Maschinelle Sprachverarbeitung

Part-Of-Speech Tagging and Hidden Markov Models

Ulf Leser
Terminänderung

- Vorlesung vom 23.11.
- wird auf den 25.11., 11.00 Uhr, Raum 3.113
- verschoben
Content of this Lecture

• Part-Of-Speech (POS)
• Simple methods for POS tagging
• Hidden Markov Models
• Closing Remarks

• Most material from
  - [MS99], Chapter 9/10
Part-of-Speech (POS)

- In a sentence, each word has a **grammatical class**
- Simplest case: Noun, verb, adjective, adverb, article, …
  - That’s not a grammatical role: Subject, object, …
Tag Sets

• (POS-) tag set: Set of labels representing POS-classes
  – Simple tag set: Only broad word classes
  – Complex tag sets: Include morphological information
    • Noun: Gender, case, number
    • Verb: Tense, number, person
    • Adjective: normal, comparative, superlative
    • …

• Word classes even for the same language are not defined
  – London-Lund Corpus of Spoken English: 197 tags
  – Lancaster-Oslo/ Bergen: 135 tags
  – Penn tag set: 45 tags
  – Brown tag set: 87 tags
  – STTS (Stuttgart-Tübingen Tagset): ~50 tags
<table>
<thead>
<tr>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
<th>Tag</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>CC</td>
<td>Coordin. Conjunction</td>
<td><em>and, but, or</em></td>
<td>SYM</td>
<td>Symbol</td>
<td><em>+%, &amp;</em></td>
</tr>
<tr>
<td>CD</td>
<td>Cardinal number</td>
<td><em>one, two, three</em></td>
<td>TO</td>
<td>“to”</td>
<td><em>to</em></td>
</tr>
<tr>
<td>DT</td>
<td>Determiner</td>
<td><em>a, the</em></td>
<td>UH</td>
<td>Interjection</td>
<td><em>ah, oops</em></td>
</tr>
<tr>
<td>EX</td>
<td>Existential ‘there’</td>
<td><em>there</em></td>
<td>VB</td>
<td>Verb, base form</td>
<td><em>eat</em></td>
</tr>
<tr>
<td>FW</td>
<td>Foreign word</td>
<td><em>mea culpa</em></td>
<td>VBD</td>
<td>Verb, past tense</td>
<td><em>ate</em></td>
</tr>
<tr>
<td>IN</td>
<td>Preposition/sub-conj</td>
<td><em>of, in, by</em></td>
<td>VBG</td>
<td>Verb, gerund</td>
<td><em>eating</em></td>
</tr>
<tr>
<td>JJ</td>
<td>Adjective</td>
<td><em>yellow</em></td>
<td>VBN</td>
<td>Verb, past participle</td>
<td><em>eaten</em></td>
</tr>
<tr>
<td>JJR</td>
<td>Adj., comparative</td>
<td><em>bigger</em></td>
<td>VBP</td>
<td>Verb, non-3sg pres</td>
<td><em>eat</em></td>
</tr>
<tr>
<td>JJS</td>
<td>Adj., superlative</td>
<td><em>wildest</em></td>
<td>VBZ</td>
<td>Verb, 3sg pres</td>
<td><em>eats</em></td>
</tr>
<tr>
<td>LS</td>
<td>List item marker</td>
<td><em>1, 2, One</em></td>
<td>WDT</td>
<td>Wh-determiner</td>
<td><em>which, that</em></td>
</tr>
<tr>
<td>MD</td>
<td>Modal</td>
<td><em>can, should</em></td>
<td>WP</td>
<td>Wh-pronoun</td>
<td><em>what, who</em></td>
</tr>
<tr>
<td>NN</td>
<td>Noun, sing. or mass</td>
<td><em>llama</em></td>
<td>WPS</td>
<td>Possessive wh-</td>
<td><em>whose</em></td>
</tr>
<tr>
<td>NNS</td>
<td>Noun, plural</td>
<td><em>llamas</em></td>
<td>WRB</td>
<td>Wh-adverb</td>
<td><em>how, where</em></td>
</tr>
<tr>
<td>NNP</td>
<td>Proper noun, singular</td>
<td><em>IBM</em></td>
<td>$</td>
<td>Dollar sign</td>
<td>$</td>
</tr>
<tr>
<td>NNPS</td>
<td>Proper noun, plural</td>
<td><em>Carolinias</em></td>
<td>#</td>
<td>Pound sign</td>
<td>#</td>
</tr>
<tr>
<td>PDT</td>
<td>Predeterminer</td>
<td><em>all, both</em></td>
<td>“</td>
<td>Left quote</td>
<td>(‘ or “)</td>
</tr>
<tr>
<td>POS</td>
<td>Possessive ending</td>
<td>’s</td>
<td>”</td>
<td>Right quote</td>
<td>(‘ or ”)</td>
</tr>
<tr>
<td>PRP</td>
<td>Personal pronoun</td>
<td><em>I, you, he</em></td>
<td>(</td>
<td>Left parenthesis</td>
<td>([, (, {, &lt;)</td>
</tr>
<tr>
<td>PRP$</td>
<td>Possessive pronoun</td>
<td><em>your, one’s</em></td>
<td>)</td>
<td>Right parenthesis</td>
<td>(], ), }, &gt;)</td>
</tr>
<tr>
<td>RB</td>
<td>Adverb</td>
<td><em>quickly, never</em></td>
<td>,</td>
<td>Comma</td>
<td>,</td>
</tr>
<tr>
<td>RBR</td>
<td>Adverb, comparative</td>
<td><em>faster</em></td>
<td>.</td>
<td>Sentence-final punc</td>
<td>(. ! ?)</td>
</tr>
<tr>
<td>RBS</td>
<td>Adverb, superlative</td>
<td><em>fastest</em></td>
<td>:</td>
<td>Mid-sentence punc</td>
<td>(: ; ... – -)</td>
</tr>
<tr>
<td>RP</td>
<td>Particle</td>
<td><em>up, off</em></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Tagged Sentences

