自然言語処理 (Natural Language Processing)

2017年度III期

- 担当：松本裕治(Yuji Matsumoto)
  柏岡秀樹(Hideki Kashioka)
  進藤裕之(Hiroyuki Shindo)
- 内容(Contents)：
  - Matsumoto：Natural Language Processing algorithms
    - Morphological Analysis (Part-of-speech tagging), Parsing
  - Shindo: Advanced topic of parsing algorithms
  - Kashioka：Machine Translation
    - Methodologies and algorithms for translationg natural language sentences into another language

Materials are available from the syllabus page of "自然言語処理 (Natural Language Processing)"

Schedule of this lecture

- Schedule：Thursdays (hour I)
  - 松本(Matsumoto)
    - October 5, 12, 19
  - 進藤(Shindo)
    - October 26, November 2
  - 柏岡(Kashioka)
    - November 9 (I and IV), 30

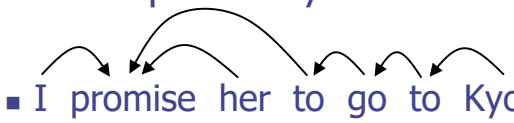
What does a sentence mean?

- Meaning of a sentence： Logical formula
  - predicate（verb）＋arguments（nouns）
  - Ex.： John loves Mary. → love(John, Mary)
  - John gives her a present → give(John, present, her)
  - John gives a present to her → the same as above
- It's not that simple!
  - I want to go to Kyoto. (I want and I go)
    - want(I, go(I, Kyoto))
  - I want her to go to Kyoto. (I want and she goes)
    - want(I, go(her, Kyoto))

- Different meaning by different verbs
  - I promise her to go to Kyoto.
    - promise(I, her, go(I, Kyoto))
  - I persuade her to go to Kyoto.
    - Persuade(I, her, go(her, Kyoto))

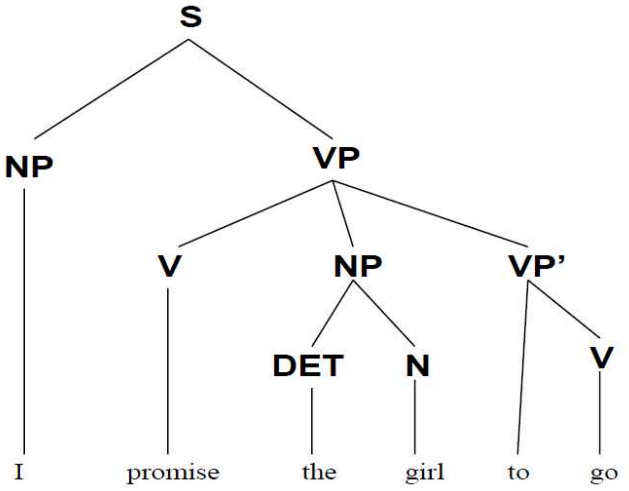
Syntactic Structures

- Word dependency trees



Describing modifier and modifyee relationship

Phrase Structure Tree(s)



What does a sentence describe?

- Truth-conditional view (真偽条件的視点)
  - A sentence describes a logical expression
  - Ex.: John loves Mary. → love(John, Mary)
- A sentence does not necessarily represent a complete logical expression
  - Ex.: He treated everyone.
    - Who is "He"? – anaphora resolution
    - Who is "everyone"? – interpretation of quantification
      - "everyone" does not mean all humans.
    - What does "treat" mean? – lexical sense ambiguity
    - When and where did he do it?

What a sentence encodes

(Explanation from Relevance Theory)

- 明意(explicature): Logical meaning of a sentence to be completed by the following refinements
  - Anaphora resolution (照応解析): Identification of antecedents of pronouns, demonstratives, definite expressions, etc. (代名詞, 指示表現などの先行詞)
  - Word sense disambiguation (語義曖昧性解消)
  - Completion of ellipsis (省略された語の補完)
  - Refinement of word interpretation (語の意味の詳細化)
    - enrichment (特定化): "everyone" should be interpreted in the context.
    - loosening (拡張): "I am 180cm tall" should be interpreted as "about 180cm tall"
  - Scope disambiguation of quantifiers (every, some, all, etc.) (限定詞の作用範囲の特定)

What a sentence implies

- Implicature (暗意)
  - Meaning implied by an utterance (its explicature) and a context
  - A: John said he would treat everyone.
    B: But, he only said it.
    - B's utterance implies that John didn't treat anyone.
  - To obtain an implicature from an utterance and the context, processes such as hypothesis construction (仮説形成) and deductive inference (演繹推論) are necessary
  - This process needs to take into account accessibility to the world knowledge and processing effort of the cognitive process

