

Part of Speech Tagging

(Slides adapted from Joakim Nivre and Staffan Larsson)

Part-of-Speech Tagging

- ▶ Given a word sequence $w_1 \cdots w_m$, determine the corresponding part-of-speech (tag) sequence $t_1 \cdots t_m$.
- ▶ Probabilistic view of the problem:

$$\arg \max_{t_1 \cdots t_m} P(t_1 \cdots t_m \mid w_1 \cdots w_m) \quad =$$

$$\arg \max_{t_1 \cdots t_m} \frac{P(t_1 \cdots t_m)P(w_1 \cdots w_m \mid t_1 \cdots t_m)}{P(w_1 \cdots w_m)} \quad =$$

$$\arg \max_{t_1 \cdots t_m} P(t_1 \cdots t_m)P(w_1 \cdots w_m \mid t_1 \cdots t_m)$$

Independence Assumptions

- ▶ Contextual model: Tags are dependent only on $n - 1$ preceding tags (tag n -grams, n -classes):

$$P(t_1 \cdots t_m) = \prod_{i=1}^m P(t_i \mid t_{i-(n-1)} \cdots t_{i-1})$$

For biclass models ($n = 2$),

$$P(t_1 \cdots t_m) = \prod_{i=1}^m P(t_i \mid t_{i-1})$$

- ▶ Lexical model: Word forms are dependent only on their own part-of-speech:

$$P(w_1 \cdots w_m \mid t_1 \cdots t_m) = \prod_{i=1}^m P(w_i \mid t_i)$$

For biclass models ($n = 2$),

$$\arg \max_{t_1 \dots t_m} P(t_1 \dots t_m) P(w_1 \dots w_m \mid t_1 \dots t_m) =$$

$$\arg \max_{t_1 \dots t_m} \prod_{i=1}^m P(t_i \mid t_{i-1}) \prod_{i=1}^m P(w_i \mid t_i) =$$

$$\arg \max_{t_1 \dots t_m} \prod_{i=1}^m P(t_i \mid t_{i-1}) P(w_i \mid t_i) =$$

Example

- ▶ Tagging *the can smells* using the triclass model:

$$P(dt \ nn \ vb \mid \textit{the can smells}) = P(dt \mid \# \#) \cdot P(nn \mid \# \ dt) \cdot P(vb \mid dt \ nn) \cdot P(\textit{the} \mid dt) \cdot P(\textit{can} \mid nn) \cdot P(\textit{smells} \mid vb)$$

- ▶ Compare:

$$\begin{aligned} P(\textit{can} \mid vb) &> P(\textit{can} \mid nn) \\ P(\textit{smells} \mid vb) &> P(\textit{smells} \mid nn) \\ P(nn \mid \# \ dt) &>> P(vb \mid \# \ dt) \\ P(vb \mid dt \ nn) &> P(nn \mid dt \ nn) (?) \end{aligned}$$

Hidden Markov Models 1

- ▶ Markov models are probabilistic finite automata that are used for many kinds of (sequential) disambiguation tasks such as:
 1. Speech recognition
 2. Spell checking
 3. Part-of-speech tagging
 4. Named entity recognition

- ▶ A (discrete) Markov model runs through a sequence of states emitting signals. If the state sequence cannot be determined from the sequence of emitted signals, the model is said to be hidden.

Hidden Markov Models 2

- ▶ A Markov model consists of five elements:
 1. A finite set of states $\Omega = \{s^1, \dots, s^k\}$.
 2. A finite signal alphabet $\Sigma = \{\sigma^1, \dots, \sigma^m\}$.
 3. Initial probabilities $P(s)$ (for every $s \in \Omega$) defining the probability of starting in state s .
 4. Transition probabilities $P(s^i | s^j)$ (for every $(s^i, s^j) \in \Omega^2$) defining the probability of going from state s^j to state s^i .
 5. Emission probabilities $P(\sigma | s)$ (for every $(\sigma, s) \in \Sigma \times \Omega$) defining the probability of emitting symbol σ in state s .