- **Simple tag set**
  - The/*D* koala/*N* put/*V* the/*D* keys/*N* on/*P* the/*D* table/*N*

- **Including morphological information**
  - The/*D* koala/*Ns* put/*V-past-3rd* the/*D* keys/*N-p* on/*P* …

- **Using Penn tag set**
  - The/*DT* koala/*NN* put/*VBN* the/*DT* keys/*NNS* on/*P* …

<table>
<thead>
<tr>
<th>The</th>
<th>koala</th>
<th>put</th>
<th>the</th>
<th>keys</th>
<th>on</th>
<th>the</th>
<th>table</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>N</td>
<td>V</td>
<td>D</td>
<td>N</td>
<td>P</td>
<td>D</td>
<td>N</td>
</tr>
<tr>
<td>D</td>
<td>N-sing</td>
<td>V-past-3rd</td>
<td>D</td>
<td>N-plu</td>
<td>P</td>
<td>D</td>
<td>N-sing</td>
</tr>
<tr>
<td>DT</td>
<td>NN</td>
<td>VBN</td>
<td>DT</td>
<td>NNS</td>
<td>P</td>
<td>DT</td>
<td>NN</td>
</tr>
</tbody>
</table>
POS Tagging

• Maybe each term has a single intrinsic grammatical class?
  – Peter, deliberately, school, the, better (?), …

• No: Homonyms
  – One term can represent many words (senses)
  – Different senses can have different word classes
  – “ist modern”–“Balken modern”, “Win a grant”–“to grant access”

• No: Words intentionally used in different word classes
  – “We flour the pan”, “Put the buy here”, “the buy back of …”
  – In German, things are easier: kaufen – Einkauf, gabeln – Gabelung
    • Of course, there are exceptions: wir essen – das Essen

• Still, most words have a preferred class
Problems

• Correct class depends on context within the sentence
  - The back door = JJ
  - On my back = NN
  - Win the voters back = RB
  - Promised to back the bill = VB

• Note: Also sentences may be ambiguous
  - The representative put chairs on the table
    • The/DT representative/NN put/VBD chairs/NNS on/IN the/DT table/NN
    • The/DT representative/JJ put/NN chairs/VBZ on/IN the/DT table/NN
  - Presumably the first is more probable than the second

• Another big problem (prob. the biggest): Unseen words
  - Recall Zipf’s law – there will always be unseen words
### A Real Issue

