



Maschinelle Sprachverarbeitung

Part-Of-Speech Tagging and Hidden Markov Models

Ulf Leser

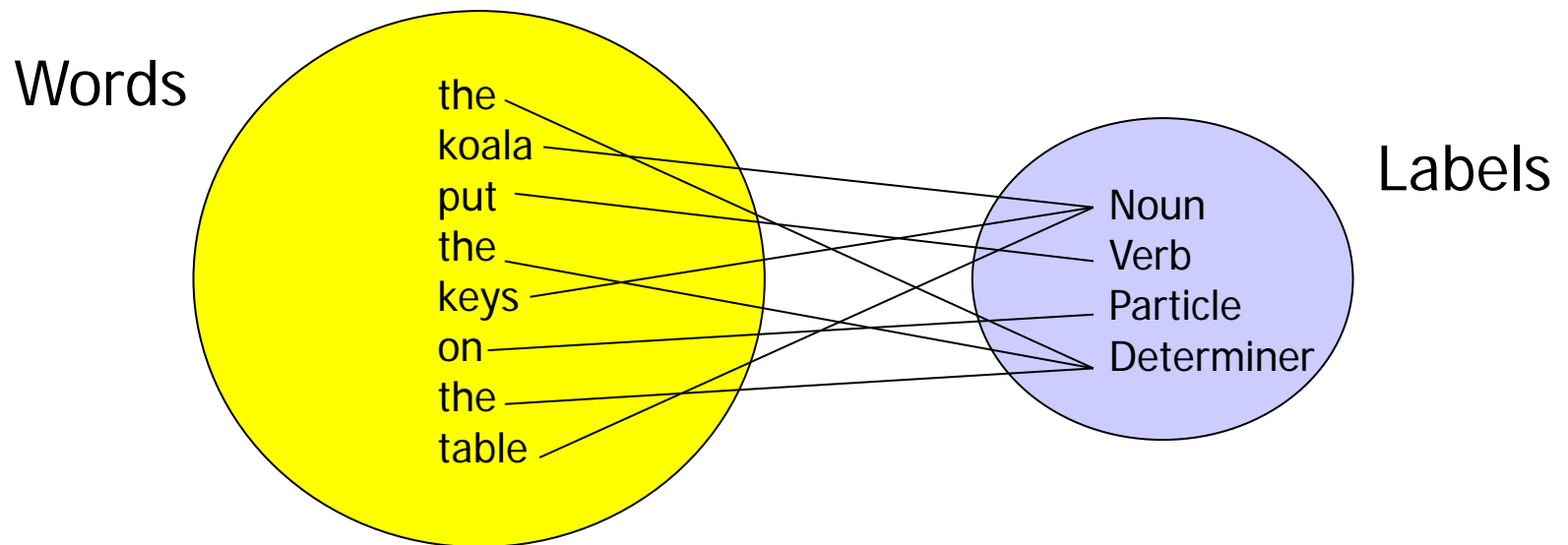
Content of this Lecture

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

- Most material from
 - [MS99], Chapter 9/10
 - Durbin, R., Eddy, S., Krogh, A. and Mitchison, G. (1998). "Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids". Cambridge University Press.
 - Rabiner, L. R. (1988). "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition." *Proceedings of the IEEE* **77**(2): 257-286.

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 within a language are not really 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

U-Penn TreeBank Tag Set (45 tags)

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

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** ...

The	koala	put	the	keys	on	the	table
D	N	V	D	N	P	D	N
D	N-sing	V-past-3rd	D	N-plu	P	D	N-sing
DT	NN	VBN	DT	NNS	P	DT	NN

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

	Original 87-tag corpus	Treebank 45-tag corpus
Unambiguous (1 tag)	44,019	38,857
Ambiguous (2–7 tags)	5,490	8844
Details:		
2 tags	4,967	6,731
3 tags	411	1621
4 tags	91	357
5 tags	17	90
6 tags	2 (<i>well, beat</i>)	32
7 tags	2 (<i>still, down</i>)	6 (<i>well, set, round, open, fit, down</i>)
8 tags		4 (<i>'s, half, back, a</i>)
9 tags		3 (<i>that, more, in</i>)

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
 - Assume frequent bi-grams <DT JJ> and <VBZ IN>
 - How to tag “the bank in” -> DT * IN
- **Pattern-cover problem**: Cover a sentence with patterns such that the **sum of their relative frequencies** is maximal

Transformation-Based Tagger

- Brill: „Transformation-Based Error-Driven Learning and Natural Language Processing: A Case Study in Part-of-Speech Tagging“, Computational Linguistics, 1995.
- 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?

