



CAMPUS  
DE EXCELENCIA  
INTERNACIONAL

POLITÉCNICA

"Ingeniamos el futuro"

# Text Mining 4

# Information Extraction

Madrid Summer School on  
Advanced Statistics and Data Mining

Florian Leitner  
Data Catalytics, S.L.  
leitner@datacaltitics.com

# Collocations (and idioms)

A **Collocation** is an expression consisting of two or more words that correspond to some conventional way of saying things.

"Collocations of a given word are statements of the habitual or customary places of that word." -Firth (1957: 181)

Collocations are **non-compositional** (i.e., the meaning of the expression cannot be understood [viz., composed] from its parts): meaning is added to the term (making it a special version of the linguistic concept of a **term**).

*strong tea, weapons of mass destruction, to make up, the rich and powerful, stiff breeze, broad daylight, white wine, ...*

*collocation test: Is a literal word-by-word translation to another language possible? (if not, it's probably a collocation)*

Manning and Schütze. Foundations of Statistical Natural Language Processing. 2000

# Detecting collocations: standard statistical methods

- Frequency-based methods
  - Measure the likelihood of bigram co-occurrence to detect "collocation-ness"



- **t-test** (using the **expected** probability  $p$  of a candidate bigram as the variance  $\sigma^2$ )
  - has problems with more frequent words and is typically only used to **rank** collocations (rather than detect them)
- **$\chi^2$ -test** (using the ratios of either word being pre-/proceeded by the other)
  - has problems when the frequency of words being compared is very small
- **PMI** (point-wise mutual information; log-ratio of the joint over the expected probability of the bigram)
  - as the other tests, has problems with data sparseness (i.e., bigrams with few examples; more robust variants use a frequency-weighted PMI)

# Detecting collocations: likelihood ratio test

- MLE assuming a **binomial distribution of bigrams**
- For bigram "word<sub>1</sub> word<sub>2</sub>", with counts  $c$  and total words  $N$ :

$$\lambda = -2 \log \frac{B(c_{12}, c_1, p_2) B(c_2 - c_{12}, N - c_1, p_2)}{B(c_{12}, c_1, p_{2|1}) B(c_2 - c_{12}, N - c_1, p_{2|\neg 1})}$$

$$B(k, n, p) = \binom{n}{k} p^k (1 - p)^{n-k} \quad p_2 = \frac{c_2}{N} \quad p_{2|1} = \frac{c_{12}}{c_1} \quad p_{2|\neg 1} = \frac{c_2 - c_{12}}{N - c_1}$$

- ▶ Asymptotically  $\chi^2$ -distributed (*explaining the "-2" multiplier*)
- Can use the  $\chi^2$  statistic to find the critical values for  $\lambda$   
*NB: d.f. is always 1 (→ bigrams!);  $\lambda > 10.83$  is pretty solid*
- ▶ Fairly **robust** with both frequent words and infrequent bigrams

Dunning. Accurate methods for the statistics of surprise and coincidence. 1993, Comp. Ling.

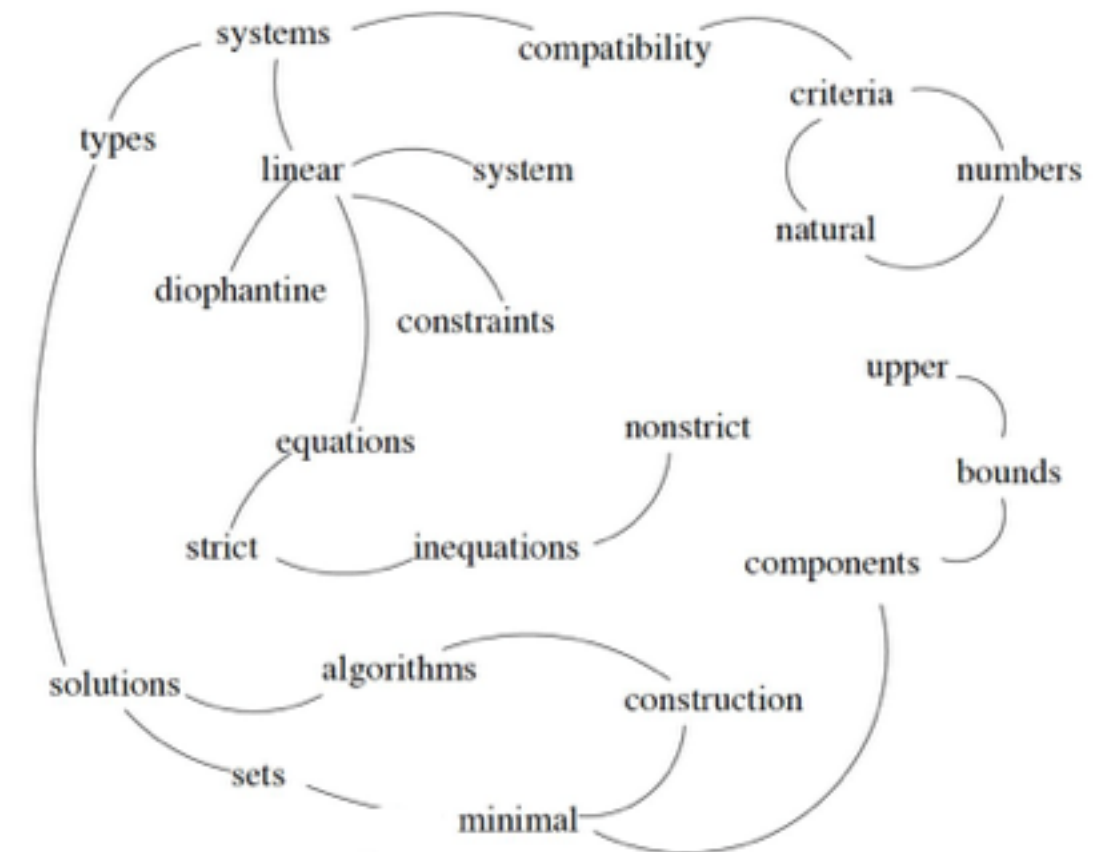
# Detecting collocations: phrasal filtering rules

