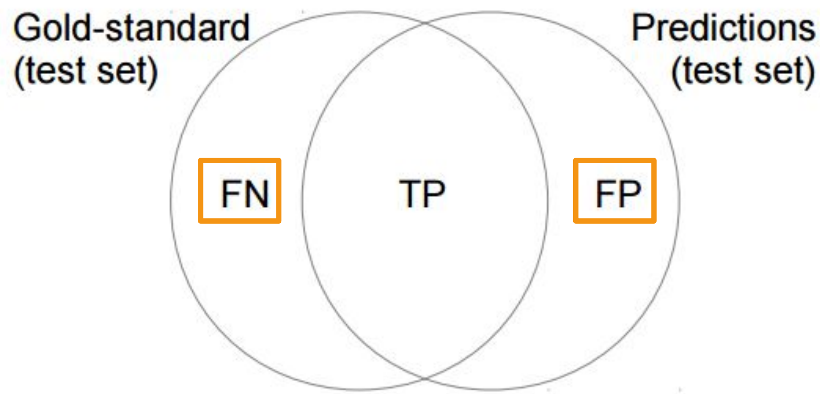
- Bigram tagger manages to tag every word in a sentence it saw during training, but does badly on an unseen sentence
- Overall accuracy score is very low :

```
>>> bigram_tagger = nltk.BigramTagger(train_sents)
>>> bigram_tagger.tag(brown_sents[2007])
[('Various', 'JJ'), ('of', 'IN'), ('the', 'AT'), ('apartments', 'NNS'),
('are', 'BER'), ('of', 'IN'), ('the', 'AT'), ('terrace', 'NN'),
('type', 'NN'), (',', ','), ('being', 'BEG'), ('on', 'IN'), ('the', 'AT'),
('ground', 'NN'), ('floor', 'NN'), ('so', 'CS'), ('that', 'CS'),
('entrance', 'NN'), ('is', 'BEZ'), ('direct', 'JJ'), ('.', '.')]
>>> unseen_sent = brown_sents[4203]
>>> bigram_tagger.tag(unseen_sent)
[('The', 'AT'), ('population', 'NN'), ('of', 'IN'), ('the', 'AT'), ('Congo', 'NP'),
('is', 'BEZ'), ('13.5', None), ('million', None), (',', None), ('divided', None),
('into', None), ('at', None), ('least', None), ('seven', None), ('major', None),
('``', None), ('culture', None), ('clusters', None), ("''", None), ('and', None),
('innumerable', None), ('tribes', None), ('speaking', None), ('400', None),
('separate', None), ('dialects', None), ('.', None)]
```

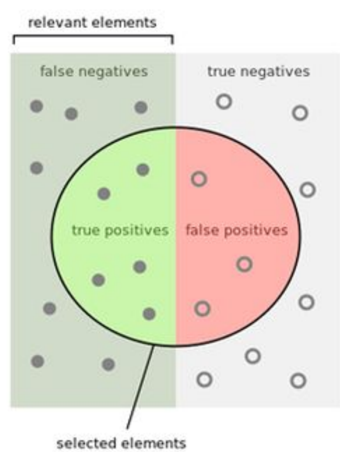
```
>>> bigram_tagger.evaluate(test_sents)
0.102063...
```

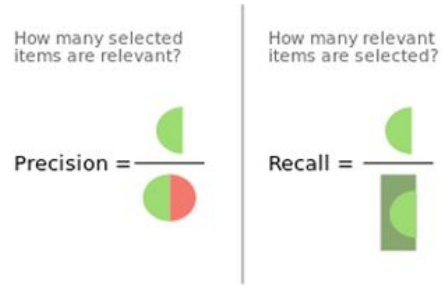
- As *n* gets larger, the specificity of the contexts increases, as does the chance that the data we wish to tag contains contexts that were not present in the training data, known as the *sparse data* problem

POS Tagging Evaluation



$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN} \qquad F - measure = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall}$$





Example

- Calculate the precision, recall and f-measure of the following POS task.
- Gold standard

Secretariat$_{[NNP]}$ is$_{[VBZ]}$ expected$_{[VBN]}$ to$_{[TO]}$ race$_{[VB]}$ tomorrow$_{[NR]}$.

- Prediction

Secretariat$_{[NNP]}$ is$_{[VBZ]}$ expected$_{[VBN]}$ to$_{[TO]}$ race$_{[NN]}$ tomorrow$_{[NR]}$.

5 TP, 1 FP [NN], 1 FN [VB]: Precision = 5/6

Exercise

- Calculate the precision, recall and f-measure of the following POS task.

- Gold standard

Time[NN] flies[VBZ] like[IN] AN[DT] ARROW[NN]

- Prediction

Time[NN] flies[NNP] like[VB][IN] AN[DT] ARROW[NN]