Natural Language Processing

- Practical analysis of natural language roughly stays within the analysis of the explicature of a sentence
- Basic natural language processing techniques
  - Morphological analysis / Part-of-speech tagging
  - Syntactic analysis
    - Phrase structure / dependency structure analysis
  - Word sense disambiguation
  - Anaphora / coreference analysis
  - Completion of ellipses
  - Predicate argument structure analysis

Contents of this lecture (Matsumoto)

- Morphological Analysis (形態素解析)
  - Tokenization / Segmentation (単語分割)
  - Part-of-speech tagging (品詞タグ付け)
- Syntactic Parsing Algorithms
  - Phrase structure parsing (句構造解析)
  - Word dependency parsing (依存構造解析)

POS-tagging, phrase chunking, parsing
品詞タグ付け、句チャンキング、統語解析

Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

Part-of-speech tagging

POS-tagged sentence

He reckons the current account deficit will narrow to only 1.8 billion in September .

PRP VBZ DT JJ NN NN MD VB TO RB CD CD IN NNP .

Base phrase chunking

Base phrase-chunked sentence

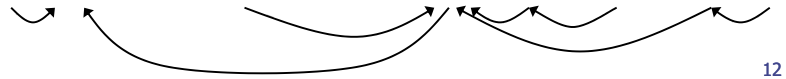
He reckons the current account deficit will narrow to only 1.8 billion in September .

NP VP NP VP PP NP PP NP

Dependency parsing

Dependency parsed sentence

He reckons the current account deficit will narrow to only 1.8 billion in September .



Word Dependency Parsing
単語依存構造解析

Raw sentence

He reckons the current account deficit will narrow to only 1.8 billion in September.

Part-of-speech tagging

POS-tagged sentence

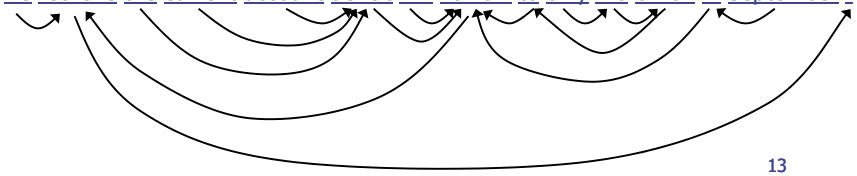
He reckons the current account deficit will narrow to only 1.8 billion in September.

PRP VBZ DT JJ NN NN MD VB TO RB CD CD IN NNP .

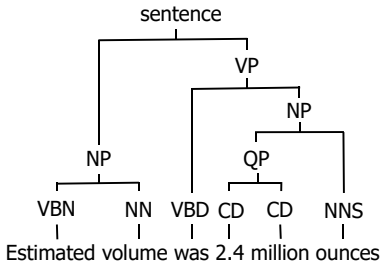
Word dependency parsing

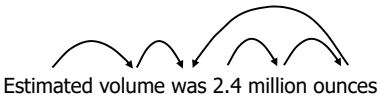
Word dependency parsed sentence

He reckons the current account deficit will narrow to only 1.8 billion in September .

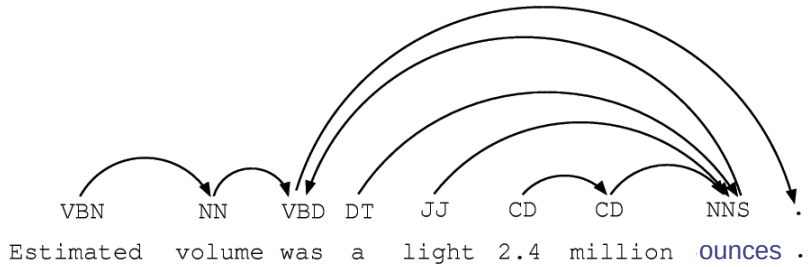


A phrase structure tree and a dependency tree(句構造木、依存構造木)





Dependency Tree
依存構造木の他の例



Contents of this lecture
講義内容

- Morphological Analysis
  - Tokenization (文の単語分割)
  - Part-of-speech tagging
- Syntactic Parsing Algorithms
  - Phrase structure parsing
  - Word dependency parsing

Tokenization

- Tokenization is a process of segmenting a sentence into its constituent words.
- Then, what are words?
  - Graphical words are those separated by spaces or delimiters (punctuation marks)
  - The narrow definition of tokenization is the process of segmenting a sentence into graphical words
  - Caution: delimiters are not necessarily separators (e.g., Mr., U.S.A., don't)

Narrow view of tokenization
狭い意味での単語分割

I'm going to meet Mr. Smith in New York tomorrow.