- Select only terms that are valid noun and verb phrases
    - ▶ NB: not all collocations are noun phrases, e.g. "to make up".
    - ▶ Advantage: Can be applied after any statistical selection method.
    - ▶ Disadvantage: Requires tagging tokens with their part-of-speech.
  - i.e., this is a fully supervised approach
  - [Part-of-speech tagging is coming up later in this lecture]
  - ▶ Justeson & Katz, Comp. Ling., 1995, suggested the following rules:
    - [Adjective | Noun] - [Noun] → *"strong tea", "noun phrase"*
    - [Adjective] - [Adjective | Noun] - Noun → *"free legal advice", "online web tutorial"*
    - Noun - [Adjective | Noun] - Noun → *"mean squared error", "New York City"*
    - Noun - Preposition - Noun → *"parts of speech", "rich and powerful"*
- NB: no rules for verb phrases were suggested (far more tricky)*

# Keyword Extraction with TextRank using Co-occurrence

see day 2 for details

Compatibility of systems of linear constraints over the set of natural numbers. Criteria of compatibility of a system of linear Diophantine equations, strict inequations, and nonstrict inequations are considered. Upper bounds for components of a minimal set of solutions and algorithms of construction of minimal generating sets of solutions for all types of systems are given. These criteria and the corresponding algorithms for constructing a minimal supporting set of solutions can be used in solving all the considered types systems and systems of mixed types.



**Keywords assigned by TextRank:**

linear constraints; linear diophantine equations; natural numbers; nonstrict inequations; strict inequations; upper bounds

**Keywords assigned by human annotators:**

linear constraints; linear diophantine equations; minimal generating sets; non-strict inequations; set of natural numbers; strict inequations; upper bounds

Mihalcea, R., and Tarau, P. (2004). TextRank: Bringing order into texts.

# Keyword Extraction using phrase-breaking lists: RAKE

- Split sentences, then phrases, at stop-words and punctuation.
- Next, proceed with the same "word graph" evaluation as TextRank
- The authors claim RAKE performs (slightly) better than TextRank
  - ▶ Caveat emptor: manual stop-word tuning required
    - Performance depends on stop-word quality (wrt. both hits, and misses)
    - requires a language where stop-words are not common inside keywords (e.g., in Spanish, "de", commonly used to chain noun phrases, might be an issue)

Rose, S., Engel, D., Cramer, N., and Cowley, W. (2010). Automatic keyword extraction from individual documents.



# Keyword Extraction Alternatives

- Use the PageRank (here: "TextRank") approach as foundation
- But instead of stop-word delimiters or all n-grams, use
  - **Collocations** *(beginning of this day)*
  - **Noun Phrases** *(coming up next)*



# Probabilistic language modeling

- ▶ Manning & Schütze. Statistical Natural Language Processing. 1999
- A sentence  $W$  is defined as a **sequence** of words  $w_1, \dots, w_n$
- Probability of **next word**  $w_n$  in a sentence is:  $P(w_n | w_1, \dots, w_{n-1})$ 
  - ▶ a **conditional probability**
- The probability of the **whole sentence** is:  $P(W) = P(w_1, \dots, w_n)$ 
  - ▶ the **chain rule of conditional probability**
- These counts & probabilities form the **language model** [for a given document collection (= **corpus**)].
  - ▶ the model variables are **discrete** (counts)
  - ▶ only needs to deal with **probability mass** (not density)