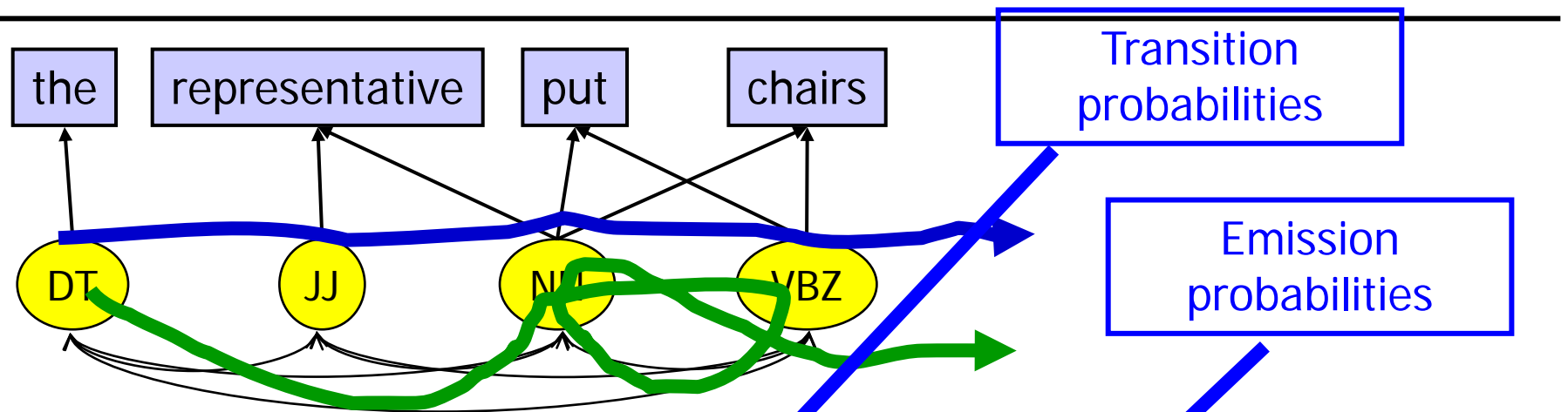
Content of this Lecture

- 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, \dots, s_n with ...
 - Every state emits exactly one symbol from Σ
 - No two states emit the same symbol
 - $p(w_n=s_n/w_{n-1}=s_{n-1}, w_{n-2}=s_{n-2}, \dots, w_1=s_1) = p(w_t=s_t/w_{n-1}=s_{n-1})$
- 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
 - $p(\text{DT}|\text{"start"}) * p(\text{JJ}|\text{DT}) * p(\text{NN}|\text{JJ}) * p(\text{VBZ}|\text{NN}) * p(\text{"the"}|\text{DT}) * p(\text{"representative"}|\text{JJ}) * p(\text{"put"}|\text{NN}) * p(\text{"chairs"}|\text{VBZ})$
 - DT - NN - VBZ - NN
 - $p(\text{DT}|\text{"start"}) * p(\text{NN}|\text{DT}) * p(\text{VBZ}|\text{NN}) * p(\text{NN}|\text{VBZ}) * p(\text{"the"}|\text{DT}) * p(\text{"representative"}|\text{NN}) * p(\text{"put"}|\text{VBZ}) * p(\text{"chairs"}|\text{NN})$