<table>
<thead>
<tr>
<th></th>
<th>Original 87-tag corpus</th>
<th>Treebank 45-tag corpus</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Unambiguous (1 tag)</strong></td>
<td>44,019</td>
<td>38,857</td>
</tr>
<tr>
<td><strong>Ambiguous (2–7 tags)</strong></td>
<td>5,490</td>
<td>8844</td>
</tr>
<tr>
<td>Details:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 tags</td>
<td>4,967</td>
<td>6,731</td>
</tr>
<tr>
<td>3 tags</td>
<td>411</td>
<td>1,621</td>
</tr>
<tr>
<td>4 tags</td>
<td>91</td>
<td>357</td>
</tr>
<tr>
<td>5 tags</td>
<td>17</td>
<td>90</td>
</tr>
<tr>
<td>6 tags</td>
<td>2 (well, beat)</td>
<td>32</td>
</tr>
<tr>
<td>7 tags</td>
<td>2 (still, down)</td>
<td>6 (well, set, round, open, fit, down)</td>
</tr>
<tr>
<td>8 tags</td>
<td></td>
<td>4 (’s, half, back, a)</td>
</tr>
<tr>
<td>9 tags</td>
<td></td>
<td>3 (that, more, in)</td>
</tr>
</tbody>
</table>

Source: Jurasky / Martin
Why POS Tagging?

- Parsing a sentence usually starts with POS tagging
- Finding phrases (shallow parsing) requires POS tagging
  - Noun phrases, verb phrases, adverbial phrases, …
- POS tags are beneficial for word sense disambiguation
- Applications in all areas of Text Mining
  - NER: ~10% boost using POS-features for single-token entities
  - NER: ~20% boost using POS-tags during post-processing of multi-token entities
- High accuracy with relative simple methods (97%)
- Many tagger available (BRILL, TNT, MedPost, …)
Content of this Lecture

• Part-Of-Speech (POS)
• **Simple methods for POS tagging**
  – Most frequent class
  – Syntagmatic rules
  – Transformation-based tagging
• Hidden Markov Models
• Closing Remarks
Simplest Method: Most Frequent Class

- Words have a preferred POS
  - The POS tag which a word most often gets assigned to
  - Recall school: We use words such as “adjektiviertes Verb”, “adjektiviertes Nomen”, “a noun being used as an adjective”
- Method: Tag each word with its preferred POS
Using Syntagmatic Information

- **Syntagmatic**: „the relationship between linguistic units in a construction or sequence“
  - [http://www.thefreedictionary.com]

- **Idea**: Look at surrounding POS tags
  - Some **POS-tag sequences are frequent**, others impossible
  - DT JJ NN versus DT JJ VBZ

- **Idea**: Count **frequencies of POS-patterns** in a tagged corpus
  - Count all tag bi-grams, tag tri-grams, …
  - Count regular expressions (DT * NN versus DT * VBZ)
  - … (many ways to define a pattern)
Usage for Tagging

- Start with words with **unique POS tags** (the, to, lady, …)
  - But: “The The”
- Find and apply the **most frequent patterns** over these tags
  - Assume `<DT JJ>` and `<JJ NN>` are frequent
  - “The blue car” -> DT ** * ** -> DT JJ ** * -> DT JJ NN
  - But: “The representative put chairs” -> DT ** ** * -> DT JJ ** ** -> DT JJ NN *
- Needs conflict resolution: “the bank in” -> DT ** 1 IN
  - Assume frequent bi-grams `<DT JJ>` and `<VBZ IN>`
- **Pattern-cover problem**: Cover a sentence with patterns such that the **sum of their relative frequencies** is maximal
Transformation-Based Tagger


- Idea: Identify „typical situations“ in a tagged corpus
  - Example: After “to”, there usually comes a verb
  - Situations may combine words, tags, morphological information, etc.

- Capture “situations” by transformation rules
- Apply when seeing untagged text
- Sort-of generalization of the syntagmatic approach
Transformation-Based Tagging

- **Learning rules**
  - We simulate the real case: “Untag” a tagged corpus
  - Tag each word with its **most probably** POS-tag
  - Find the **most frequent differences** between the original (tagged) text and the retagged text and encode as a rule
    - These are the most typical errors one performs when using only the most probable classes
    - Their correction (using the gold standard) is learned

- **Tagging**
  - Assign each word its most probable POS tag
  - Apply transformation rules to **rewrite** tag sequence
  - Issues: Order of application of rules? Termination?
Content of this Lecture