↓ tokenization

I 'm going to meet Mr. Smith in New York tomorrow .

- English POS tagging (when Penn Treebank is used) is usually defined on graphical words
- In the above example, "New" and "York" are tagged individually in Penn Treebank, as follows

I	'm	going	to	meet	Mr.	Smith	in	New	York	tomorrow	.
PRP	VBP	VBG	TO	VB	NNP	NNP	IN	NNP	NNP	NN	.

Definition of Major POS tags
主な品詞タグの定義

- PRP: personal pronoun (人称代名詞)
- VBP: verb, present tense (現在形動詞)
- VRG: verb, present participle (現在分詞)
- VB: verb, base form (原形動詞)
- VBD, VBN: past tense, past participle (過去形、過去分詞)
- TO: to
- NN: common noun (普通名詞), NNS: plural noun
- NNP: proper noun (固有名詞)
- IN: preposition (前置詞)
- JJ: adjective (形容詞)
- RB: adverb (副詞)
- etc

How to define words for tokenization
単語分割における単語の定義とは

- Lexeme(語彙項目): a set of all the forms that represent the same thing
- Lemma(代表形): the canonical form of a lexeme
  - An example of a lexeme: {take, takes, taking, took, taken, Take, Takes,...}
  - The lemma of this lexeme is "take"
- Word(単語): vague concept that corresponds to a form of a lexeme (In some cases, we use "word" and "lexeme" interchangeably)
- Lemmatization is a process to identify the lemma for each occurrence of a word
  - Note: Many Part-of-speech taggers do not return the lemma information

Tokenization and Part-of-speech Tagging of Japanese sentences

Japanese is a non-segmented language (no spaces between words)

私は新しく建った図書館へ行った(I went to the newly built library)

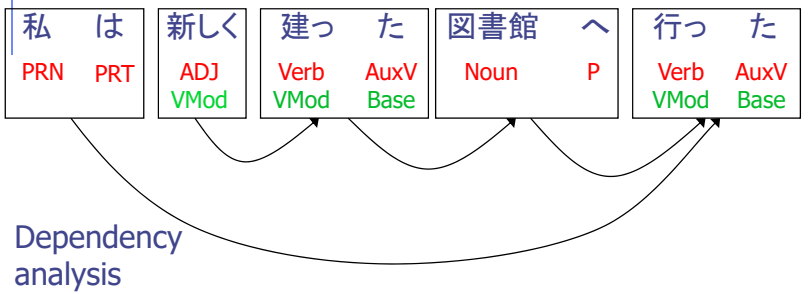
tokenization

私	は	新しく	建っ	た	図書館	へ	行っ	た
PRN	PRT	ADJ VMod	Verb VMod	AuxV Base	Noun	P	Verb VMod	AuxV Base
		新しい	建つ				行く	

word segmentation (tokenization)
POS tagging
Inflection form
Lemma

Base Phrase Chunking and Dependency Parsing of Japanese sentences

Base phrase chunking (文節分かち書き)



Basic Processes of POS tagging

1. Dictionary look-up
   - If tokenization is done in advance, dictionary look-up is a simple task
   - Non-segmented languages require to find all possible words appearing in an input sentence
2. Search of the most plausible tokenization and POS sequence
   - An algorithm is necessary to find the most plausible POS tag sequence from all possible tag sequences
   - A dynamic programming method (Viterbi algorithm) is usually used

Contents of this lecture

- Morphological Analysis
  - Tokenization
  - Part-of-speech tagging (品詞タグ付け)
- Syntactic Parsing Algorithms
  - Phrase structure parsing
  - Word dependency parsing

Statistical Part-of-speech tagging

- Given an input sentence, find the POS tag sequence that has the largest probability
- Many models assume Markov assumption
  - Limited horizon: the probability of the POS tag for the n-th word depends only on a previous few POS tags
  - Time (positional) invariance: conditional probability is not influenced by the position in sentence (only by the preceding POS tags)
- The overall sequence probability is calculated by the product of local probabilities