Definition

- Definition

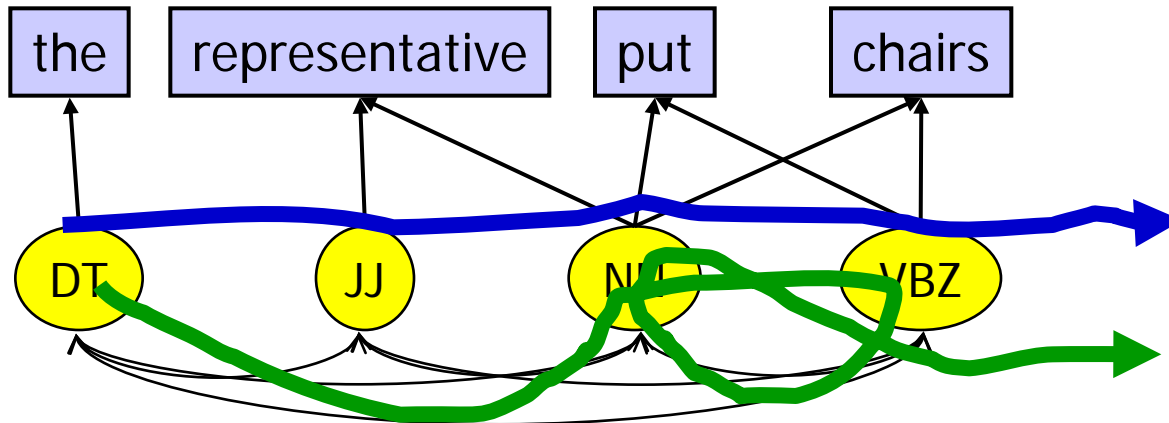
*A **Hidden Markov Model** of order one is a sequential stochastic process with k states s_1, \dots, s_k with*

- *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
DT	0	0,4	0,6	0
JJ	0	0,3	0,6	0,1
NN	0	0,2	0,2	0,6
VBZ	0,2	0	0,7	0,1

- DT – JJ – NN – VBZ

$$\begin{aligned} & p(\text{DT}|\text{start}) * p(\text{JJ}|\text{DT}) * p(\text{NN}|\text{JJ}) * p(\text{VBZ}|\text{NN}) * \\ & p(\text{the}|\text{DT}) * p(\text{representative}|\text{JJ}) * p(\text{put}|\text{NN}) * p(\text{chairs}|\text{VBZ}) \\ & = \dots * 0,4 * 0,6 * 0,6 * \dots \end{aligned}$$

- DT – NN – VBZ – NN

$$\begin{aligned} & p(\text{DT}|\text{start}) * p(\text{NN}|\text{DT}) * p(\text{VBZ}|\text{NN}) * p(\text{NN}|\text{VBZ}) * \\ & p(\text{the}|\text{DT}) * p(\text{representative}|\text{NN}) * p(\text{put}|\text{VBZ}) * p(\text{chairs}|\text{NN}) \\ & = \dots * 0,6 * 0,6 * 0,7 * \dots \end{aligned}$$

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 tagged sequence S : **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

$$p(1)=p(2)=p(3)=p(4)=p(5)=p(6)=1/6$$

- Loaded die

$$p(1)=p(2)=p(3)=p(4)=p(5)=1/10$$

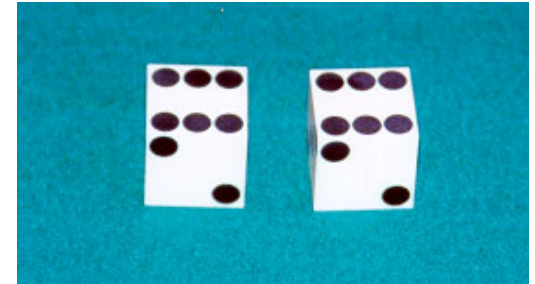
$$p(6) = 1/2$$

Casino occasionally switches between dice

(and you want to know when)