# Modeling the stochastic process of “generating words” using the chain rule

“This is a long sentence with many words...” ➔

$$P(W) = P(\text{this}) \times$$

$$P(\text{is} \mid \text{this}) \times$$

$$P(\text{a} \mid \text{this, is}) \times$$

$$P(\text{long} \mid \text{this, is, a}) \times$$

$$P(\text{sentence} \mid \text{this, is, a, long}) \times$$

$$P(\text{with} \mid \text{this, is, a, long, sentence}) \times$$

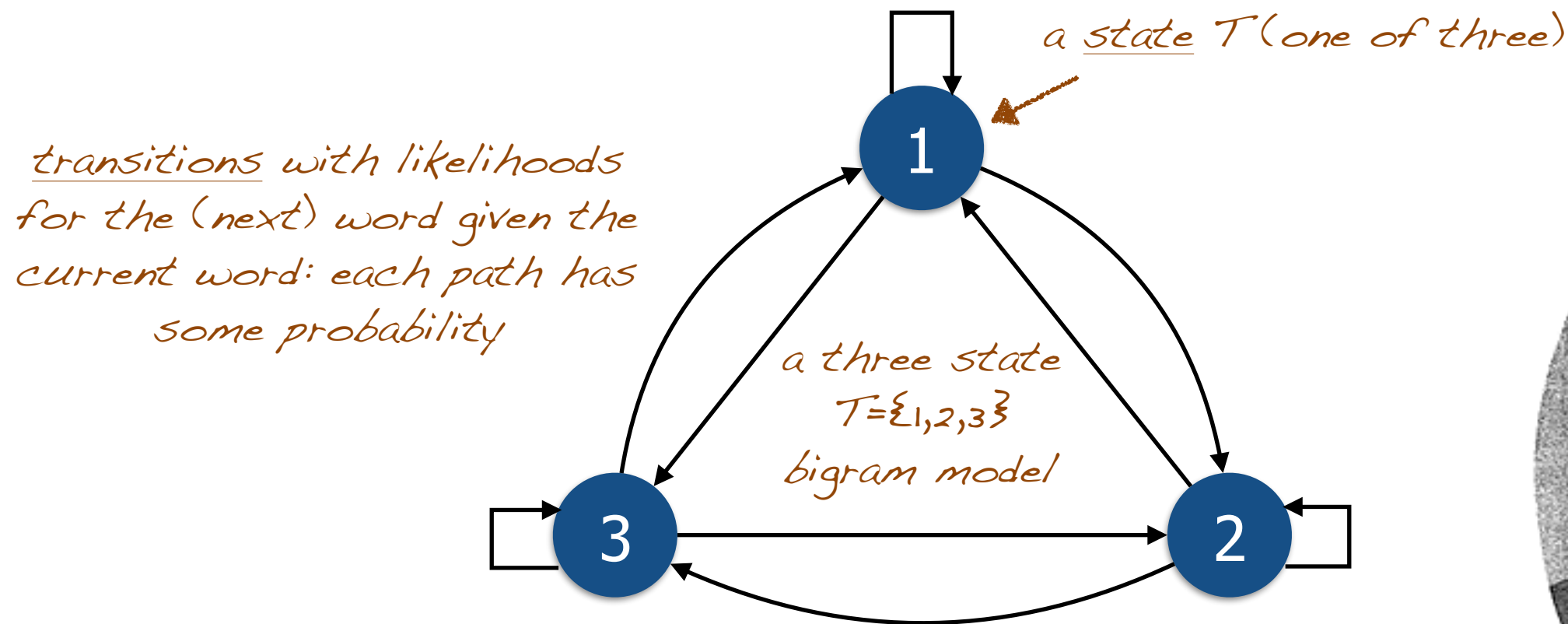
$$P(\text{many} \mid \text{this, is, a, long, sentence, with}) \times$$

....

n-grams for  $n > 5$ :  
insufficient (**sparse**) data  
(and expensive to calculate)

# The Markov property

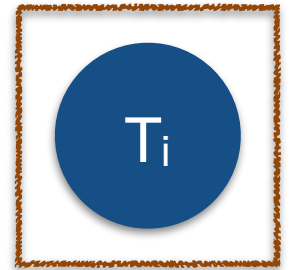
A Markov process is a stochastic process whose future (next) state only depends on the current state  $T$ , but not its past. ← a bigram!