Definition of symbols

w_i : i-th word in the sentence

$w_{i,j}$: word sequence form w_i to w_j

$w_{1,n}$: the input sentence with length n

t_i : the POS tag of the i-th word, w_i

$t_{i,j}$: POS tag sequence of $w_{i,j}$

t_0 : a dummy POS tag at the beginning of sentence

t_{n+1} : a dummy POS tag at the end of sentence

$\arg\max_x M(x)$: the value x that gives the maximum value of M(x)

POS tagging by probability maximization

- Given a sentence ($w_{1,n}$=w_1,w_2,…,w_n), we need to get the most plausible POS sequence ($t_{1,n}$=t_1,t_2,…,t_n)
- In statistical methods, we get the POS sequence that gives the maximum probability

$$\arg\max_{t_{0,n+1}} P(t_{0,n+1} \mid w_{1,n})$$ ⟸ Probability of POS sequence $t_{0,n+1}$ given $w_{1,n}$

$$= \arg\max_{t_{0,n+1}} \frac{P(t_{0,n+1}, w_{1,n})}{P(w_{1,n})}$$ (definition of conditional prob.)

$$= \arg\max_{t_{0,n+1}} P(t_{0,n+1}, w_{1,n})$$

$$= \arg\max_{t_{0,n+1}} P(w_{1,n} \mid t_{0,n+1}) P(t_{0,n+1})$$

t_0 and t_{n+1} are dummy symbols representing the beginning and the end of the sentence

Simplification on the formulas

1. Word probabilities depend only on their POS tags

$$P(w_{1,n} \mid t_{0,n+1}) \approx P(w_1 \mid t_1) P(w_2 \mid t_2) \cdots P(w_n \mid t_n)$$

$$= \prod_{i=1}^{n} P(w_i \mid t_i)$$

2. POS probabilities depend only of a limited history

$$P(t_{0,n+1}) = P(t_0) P(t_1 \mid t_0) P(t_2 \mid t_0, t_1) \cdots P(t_{n+1} \mid t_{0,n})$$

$$\approx P(t_1 \mid t_0) \cdots P(t_i \mid h(t_{0,i-1})) \cdots P(t_{n+1} \mid h(t_{0,n}))$$

$$= P(t_1 \mid t_0) \prod_{i=1}^{n} P(t_{i+1} \mid h(t_{0,i}))$$

Note:
$P(t_0)=1$
$h(t_{0,i})$=limited POS tag history

Then, the original formula becomes as follows:

$$\arg\max_{t_{0,n+1}} P(t_{0,n+1} \mid w_{1,n}) = \arg\max_{t_{0,n+1}} P(w_{1,n} \mid t_{0,n+1}) P(t_{0,n+1})$$

$$= \arg\max_{t_{0,n+1}} P(t_1 \mid t_0) \prod_{i=1}^{n} P(w_i \mid t_i) P(t_{i+1} \mid h(t_{0,i}))$$

How to define the History

- Usually, the history is defined by a few preceding POS tags
  - bigram: history length = 1

$$h(t_{0,i}) = t_i$$

  - trigram: history length =2

$$h(t_{0,i}) = t_{i-1}, t_i$$

  - n-gram: history length = n-1

$$h(t_{0,i}) = t_{i-k+1}, \cdots, t_i = t_{i-k+1,i}$$ (k-gram)

From Probabilities to Costs

- POS tagging is a problem to find the POS sequence with the maximum probability
- Direct handling of probabilities may cause underflow in computation of long sentences because the total probability of such sentences tend to be very low
- Instead of using probability values, costs (logarithm of the inverse probabilities) are used in some cases
  - Maximization of probability is equivalent to minimization of its inverse
  - Taking its logarithm converts multiplication to addition

Taking log of unverse probabilities

- Convert the probability maximization problem into log minimization problem, we can use addition in stead of multiplication

$$\begin{aligned}
&\arg\max_{t_{0,n+1}} P(t_1 \mid t_0)\prod_{i=1}^{n} P(w_i \mid t_i)P(t_{i+1} \mid h(t_{0,i})) \\
&= \arg\min_{t_{0,n+1}} \frac{1}{P(t_1 \mid t_0)}\prod_{i=1}^{n}\frac{1}{P(w_i \mid t_i)}\frac{1}{P(t_{i+1} \mid h(t_{0,i}))} \\
&= \arg\min_{t_{0,n+1}} \log\left[\frac{1}{P(t_1 \mid t_0)}\prod_{i=1}^{n}\frac{1}{P(w_i \mid t_i)}\frac{1}{P(t_{i+1} \mid h(t_{0,i}))}\right] \\
&= \arg\min_{t_{0,n+1}} \left[\log\frac{1}{P(t_1 \mid t_0)} + \sum_{i=1}^{n}\left(\log\frac{1}{P(w_i \mid t_i)} + \log\frac{1}{P(t_{i+1} \mid h(t_{0,i}))}\right)\right] \\
&= \arg\min_{t_{0,n+1}} \left[(\text{cost of initial transition}) + \sum_{i=1}^{n}((\text{cost of word}) + (\text{cost of transition}))\right]
\end{aligned}$$