• Part-Of-Speech (POS)
• Simple methods for POS tagging
• Hidden Markov Models
  – Definition and Application
  – Learning the Model
  – Tagging
• Closing Remarks
Sequential Probabilistic Model

• Recall Markov Models (1st order)
  – A Markov Model is a stochastic process with states $s_1, \ldots, s_n$ with …
    • Every state emits exactly one symbol from $\Sigma$
    • No two states emit the same symbol
    • $p(w_n=s_n|w_{n-1}=s_{n-1},w_{n-2}=s_{n-2},\ldots,w_1=s_1) = p(w_t=s_t|w_{n-1}=s_{n-1})$

• That doesn’t help: Relationship POS – WORD is m:n

• We need an extension
  – We assume one state per POS tag
  – Each state may emit any word with a given probability
  – This is a Hidden Markov Model
  – When seeing a sentence, we can only observe the sequence of emissions, but not the underlying sequence of (hidden) states
Example

- Several possible paths, each with individual probability
  - DT – JJ – NN – VBZ
    - \( p(DT|"start") \times p(JJ|DT) \times p(NN|JJ) \times p(VBZ|NN) \times p(the|DT) \times p(representative|JJ) \times p(put|NN) \times p(chairs|VBZ) \)
  - DT – NN – VBZ – NN
    - \( p(DT|"start") \times p(NN|DT) \times p(VBZ|NN) \times p(NN|VBZ) \times p(the|DT) \times p(representative|NN) \times p(put|VBZ) \times p(chairs|NN) \)
Definition

• Definition

A Hidden Markov Model of order one is a sequential stochastic process with k states $s_1, \ldots, s_k$ with

- Every state $s$ emits every symbol $x \in \Sigma$ with probability $p(x|s)$
- The sequence of states is a 1st order Markov Model
- The $a_{0,1}$ are called start probabilities
- The $a_{t-1,t}$ are called transition probabilities
- The $e_s(x) = p(x|s)$ are called emission probabilities

• Note

- A given sequence of symbols can be emitted by many different sequences of states
- These have individual probabilities depending on the transition probabilities and the emission probabilities in the state sequence
Example

- **DT – JJ – NN – VBZ**
  \[
  p(DT|"start") \times p(JJ|DT) \times p(NN|JJ) \times p(VBZ|NN) \times \]
  \[
  p(\text{the}|DT) \times p(\text{representative}|JJ) \times p(\text{put}|NN) \times p(\text{chairs}|VBZ)
  \]
  \[
  = \ldots \times 0,4 \times 0,6 \times 0,6 \times \ldots
  \]

- **DT – NN – VBZ – NN**
  \[
  p(DT|"start") \times p(\text{NN}|DT) \times p(\text{VBZ}|\text{NN}) \times p(\text{NN}|\text{VBZ}) \times
  \]
  \[
  p(\text{the}|DT) \times p(\text{representative}|\text{NN}) \times p(\text{put}|\text{VBZ}) \times p(\text{chairs}|\text{NN})
  \]
  \[
  = \ldots \times 0,6 \times 0,6 \times 0,7 \times \ldots
  \]
HMM: Classical Problems

- **Decoding/parsing**: Given a sequence $S$ of symbols and a HMM $M$: Which *sequence of states* did most likely emit $S$?
  - This is our tagging problem once we have the model
  - Solution: Viterbi algorithm

- **Evaluation**: Given a sequence $S$ of symbols and a HMM $M$: With which *probability did M emit S*?
  - Fit of the model for the observation
  - Different than parsing, as many sequence may have emitted $S$
  - Solution: Forward/Backward algorithm (skipped here)

- **Learning**: Given a sequence $S$ and a set of states: Which HMM emits $S$ with the highest probability?
  - We need to learn start, emission, and transition probabilities
  - Solution: MLE or Baum-Welch algorithm (skipped here)
Another Example: The Dishonest Casino

A casino has two dice:
- **Fair die**
  \[ p(1) = p(2) = p(3) = p(4) = p(5) = p(6) = \frac{1}{6} \]
- **Loaded die**
  \[ p(1) = p(2) = p(3) = p(4) = p(5) = \frac{1}{10} \]
  \[ p(6) = \frac{1}{2} \]

Casino occasionally switches between dice
(and you want to know when)

**Game:**
1. You bet $1
2. You roll (always with a fair die)
3. You may **bet more or surrender**
4. Casino player rolls (with some die...)
5. Highest number wins