# Modeling the stochastic process of “generating words” assuming it is Markovian

*stochastic process:  
“Unigram Model”*

$$\prod P(w_i | w_1^{i-1}) \approx \prod P(w_i | w_{i-k}^{i-1})$$



- 1<sup>st</sup> Order Markov Chains

- ▶ Bigram Model,  $k=1$ ,  $P(\text{be} \mid \text{this})$

- 2<sup>nd</sup> Order Markov Chains

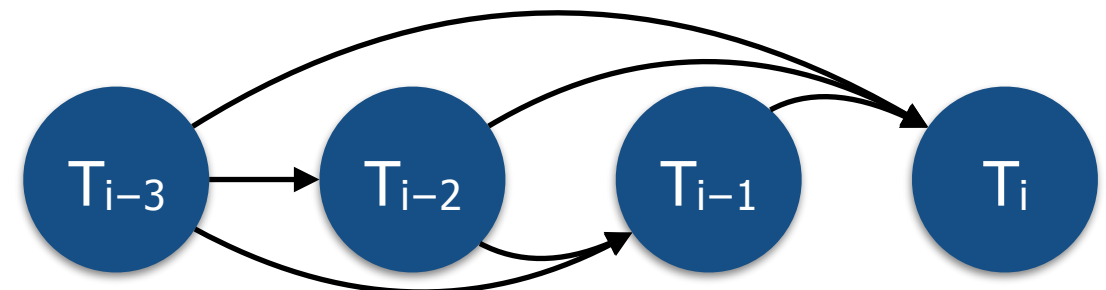
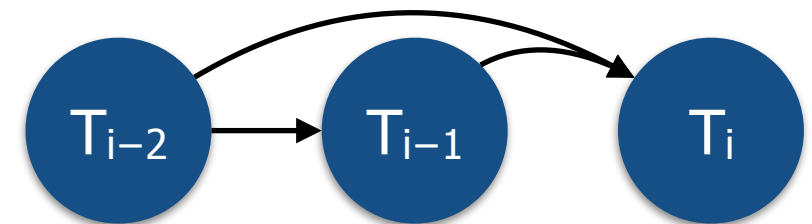
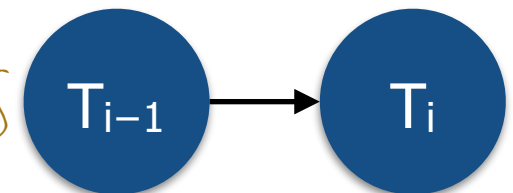
- ▶ Trigram Model,  $k=2$ ,  $P(\text{long} \mid \text{this}, \text{be})$

- 3<sup>rd</sup> Order Markov Chains

- ▶ Quadrigram Model,  $k=3$ ,  
 $P(\text{sentence} \mid \text{this}, \text{be}, \text{long})$

- ...

*simplified version of the former  
diagram (“The Markov property”)*



*dependencies could span over a dozen tokens, but these  
sizes are generally sufficient to work by*

# Measuring n-gram (transition) probabilities

- Unigrams:

$N$  = total word count

$$P(w_i) = \frac{\text{count}(w_i)}{N}$$

- Bigrams:

$$P(w_i|w_{i-1}) = \frac{\text{count}(w_{i-1}, w_i)}{\text{count}(w_{i-1})}$$

- **N-grams** ( $n=k+1$ ):

$$P(w_i|w_{i-k}^{i-1}) = \frac{\text{count}(w_{i-k}^i)}{\text{count}(w_{i-k}^{i-1})}$$

## ► Language Model:

$$P(W) = \prod P(w_i|w_{i-k}^{i-1}) = \\ = \prod P(w_i|w_{i-k}, \dots, w_{i-1})$$

*Programming tip:  
transform probabilities  
to logs to avoid underflows  
and work with addition*

$k$  = n-gram size - 1

# The Parts of Speech (PoS)

I	ate	the	pizza	with	green	peppers	.
PRP	VB	DT	NN	IN	JJ	NN	.

*"PoS tags"*

- The Parts of Speech:

- ▶ noun: **NN**, verb: **VB**, adjective: **JJ**, adverb: **RB**, preposition: **IN**, personal pronoun: **PRP**, ...
- ▶ e.g. the full **Penn Treebank PoS tagset** contains 48 tags:
- ▶ 34 grammatical tags (i.e., "real" parts-of-speech) for words
- ▶ one for cardinal numbers ("CD"; i.e., a series of digits)
- ▶ 13 for [mathematical] "SYM" and currency "\$" symbols, various types of punctuation, as well as for opening/closing parenthesis and quotes