Summarization of probability (cost)-based POS tagging

- Tokenization of sentence into words
- A word may be associated with more than one POS tag
- A cost (probability) is defined to each word and each transition (all possible transitions form a trellis)
- The problem is to find a POS tag sequence that generates the input word sequence with the minimum cost (maximum probability)

$$\arg\min_{t_{0,n+1}}\left[\log\frac{1}{P(t_1 \mid t_0)} + \sum_{i=1}^{n}\left(\log\frac{1}{P(w_i \mid t_i)} + \log\frac{1}{P(t_{i+1} \mid h(t_{0,i}))}\right)\right]$$

- The search is done by a dynamic programming algorithm (Viterbi algorithm)

Viterbi search: an example

An example of Japanese analysis based on bi-gram model

POS_search.ppt

Viterbi algorithm: notation

- Algorithm to find the minimum cost path
- In the following, we show the bi-gram case

$T = \{t^1, t^2, \cdots t^p\}$: POS tag set

$c_{t^i}(k)$: total cost at position k with the POS tag t^i

$path_{t^i}(k)$: the minimum cost path at position k ending with the POS tag t^i

$c(t^i \rightarrow t^j)$: state transition cost from t^i to t^j

$c(t^i \rightarrow w_k)$: word emission cost of w_k from t^i

BOS, EOS : start and end tags

Viterbi algorithm: rough sketch

initialization : $c_{BOS}(0) = 0, \quad path(0) = \varepsilon$

for $k = 0, n$

  for $j = 1, p$

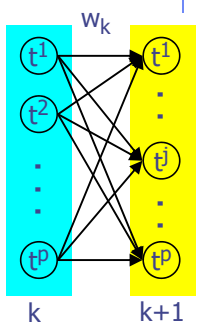
    $c_{t^j}(k+1) = \min_{1 \le i \le p} \left[c_{t^i}(k) + c(t^i \rightarrow t^j) + c(t^i \rightarrow w_k) \right]$

    $i' = \arg\min_{1 \le i \le p} \left[c_{t^i}(k) + c(t^i \rightarrow t^j) + c(t^i \rightarrow w_k) \right]$

    $path(k+1) = path(k) \cdot t^{i'}$

  end

end



Viterbi algorithm: initial step

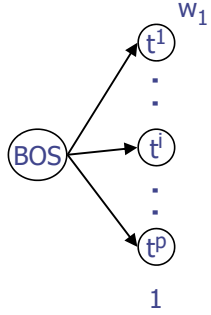
initialization : $c_{BOS}(0) = 0$

for $i = 1, p$

  $c_{t^i}(1) = c_{BOS}(0) + c(BOS \rightarrow t^i)$

  $path(1) = \varepsilon$

end

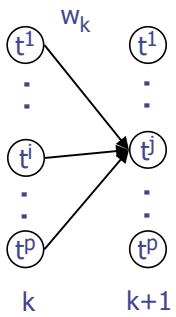


Viterbi algorithm: intermediate steps

$$c_{t^j}(k+1) = \min_{1 \le i \le p} \left[c_{t^i}(k) + c(t^i \rightarrow t^j) + c(t^i \rightarrow w_k) \right]$$

$$i' = \arg\min_{1 \le i \le p} \left[c_{t^i}(k) + c(t^i \rightarrow t^j) + c(t^i \rightarrow w_k) \right]$$

$$path(k) = path(k-1) \cdot t^{i'}$$

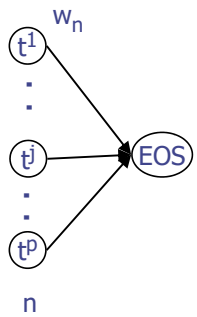


Viterbi algorithm: final step

$$c_{EOS}(n+1) = \min_{1 \le i \le p} \left[c_{t^i}(n) + c(t^i \rightarrow EOS) + c(t^i \rightarrow w_n) \right]$$

$$i' = \arg\min_{1 \le i \le p} \left[c_{t^i}(n) + c(t^i \rightarrow EOS) + c(t^i \rightarrow w_n) \right]$$

$$path(n+1) = path(n) \cdot t^{i'}$$



Estimation of the probabilities
確率値の推定法

- Unsupervised learning (no tagged data)
  - Baum-Welch algorithm (Hidden Markov Model)：隠れマルコフモデル
- Supervised learning (with tagged data)
  - Maximum likelihood（最尤推定）
  - Decision Trees（決定木）
  - Maximum Entropy model（最大エントロピー法）
  - Conditional Random Fields（CRF:条件付き確率場）
  - Recurrent Neural Networks（再帰型NN）+ CRF