Hidden Markov Models 3

- ▶ State transitions are assumed to be independent of everything except the current state:

$$P(s_1 \cdots s_n) = P(s_1) \prod_{i=2}^n P(s_i | s_{i-1})$$

- ▶ Signal emissions are assumed to be independent of everything except the current state:

$$P(s_1 \cdots s_n, \sigma_1 \cdots \sigma_n) = P(s_1) P(\sigma_1 | s_1) \prod_{i=2}^n P(s_i | s_{i-1}) P(\sigma_i | s_i)$$

Hidden Markov Models 4

- ▶ If we want, we can simplify things by adding a dummy state s^0 such that $P(s) = P(s | s^0)$ (for every $s \in \Omega$)
- ▶ State transitions:

$$P(s_1 \cdots s_n) = \prod_{i=1}^n P(s_i | s_{i-1})$$

- ▶ Signal emissions:

$$P(s_1 \cdots s_n, \sigma_1 \cdots \sigma_n) = \prod_{i=1}^n P(s_i | s_{i-1}) P(\sigma_i | s_i)$$

Hidden Markov Models 5

- ▶ The probability of a signal sequence is obtained by summing over all state sequences that could have produced that signal sequence:

$$P(\sigma_1 \cdots \sigma_n) = \sum_{s_1 \cdots s_n \in \Omega^n} \prod_{i=1}^n P(s_i | s_{i-1}) P(\sigma_i | s_i)$$

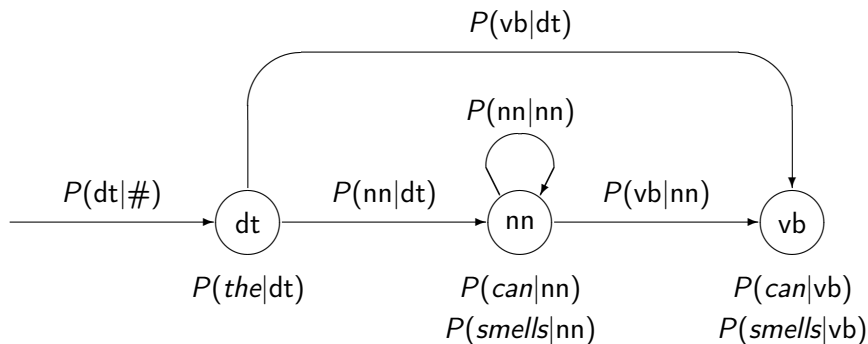
- ▶ Problems for HMMs:
 1. Optimal state sequence: $\arg \max_{s_1 \cdots s_n} P(s_1 \cdots s_n | \sigma_1 \cdots \sigma_n)$.
 2. Signal probability: $P(\sigma_1 \cdots \sigma_n)$.
 3. Parameter estimation: $\hat{P}(s)$, $\hat{P}(s_i | s_j)$, $\hat{P}(\sigma | s)$.

HMM Tagging

- ▶ Contextual model:
 1. The biclass model can be represented by a first-order Markov model, where each state represents one tag; $s_i = t_i$
 2. The triclass model can be represented by a second-order Markov model, where each state represents a pair of tags; $s_i = \langle t_i, t_{i-1} \rangle$
 3. In both cases, transition probabilities represent contextual probabilities.

- ▶ Lexical model:
 1. Signals represented word forms; $\sigma_i = w_i$
 2. Emission probabilities represent lexical probabilities; $P(\sigma_i | s_i) = P(w_i | t_i)$
 3. (t_0 written as # for biclass model)

Example: First-Order HMM



Parameter Estimation

- ▶ Two different methods for estimating probabilities (lexical and contextual):
 1. Supervised learning: Probabilities can be estimated using frequency counts in a (manually) tagged training corpus.
 2. Unsupervised learning: Probabilities can be estimated from an untagged training corpus using expectation-maximization.
- ▶ Experiments have shown that supervised learning yields superior results even with limited amounts of training data.

Supervised Learning

- ▶ Maximum likelihood estimation:

1. Contextual probabilities:

$$\hat{P}(t_i | t_{i-2}t_{i-1}) = \frac{C(t_{i-2}t_{i-1}t_i)}{C(t_{i-2}t_{i-1})}$$

2. Lexical probabilities:

$$\hat{P}(w | t) = \frac{C(w, t)}{C(t)}$$

- ▶ Maximum likelihood estimates need to be smoothed because of sparse data (cf. language modeling).