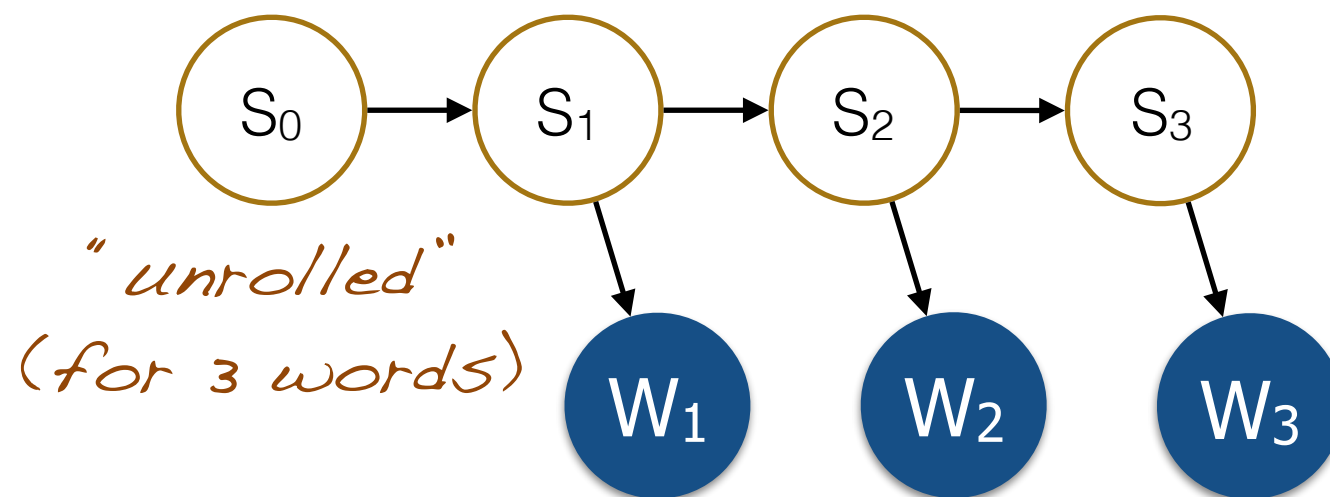
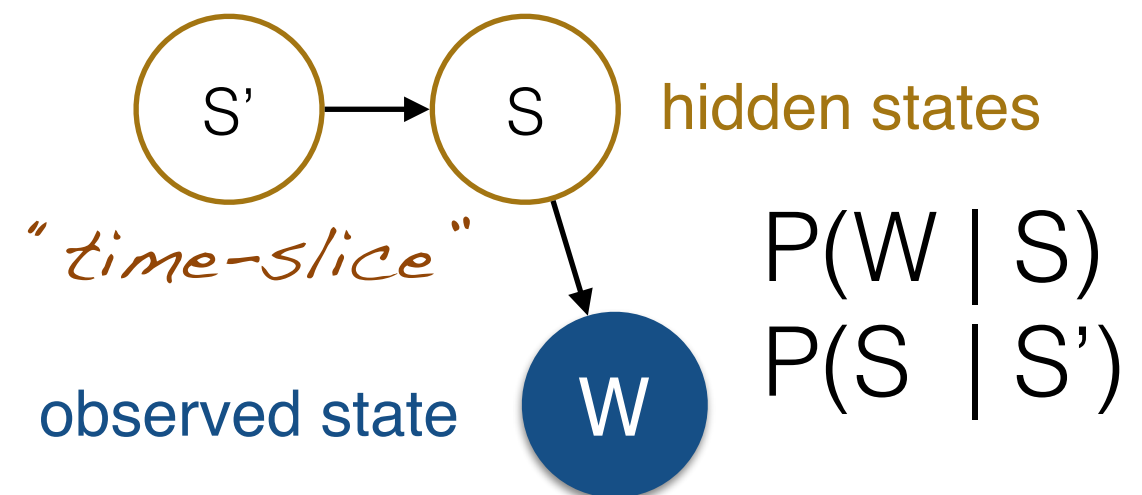
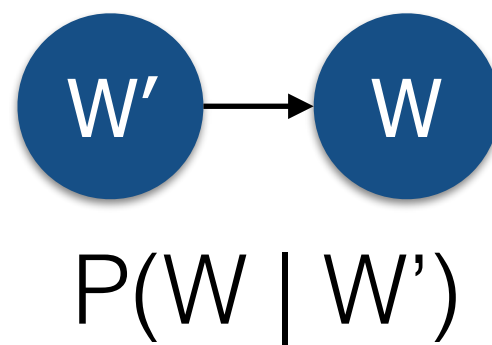
# The Parts of Speech (2/2)

I	ate	the	pizza	with	green	peppers	.
PRP	VB	DT	NN	IN	JJ	NN	.

- Automatic PoS tagging ➡ supervised machine learning
  - Maximum Entropy Markov models
  - **Conditional Random Fields**
  - **Convolutional Neural Networks**
  - Ensemble methods (bagging, boosting, etc.)



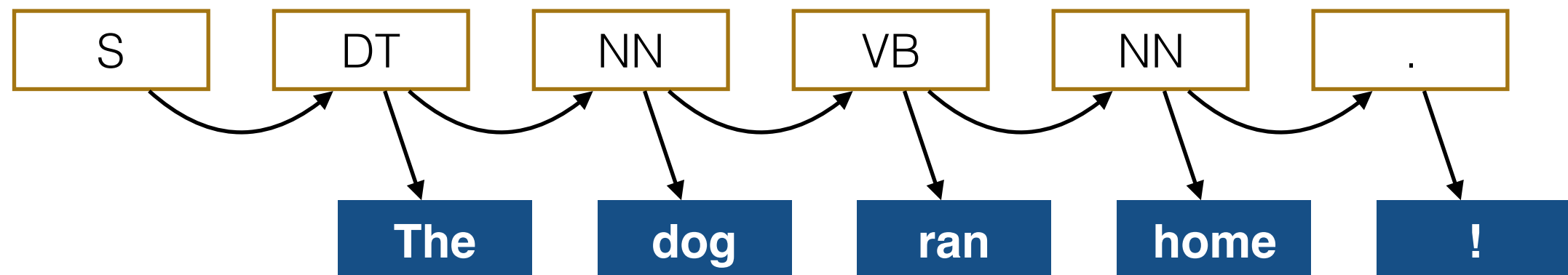
# From a Markov chain to a Hidden Markov Model (HMM)



*$W$  depends on  $S$   
and  $S$  in turn  
depends on  $S'$*

# A language-based intuition for HMMs

- A Markov Chain:  $P(W) = \prod P(w \mid w')$ 
  - assumes the observed words are in and of themselves the cause of the observed sequence.
- A HMM:  $P(S, W) = \prod P(s \mid s') P(w \mid s)$ 
  - assumes the observed words are emitted by a hidden (not observable) sequence, for example the chain of part-of-speech-states.