Maximum Likelihood Estimation (MLE)
最尤推定

- Supervised learning of probabilities
  - When POS tagged corpus is available

$$P(t^i \rightarrow t^j) = P(t^j \mid t^i) = \frac{P(t^i, t^j)}{P(t^i)} = \frac{C(t^i, t^j)}{C(t^i)}$$

$$P(t^i \rightarrow w_k) = P(w_k \mid t^i) = \frac{P(w_k, t^i)}{P(t^i)} = \frac{C(w_k, t^i)}{C(t^i)}$$

where

$C(w_k, t^i)$: frequency count of word w_k appearing as t^i in tagged corpus

$C(t^i, t^j)$: frequency count of bigram occurrences of t^i and t^j

$C(t^i)$: frequencey count of (unigram occurrences) of t^i

Examples of MLE

Sample corpus

This/DT program/NN was/VBD written/VBN at/IN the/DT Department/NNP of/IN Computer/NNP and/CC Information/NNP Science/NNP University/NNP of/IN Pennsylvania/NNP ,/, and/CC the/DT Spoken/NNP Language/NN Systems/NNPS Group/NNP Laboratory/NNP for/IN Computer/NNP Science/NNP ,/, MIT/NNP ./.

state transition probability

$$P(\mathrm{DT} \rightarrow \mathrm{NN}) = P(\mathrm{NN} \mid \mathrm{DT}) = \frac{C(\mathrm{DT}, \mathrm{NN})}{C(\mathrm{DT})} = \frac{1}{3}$$

word emission probability

$$P(\mathrm{IN} \rightarrow \mathrm{of}) = P(\mathrm{of} \mid \mathrm{IN}) = \frac{C(\mathrm{of}, \mathrm{IN})}{C(\mathrm{IN})} = \frac{2}{4}$$

Tree Tagger

- N-gram models considers all combination of n-grams, which is usually very large --- causes the data sparseness problem
- Coping with the data sparseness problem
  - Smoothing
  - Reduction of the number of probability parameters
- Tree Tagger [Schmid 95] uses decision trees to estimate probabilities (with small number of features)

Probability Model of Tree Tagger

- Instead of the following probability model

$$P(w_{1,n}, t_{1,n}) = \prod_{i=1}^{n} P(w_i \mid t_i) P(t_i \mid t_{i-k}, \cdots t_{i-1})$$

- Tree Tagger uses the following formula

$$P(w_{1,n}, t_{1,n}) = \prod_{i=1}^{n} \frac{P(t_i \mid w_i)}{P(t_i)} P(t_i \mid t_{i-k}, \cdots t_{i-1})$$

The second formula is obtained by the Bayes rule and by omitting the word probability $P(w_i)$, because this affects all alternative tag sequence evenly.

$$P(w_i \mid t_i) = \frac{P(w_i, t_i)}{P(t_i)} = \frac{P(t_i \mid w_i) P(w_i)}{P(t_i)}$$

Decision Tree for estimating probabilities in Tree Tagger

- Decision tree is used to select features as well as to estimate probabilities, $p(t_i \mid t_{i-k}, \ldots, t_{i-1})$

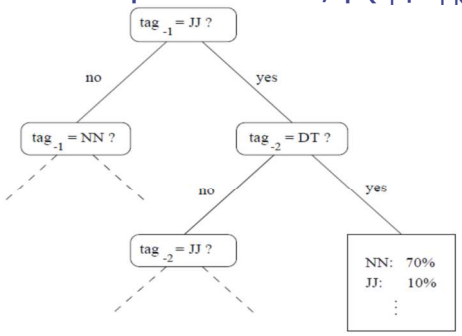


Figure 1: A sample decision tree (partially drawn).

*taken from: Helmut Schmid, "Improvements in Part-of-Speech Tagging with an Application to German," Proceedings of the ACL SIGDAT-Workshop. March 1995.