Supervised Learning Algorithm

```
for all tags  $t^j$  do
  for all tags  $t^k$  do
     $\hat{P}(t^k | t^j) := \frac{C(t^j t^k)}{C(t^j)}$ 
  end for
end for
for all tags  $t^j$  do
  for all words  $w^l$  do do
     $\hat{P}(w^l | t^j) := \frac{C(w^l : t^j)}{C(t^j)}$ 
  end for
end for
```

Supervised Learning

First tag	Second tag						Σ
	AT	BEZ	IN	NN	VB	PER	
AT	0	0	0	48636	0	19	48 655
BEZ	1973	0	426	187	0	38	2624
IN	43322	0	1325	17314	0	185	62146
NN	1067	3720	42470	11773	614	21392	81036
VB	6072	42	4758	1476	129	1522	
PER	8016	75	4656	1329	954	0	15030

$$\blacktriangleright \hat{P}(\text{AT} | \text{PER}) = \frac{C(\text{PER AT})}{C(\text{PER})} = \frac{8016}{15030} = 0.5333$$

Supervised Learning

	AT	BEZ	IN	NN	VB	PER
bear	0	0	0	10	43	0
is	0	10065	0	0	0	0
move	0	0	0	36	133	0
on	0	0	5484	0	0	0
president	0	0	0	382	0	0
progress	0	0	0	108	4	0
the	0	0	0	0	0	0
.	0	0	0	0	0	48809
total (all words)	120991	10065	130534	134171	20976	49267

$$\blacktriangleright \hat{P}(\text{bear} \mid \text{NN}) = \frac{C(\text{bear:NN})}{C(\text{NN})} = \frac{10}{134171} = 0.7453 \cdot 10^{-4}$$

Computing probabilities, example

Compute and compare the following two probabilities:

- ▶ $P(\text{AT NN BEZ IN AT NN} \mid \text{The bear is on the move.})$
- ▶ $P(\text{AT NN BEZ IN AT VB} \mid \text{The bear is on the move.})$

For this, we need $P(\text{AT} \mid \text{PER})$, $P(\text{NN} \mid \text{AT})$, $P(\text{BEZ} \mid \text{NN})$, $P(\text{IN} \mid \text{BEZ})$, $P(\text{AT} \mid \text{IN})$, and $P(\text{PER} \mid \text{NN})$, $P(\text{bear} \mid \text{NN})$, $P(\text{is} \mid \text{BEZ})$, $P(\text{on} \mid \text{IN})$, $P(\text{the} \mid \text{AT})$, $P(\text{move} \mid \text{NN})$, $P(\text{move} \mid \text{VB})$. We assume that the sentence is preceded by “.”.

Smoothing for Part-of-Speech Tagging

- ▶ Contextual probabilities are structurally similar to n-gram probabilities in language modeling and can be smoothed using the same methods.
- ▶ Smoothing of lexical probabilities breaks down into two sub-problems:
 1. Known words are usually handled with standard methods.
 2. Unknown words are often treated separately to take into account information about suffixes, capitalization, etc.

Viterbi Tagging

- ▶ HMM tagging amounts to finding the optimal path (state sequence) through the model for a given signal sequence.
- ▶ The number of possible paths grows exponentially with the length of the input.
- ▶ The Viterbi algorithm is a dynamic programming algorithm that finds the optimal state sequence in polynomial time.
- ▶ Running time is $O(ms^2)$, where m is the length of the input and s is the number of states in the model.

Viterbi algorithm

- 1: $\delta_0(\text{PERIOD}) = 1.0$
- 2: $\delta_0(t) = 0.0$ for $t \neq \text{PERIOD}$
- 3: **for** $i := 0$ to $n - 1$ **step 1 do**
- 4: **for** all tags t^j **do**
- 5: $\delta_{i+1} := \max_{1 \leq k \leq T} [\delta_i(t^k) \times P(t^j | t^k) \times P(w_{i+1} | t^j)]$
- 6: $\psi_{i+1} := \arg \max_{1 \leq k \leq T} [\delta_i(t^k) \times P(t^j | t^k) \times P(w_{i+1} | t^j)]$
- 7: **end for**
- 8: **end for**
- 9: $X_n = \arg \max_{1 \leq j \leq T} \delta_n(t^j)$
- 10: **for** $j := n - 1$ to 1 **step -1 do**
- 11: $X_j = \psi_{j+1}(X_{j+1})$
- 12: **end for**
- 13: $P(t^{X_1}, \dots, t^{X_n}) = \delta_n(t^{X_n})$

Tagset (part of full tagset) with indices

- ▶ t^1 =AT
- ▶ t^2 =BEZ
- ▶ t^3 =IN
- ▶ t^4 =NN
- ▶ t^5 =VB
- ▶ t^6 =PERIOD (PER)

Viterbi algorithm: first induction iteration

$i := 0$

for all tags t^j do do

$$\delta_1 := \max_{1 \leq k \leq T} [\delta_0(t^k) \times P(t^j | t^k) \times P(w_1 | t^j)]$$

$$\psi_1 := \arg \max_{1 \leq k \leq T} [\delta_0(t^k) \times P(t^j | t^k) \times P(w_1 | t^j)]$$

end for

We have $t^1 = \text{AT}$.