*NB, this is the "unrolled" model that does not depict the conditional dependencies*

# The two matrices of a HMM

*a.k.a. "CPDs": Conditional Probability Distributions*

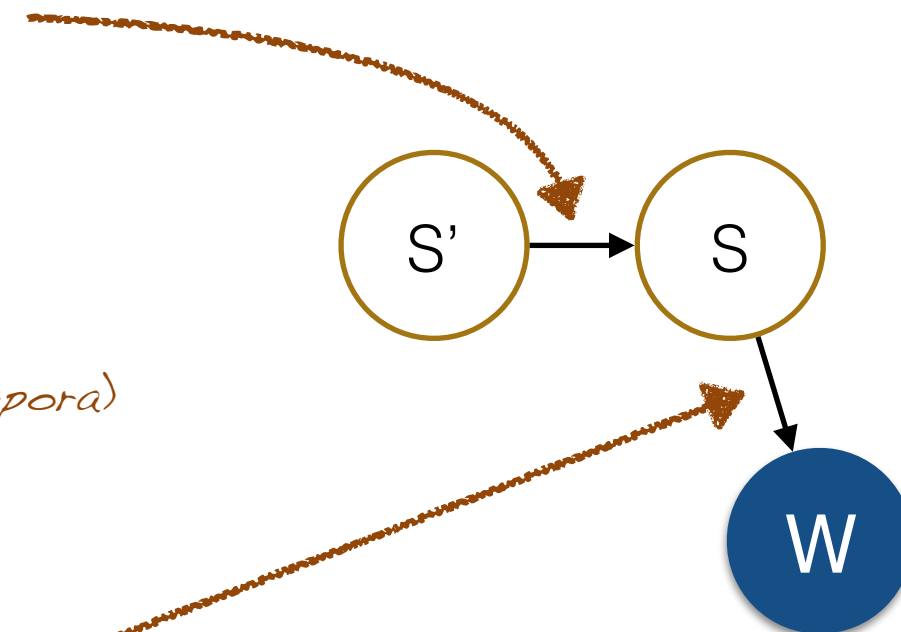
$P(s   s')$	DT	NN	VB	...
DT	0.03	0.7	0	
NN	0	0	0.5	
VB	0	0.5	0.2	
...				

*(measured as discrete factor tables from annotated PoS corpora)*

$P(w   s)$	word <sub>1</sub>	word <sub>2</sub>	word <sub>3</sub>	...
DT	0.3	0	0	
NN	0.0001	0.002	0	
VB	0	0	0.001	
...				

*underflow danger  $\Rightarrow$  use "log Ps"!*

## Transition Matrix



## Observation Matrix

*very sparse ( $W$  is large)  
 $\Rightarrow$  Smoothing!*

# Three limitations of HMMs

“unsolved”

MEMM

CRF

**Markov assumption:** The next state only depends on the current state.

Example issue: trigrams

*(long-range dependencies!)*

**Output assumption:** The output (observed value) is independent of all previous outputs (given the current state).

Example issue: word morphology

*(inflection, declension!)*

**Stationary assumption:** Transition probabilities are independent of the actual time when they take place.

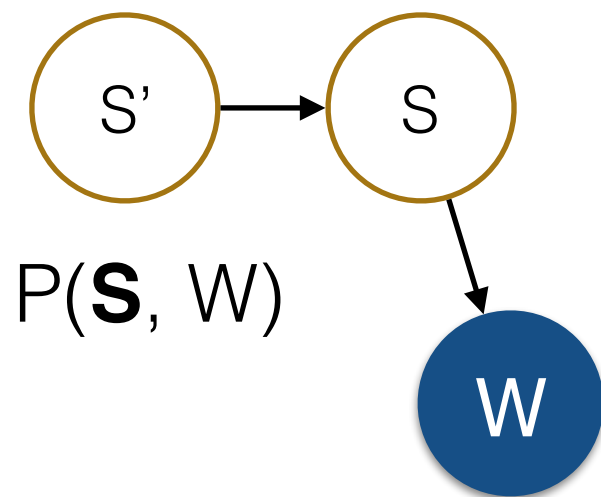
Example issue: position in sentence

*(label bias problem, see next!)*

(CRF)

