POS TAGGING

AMBIGUITY

- Many common English words can be more than one part of speech
- In the Brown corpus:
  - 11.5% of English word types are ambiguous
  - 40% of tokens are ambiguous

APPROACHES

- Rule-based
  - Many rules written by hand
- Stochastic
  - Model trained on a pre-tagged corpus

RULE-BASED EXAMPLE

- From EngCG system:

Given input: 'that'

if

(+1 A/ADV/QUANT) // if the next word is an adj, adv, or quantifier

(+2 SENT-LIM) //and the word after that is a sentence boundary

(NOT -1 SVOC/A) //and the previous word is not verb that allows complements

then eliminate non-ADV tags

else eliminate ADV tag

STOCHASTIC POS TAGGING

- Hidden Markov Model Part of Speech Tagging
  - Uses statistics to classify parts of speech
  - Considers all possible sequences of tags and chooses the most probable

IMPORTANT DEFINITION

- Argmax of a function

  $argmax_x f(x)$

  - means the x such that f(x) is maximized
- Example:
  - The estimated tag sequence (t-hat from 1-n of a sentence of length n) is the set of tags 1-n that return the highest probability of the tags 1-n given the words 1-n

$$\hat{t}_1^n = argmax_{t_1^n} P(t_1^n \mid w_1^n)$$

IMPORTANT DEFINITION

- Argmax of a function

  $$argmax_x f(x)$$

  - means the x such that f(x) is maximized
- Example:
  - The estimated tag sequence (t-hat from 1-n of a sentence of length n) is the set of tags 1-n that return the highest probability of the tags 1-n given the words 1-n

$$\hat{t}_i^n = argmax_{t_i^n} P(t_1^n | w_1^n)$$

BAYES' RULE

$$P(x \mid y) = \frac{P(y \mid x)P(x)}{P(y)}$$

$$P(t_1^n \mid w_1^n) = \frac{P(w_1^n \mid t_1^n)P(t_1^n)}{P(w_1^n)}$$

BAYES' RULE

- We only care about argmax!

$$P(t_1^n \mid w_1^n) = \frac{P(w_1^n \mid t_1^n)P(t_1^n)}{\cancel{P(w_1^n)}}$$

$$\hat{t}_1^n = argmax_{t_1^n} P(w_1^n \mid t_1^n)P(t_1^n)$$

HMM

- This is still too hard to compute!

$$P(w_1^n \mid t_1^n) \qquad\qquad P(t_1^n)$$

- Hidden Markov Models simplify the computation with 2 assumptions:
  - The probability of a word is independent of the words/tags around it
  - The probability of a tag is dependent only on the tag before it

HMM

- Hidden Markov Models simplify the computation with 2 assumptions:
  - The probability of a word is independent of the words/tags around it
  - The probability of a tag is dependent only on the tag before it

$$P(w_1^n \mid t_1^n) = \prod_{i=1}^{n} P(w_i \mid t_i) \quad P(t_1^n) = \prod_{i=1}^{n} P(t_i \mid t_{i-1})$$

$$\hat{t}_1^n = argmax_{t_1^n} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-1})$$

LET'S TRANSLATE INTO ENGLISH!

$$\hat{t}_1^n = argmax_{t_1^n} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-1})$$

- Tag-transition probability
  - The probability of a tag given the previous tag
  - P(JJ|DT) vs P(DT|JJ)

$$P(t_i \mid t_{i-1}) = \frac{C(t_{i-1}\ t_i)}{C(t_{i-1})}$$

LET'S TRANSLATE INTO ENGLISH!

$$\hat{t}_1^n = argmax_{t_1^n} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-1})$$