*Quelle: Batzoglou, Stanford*
The dishonest casino model

<table>
<thead>
<tr>
<th>FAIR</th>
<th>LOADED</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(1</td>
<td>F) = 1/6</td>
</tr>
<tr>
<td>P(2</td>
<td>F) = 1/6</td>
</tr>
<tr>
<td>P(3</td>
<td>F) = 1/6</td>
</tr>
<tr>
<td>P(4</td>
<td>F) = 1/6</td>
</tr>
<tr>
<td>P(5</td>
<td>F) = 1/6</td>
</tr>
<tr>
<td>P(6</td>
<td>F) = 1/6</td>
</tr>
</tbody>
</table>
Question # 1 – Decoding

**GIVEN** A sequence of rolls by the casino player
62146146136136661664661636616366163616515615115146123562344

**QUESTION** What portion of the sequence was generated with the fair die, and what portion with the loaded die?

This is the **DECODING** question
Question # 2 – Evaluation

**GIVEN** A sequence of rolls by the casino player
62146146136136661661636616366163616515615115146123562344

**QUESTION** How likely is this sequence, given our model of how the casino works?

This is the **EVALUATION** problem
Question # 3 – Learning

**GIVEN** A sequence of rolls by the casino player
6146136136661664661636616366163616515615115146123562344

**QUESTION**
How “loaded” is the loaded die? How “fair” is the fair die? How often does the casino player change from fair to loaded, and back?

This is the **LEARNING** question
[Note: We need to know how many dice there are!]
Content of this Lecture

- Part-Of-Speech (POS)
- Simple methods for POS tagging
- Hidden Markov Models
  - Definition and Application
  - Learning the Model
  - Tagging
- Concluding Remarks
Learning a HMM

• We always assume the set of states (POS tags) as fixed
• We need to learn start, emission and transition probabilities
• Assuming a large, tagged corpus, MLE does the job
  - Count relative frequencies of all starts, emissions, transitions
  - Start probabilities are straight-forward and skipped
MLE for Transition and Emission Probabilities

• Transitions
  - Count frequencies of all state transitions $s \rightarrow t$
  - Transform in relative frequencies for each outgoing state
    • Let $A_{st}$ be the number of transitions $s \rightarrow t$
    \[
    a_{st} = p(t \mid s) = \frac{A_{st}}{\sum_{t' \in M} A_{st'}}
    \]

• Emissions
  - Count frequencies of emissions over all symbols and states
  - Transform in relative frequencies for each state
    • Let $E_s(x)$ be the number of times that state $s$ emits symbol $x$
    \[
    e_s(x) = \frac{E_s(x)}{\sum_{x' \in \Sigma} E_s(x')}
    \]
Overfitting

• We have a **data sparsity problem**
  - Not so bad for the state transitions
    • Not too many POS Tags
    • But some classes are very rare
  - Quite bad for emission probabilities
    • As large as the corpus might be, most rare emissions are never seen and would (falsely) be assigned probability 0

• Need to apply **smoothing**
  - See previous lecture
Content of this Lecture

- Part-Of-Speech (POS)
- Simple methods for POS tagging
- Hidden Markov Models
  - Definition and Application
  - Learning the Model
  - Tagging
- Concluding Remarks
Viterbi Algorithm

• **Definition**

  *Let M be a HMM and S a sequence of symbols. The *parsing problem* is to find a (or all) state sequence of M that generated S with the *highest probability***

• Very often, we call a sequence of states a *path*

• Naïve solution
  + Let’s assume that $a_{ij}>0$ and $e_i(x)>0$ for all $x,i,j$ and $i,j \leq k$
  + Then there exist $k^n$ path
  + We cannot look at all of them

• **Viterbi-Algorithm**
Idea: Dynamic Programming

- Every potential state $s$ at position $i$ in $S$ is reachable by many paths.
- However, one of those must be the most probable one.
- All continuations of the path for $S$ from $s$ at position $i$ only need this highest probability over all paths reaching $s$ at $i$.
- Compute maximal probabilities iteratively for all positions.
Viterbi: Dynamic Programming

- We compute optimal (= most probable) paths for increasingly long prefixes of $S$
- Let $v_t(i)$ be the probability of the optimal path for $S[..i]$ ending in state $t$
- We want to express $v_t(i)$ using only the $v_{s \in M}(i-1)$ values
- Once we have found this formula, we may iteratively compute $v_s(1)$, $v_s(2)$, ..., $v_s(|S|)$ (for all $s \in M$)
Recursion