# From generative to discriminative sequence models

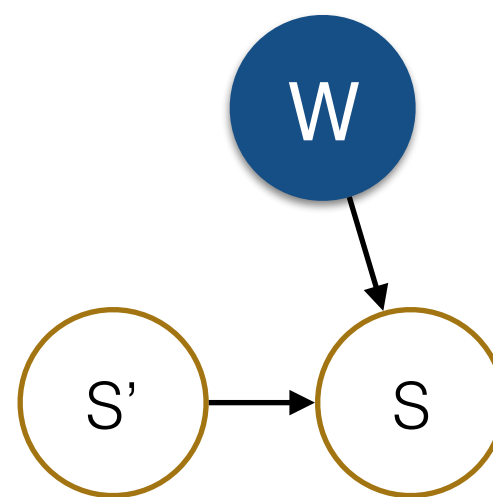
Hidden Markov Model  
(first order version)



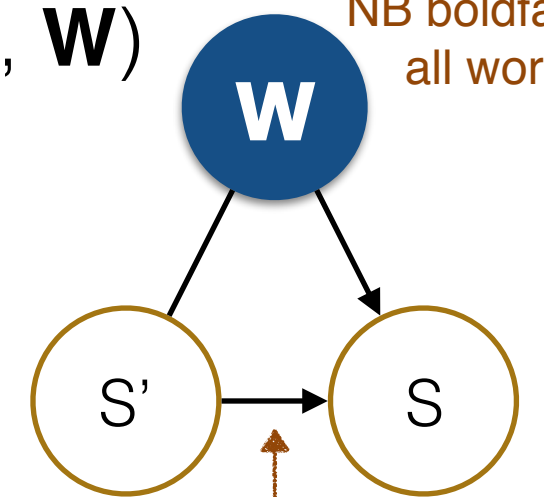
generative model; lower bias is beneficial for small training sets

Conditional Random Field  
(linear chain version)

$P(S | S', \mathbf{W})$  NB boldface  $\mathbf{W}$ : all words!



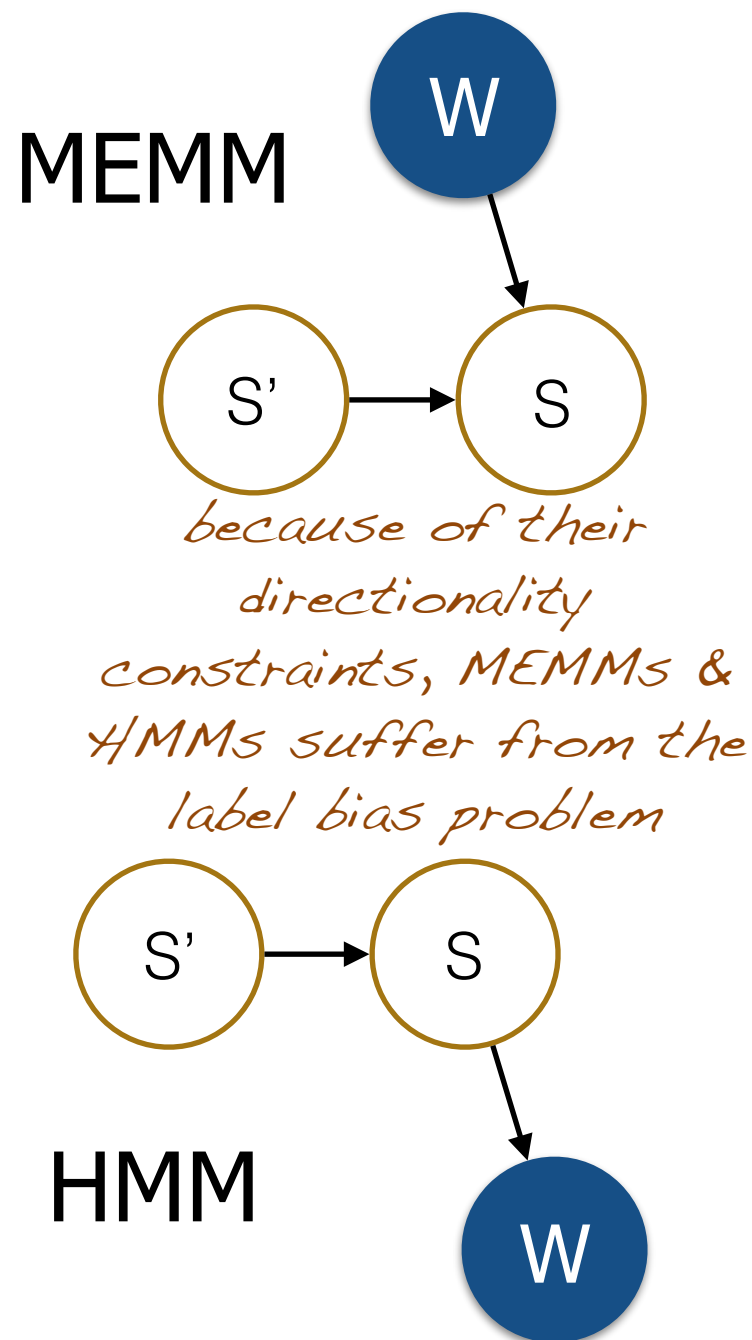
$P(S | S', \mathbf{W})$



this "clique" makes CRFs expensive to compute

Maximum Entropy Markov Model  
(first order version)

# The **label bias problem** of directional Markov models



The robot wheels Fred around.