First tag:

$t^j := \text{AT}$

$$\delta_1 := \max_{1 \leq k \leq T} [\delta_0(t^k) \times P(\text{AT} | t^k) \times P(w_1 | \text{AT})]$$

$$\psi_1 := \arg \max_{1 \leq k \leq T} [\delta_0(t^k) \times P(\text{AT} | t^k) \times P(w_1 | \text{AT})]$$

Unsupervised Learning

- ▶ Expectation-Maximization (EM) is a method for approximating optimal probability estimates :
 1. Guess an estimate $\hat{\theta}$.
 2. Expectation: Compute expected frequencies based on training data and current value of $\hat{\theta}$.
 3. Maximization: Adjust $\hat{\theta}$ based on expected frequencies.
 4. Iterate steps 2 and 3 until convergence.
- ▶ Special case for HMM known as the Baum-Welch algorithm.
- ▶ Problem: Local maxima.

Example: Expectation-Maximization 1

- ▶ Lexicon:

<i>the</i>	dt
<i>car</i>	nn
<i>can</i>	nn vb

- ▶ Training corpus:

the can
the car

- ▶ Initial estimates:

$$\hat{P}(\text{nn}|\text{dt}) = \hat{P}(\text{vb}|\text{dt}) = 0.5$$

Example: Expectation-Maximization 2

Expectation

$$E[C(nn|dt)] = 1.5$$

$$E[C(vb|dt)] = 0.5$$

$$E[C(nn|dt)] = 1.75$$

$$E[C(vb|dt)] = 0.25$$

$$E[C(nn|dt)] = 1.875$$

$$E[C(vb|dt)] = 0.125$$

Maximization

$$\hat{P}(nn|dt) = 0.75$$

$$\hat{P}(vb|dt) = 0.25$$

$$\hat{P}(nn|dt) = 0.875$$

$$\hat{P}(vb|dt) = 0.125$$

$$\hat{P}(nn|dt) = 0.9375$$

$$\hat{P}(vb|dt) = 0.0625$$

Statistical Evaluation

- ▶ Many aspects of natural language processing systems can be evaluated by performing series of (more or less controlled) experiments.
- ▶ The results of such experiments are often quantitative measurements which can be summarized and analyzed using statistical methods.
- ▶ Evaluation methods are (in principle) independent of whether the systems evaluated are statistical or not.

Empirical Evaluation of Accuracy

- ▶ Most natural language processing systems make errors even when they function perfectly.
- ▶ Accuracy can be tested by running systems on representative samples of inputs.
- ▶ Three kinds of statistical methods are relevant:
 1. Descriptive statistics: Measures
 2. Estimation: Confidence intervals
 3. Hypothesis testing: Significant differences

Test Data

- ▶ Requirements on test data set:
 1. Distinct from any training data
 2. Unbiased (random sample)
 3. As large as possible
- ▶ These requirements are not always met.
- ▶ Testing may be supervised or unsupervised depending on whether the test data set contains solutions or not.
- ▶ Gold standard: Solutions provided by human experts.

Descriptive Statistics

- ▶ Descriptive measures such as sample means and proportions are used to summarize test results.
- ▶ Examples:

1. Accuracy rate (percent correct): $\frac{1}{n} \sum_{i=1}^n x_i$

2. Recall: $\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$

3. Precision: $\frac{\text{true positives}}{\text{true positives} + \text{false positives}}$

4. Logprob: $\frac{1}{n} \sum_{i=1}^n \log_2 \hat{P}(x_i)$

Example 1: Language Modeling

- ▶ Evaluation of language models as such are always unsupervised (no gold standards for string probabilities).
- ▶ Evaluation measures:
 1. Corpus probability
 2. Corpus entropy (logprob)
 3. Corpus perplexity

Example 2: PoS Tagging and WSD

- ▶ Supervised evaluation with gold standard is the norm.
- ▶ Evaluation measures:
 1. Percent correct
 2. Recall
 3. Precision

Example 3: Syntactic Parsing

- ▶ Supervised evaluation with gold standard (treebank).
- ▶ Evaluation measures:
 1. Percent correct
 2. Labeled/bracketed recall
 3. Labeled/bracketed precision
 4. Zero crossing