• Let $v_s(i)$ be the probability of the optimal path for $S[..i]$ ending in state $s$
• Assume we proceed from $s$ in position $i$ to $t$ in position $i+1$
• What is the probability of the path ending in $t$ passing through $s$ before?
  - The probability of $s$ ($=v_s(i)$)
  - * the transition probability from $s$ to $t$ ($a_{st}$)
  - * the probability that $t$ emits $S[i+1]$ ($=e_t(S[i+1])$)
• Of course, we may reach $t$ from any state at position $i$
• This gives

$$v_t(i + 1) = e_t(S[i + 1]) \cdot \max_{s \in M} (v_s(i) \cdot a_{st})$$
Tabular Computation

- Use **table for storing** $v_s(i)$
- Special start state with prob. 1; all other states have start prob. 0
- Compute **column-wise**
- Every cell can be reached from every cell in the previous column
- If a state never emits a certain symbol, all probabilities in columns with this symbol will be 0

<table>
<thead>
<tr>
<th></th>
<th>The</th>
<th>fat</th>
<th>blue</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S_0$</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>DT</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>JJ</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>NN</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>NNS</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>VB</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>VBZ</td>
<td>0</td>
<td>0</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The probability of the most probably parse is the largest value in the right-most column.

Most probable tag sequence is determined by traceback.
Complexity

- Let $|S| = n$, $|M| = k$ (states)
- This gives
  - The table has $n \times k$ cells
  - For computing a cell value, we need to access all potential predecessor states ($=k$)
  - Together: $O(n \times k^2)$
Numerical Difficulties

- Naturally, the numbers are getting extremely small
  - We are multiplying small probabilities (all $\ll 1$)
- We need to take care of not running into problems with computational accuracy
- Solution: Use logarithms
  - Instead of
    \[
    v_t(i+1) = e_t(S[i+1]) \times \max_{s \in M}(v_s(i) \times a_{st})
    \]
  - Compute
    \[
    v_t(i+1) = \log(e_t(S[i+1])) + \max_{s \in M}(v_s(i) + \log(a_{st}))
    \]
Unknown Words

• HMM do not help in tagging **unknown words**
• The treatment of unknown words is one of the **major differentiating features** in different POS taggers
• Simple approach: Are emitted by all **tags with equal prob.**
  – Their tags are estimated only by the transition probabilities
  – Not very accurate

• Information one may use
  – Morphological clues: suffixes (-ed mostly is past tense of a verb)
  – Likelihood of a **POS class of allowing a new word**
    • Some classes are closed: Determiner, pronouns, …
  – Special characters, “Greek” syllables, … (hint to proper names)
Wrap-Up

• Advantages of HMM
  - Clean framework
  - Relative simple math
  - Good performance when learning/predicting POS-tri-grams
    • Needs large learning corpus

• Disadvantages
  - Cannot capture non-local dependencies
    • Beyond the “n” of n-grams
  - Cannot condition probability of tags on concrete preceding words
    (but only on preceding tags)
    • But language has such constraints

• Extensions exist, but these are not trivial
  - Conditional Random Fields, Markov Logic Networks, …
Content of this Lecture

- Part-Of-Speech (POS)
- Simple methods for POS tagging
- Hidden Markov Models
- Concluding Remarks
Evaluation (English)

• Most frequent: ~85% accuracy
• Rule-based: <90% accuracy
• Probabilistic: >90% accuracy
• Domain-specific: ~97% accuracy
POS Tagging Today

- Brill, TnT, TreeTagger, OpenNLP MaxEnt tagger, …
- Choosing a tagger: Which corpus was used to learn the model? Domain specificity? Can I retrain it? Treatment of unknown words? Tag-Set?
- Some figures
  - Brill tagger has ~87% accuracy on Medline abstracts
    - When learned on Brown corpus = bad model for Medline
  - Performance of >97% accuracy is possible
    - MedPost: HMM-based, with a dictionary of fixed (word / POS-tag) assignments for the 10,000 most frequent “unknown” Medline terms
    - TnT / MaxEnt tagger reach 95-98 on newspaper corpora
- Further improvements hit inter-annotator agreement
  - And depend on the tag set – the richer, the more difficult
References