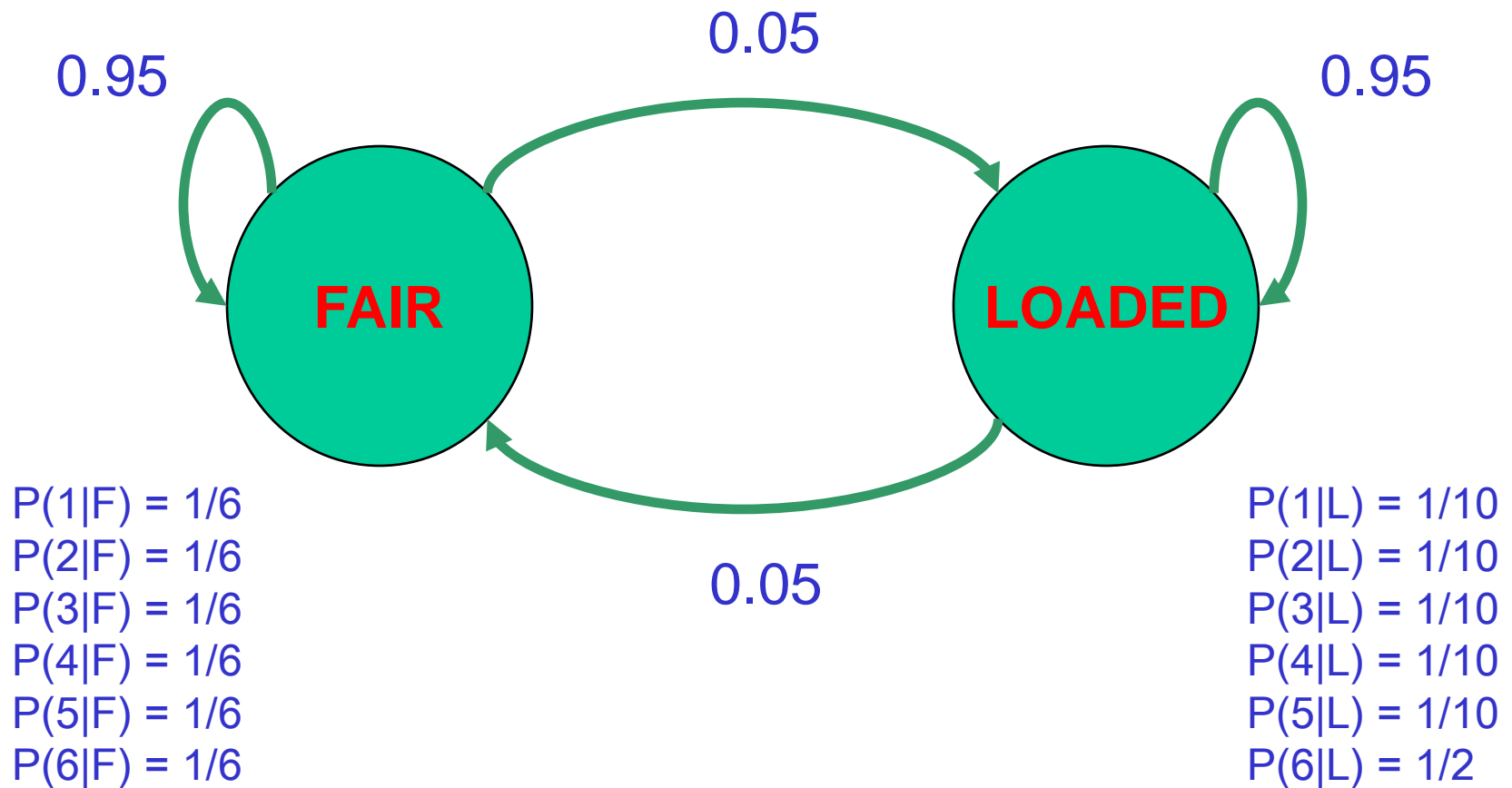
Game:

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



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

62146146136136661664661636616366163616515615115146123562344

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, and transitions

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 $s \rightarrow t$

$$a_{st} = p(t | s) = \frac{A_{st}}{\sum_{t' \in M} A_{st'}}$$

- 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

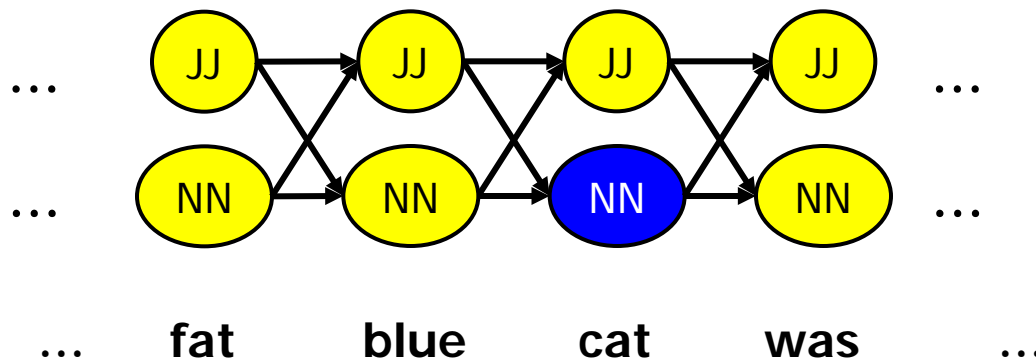
- Let's assume that $a_{ij} > 0$ and $e_i(x) > 0$ for all x, i, j and $i, j \leq k$
- Then there exist k^n path
- We cannot look at all of them

- **Viterbi-Algorithm**

- Viterbi, A. J. (1967). "Error bounds for convolution codes and an asymptotically optimal decoding algorithm." *IEEE Transact. on Information Theory* **13**: 260-269.

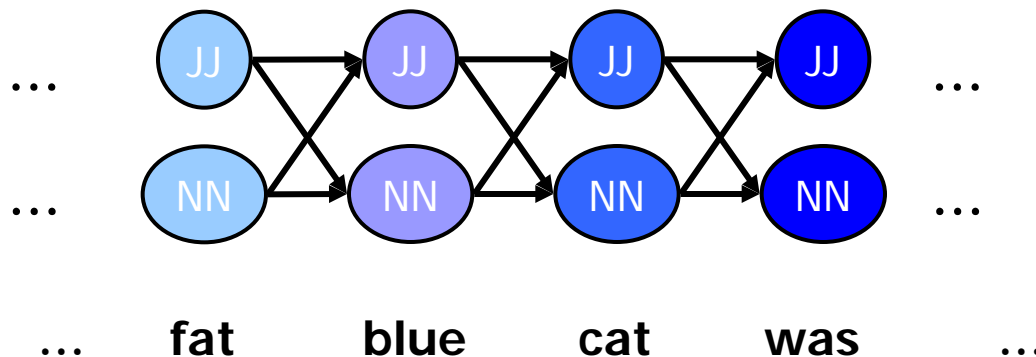
Idea: Dynamic Programming

- 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 $v_s(1), v_s(2), \dots, v_s(|S|)$ (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

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

Tabular Computation

		The	fat	blue
S_0	1	0	0	
DT	0	1	0	0
JJ	0	0
NN	0	0
NNS	0	0
VB	0	0
VBZ	0	0
...				

- 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

Result

		The	fat	blue	...	cake .
S_0	1	0	0	0	...	0
DT	0	1	0	0	...	0,004
JJ	0	0	0,0012
NN	0	0	0,034
NNS	0	0	0,0001
VB	0	0	0,002
VBZ	0	0	0,013
...						0,008

- 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 $n*k$ cells
 - For computing a cell value, we need to access all potential predecessor states ($=k$)
 - Together: $O(n*k^2)$

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

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

– 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 influenced only by the transition probabilities
 - Not very accurate
- Sophisticated approach: Use syntactical hints
 - 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)

Content of this Lecture

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

Wrap-Up

- Advantages of HMM
 - Clean framework
 - Relative simple math, linear in the sentence length
 - Good performance with **POS-tri-grams**
 - But 3-grams already need quite a large training 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 don't seem to be necessary for POS
 - Conditional Random Fields, Markov Logic Networks, ...

POS Tagger 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

- Brill, E. (1992). "A simple rule-based part of speech tagger". Conf Applied Natural Language Processing, Trento, Italy
- Brants, T. (2000). "TnT - a statistical part-of-speech tagger". Conf Applied Natural Language Processing, Seattle, USA
- Smith, L., Rindflesch, T. and Wilbur, W. J. (2004). "MedPost: a part-of-speech tagger for biomedical text." Bioinformatics