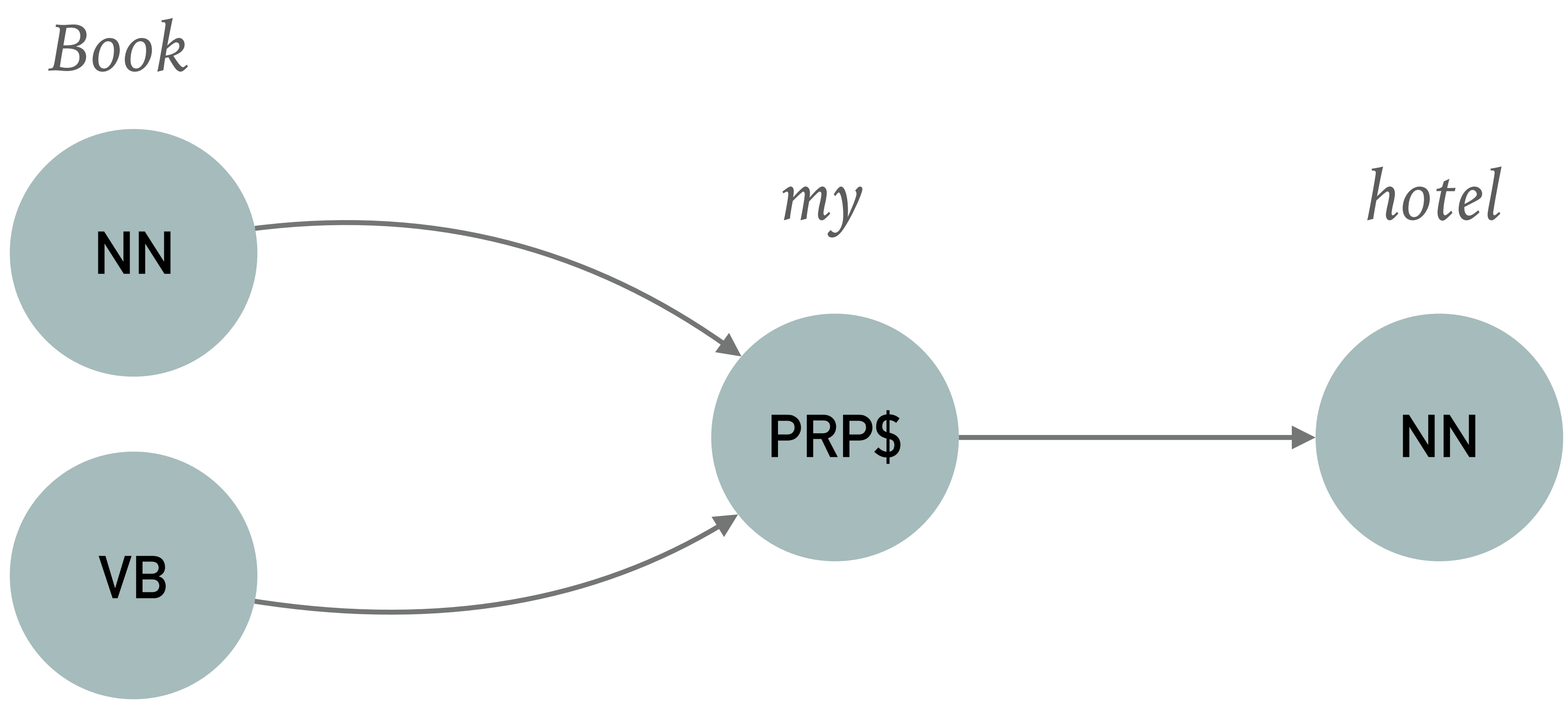
- Word likelihood probability
  - The probability of a word given the tag
  - !May seem backwards!
  - Instead of "What is the probability that 'book' is a verb?", "If we have a verb, how likely is it to be 'book'?"

$$P(w_i \mid t_i) = \frac{C(w_i \; has \; t_i)}{C(t_i)}$$

LET'S TRY!

$$\hat{t}_1^n = argmax_{t_1^n} \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-1})$$

- We want to find the best tag set for "book my hotel"



What probabilities do we need?

How do we get them?