DT **NN** **VB** NN RB

The robot wheels were broken.

DT **NN** **NN** VB JJ

The robot wheels are round.

DT **NN** **??** yet unseen!

Wallach. Efficient Training of CRFs. MSc 2002

# Model summaries: HMM, MEMM, CRF

- A **HMM**

- ▶ **generative** model
- ▶ **efficient** to learn and deploy
- ▶ trains with **little data**
- ▶ generalizes well (**low bias**)

- A **MEMM**

- ▶ better labeling **performance**
- ▶ modeling of **features**
- ▶ **label bias** problem

- **CRF**

- ▶ conditioned on **entire observation**
- ▶ **complex features** over full input
- ▶ training time **scales exponentially**
- $O(NTM^2G)$
- N: # of sequence pairs;  
T: E[sequence length];  
M: # of (hidden) states;  
G: # of gradient computations for parameter estimation

a PoS model w/45 states and 1 M words can take a week to train...

Sutton & McCallum. An Introduction to CRFs for Relational Learning. 2006



# The Parts of Speech

I	ate	the	pizza	with	green	peppers	.
PRP	VB	DT	NN	IN	JJ	NN	.

- **Corpora** for the [supervised] **training** of PoS taggers
  - ▶ **Brown** Corpus (AE from ~1961)
  - ▶ British National Corpus: **BNC** (20<sup>th</sup> century British)
  - ▶ Wall Street Journal Corpus: **WSJ Corpus** (AE from the 80s)
  - ▶ American National Corpus: **ANC** (AE from the 90s)
  - ▶ Lancaster Corpus of Mandarin Chinese: **LCMC** (Books in Mandarin)
  - ▶ The **GENIA** corpus (Biomedical abstracts from PubMed)
  - ▶ **NEGR@** (German Newswire from the 90s)
  - ▶ Spanish and Arabian corpora should be (commercially...) available... ???

Best tip: Ask on the “corpora list” mailing list!

# Linguistic morphology

→ token normalization

## ● [Verbal] **Inflections**

- ▶ conjugation (Indo-European languages)
- ▶ tense (“availability” and use varies across languages)
- ▶ modality (subjunctive, conditional, imperative)
- ▶ voice (active/passive)
- ▶ ...

## ● **Contractions** *not a contraction: possessive s*

- ▶ don't say you're in a man's world...

## ● **Declensions**

- on nouns, pronouns, adjectives, determiners
- ▶ case (nominative, accusative, dative, ablative, genitive, ...)
- ▶ gender (female, male, neuter)
- ▶ number forms (singular, plural, dual)
- ▶ possessive pronouns (I→my, you→your, she→her, it→its, ... car)
- ▶ reflexive pronouns (for myself, yourself, ...)
- ▶ ...

# Stemming → Lemmatization

⇒ token normalization

*a.k.a. token "regularization"*

*(although that is technically the wrong wording)*

- **Stemming**

- ▶ produced by "**stemmers**"
- ▶ produces a word's "stem"
- ▶ am → am
- ▶ the going → the go
- ▶ having → hav
- ▶ fast and simple (pattern-based)
- ▶ **Snowball; Lovins; Porter**

- **Lemmatization**

- ▶ produced by "**lemmatizers**"
- ▶ produces a word's "lemma"
- ▶ am → be
- ▶ the going → the going
- ▶ having → have
- ▶ requires: a dictionary for the mappings and the **PoS tags (!)**
- ▶ **LemmaGen; morpha; BioLemmatizer; geniatagger**

# Noun and verb phrase chunking with BIO-encoded labels

“shallow parsing”

a pangram (hint: check the letters)

The brown fox	quickly jumps	over	the lazy dog .
DT JJ NN	RB VBZ	IN	DT JJ NN .
B-N I-N I-N	B-V I-V	O	B-N I-N I-N O

Performance (2<sup>nd</sup> order CRF) ~ 94%

Main problem: embedded & chained NPs (N of N and N)

**Chunking** is “more robust to the highly diverse corpus of text on the Web”  
and [exponentially] faster than [deep] parsing.

Banko et al. Open Information Extraction from the Web. IJCAI 2007 [a paper with over 700 citations](#)

Wermter et al. Recognizing noun phrases in biomedical text. SMBM 2005 **error sources**

# PoS tagging and lemmatization for Named Entity Recognition (NER)

N.B.: This is all supervised (i.e., manually annotated corpora)!

de facto standard  
PoS tagset  
{NN, JJ, DT, VBZ, ...}  
**Penn Treebank**

*noun-phrase (chunk)*

Token	PoS	Lemma	NER
Constitutive	JJ	constitutive	O
binding	NN	binding	O
to	TO	to	O
the	DT	the	O
peri-κ	NN	peri-kappa	B-DNA
B	NN	B	I-DNA
site	NN	site	I-DNA
is	VBZ	be	O
seen	VCN	see	O
in	IN	in	O
monocytes	NNS	monocyte	B-cell
.	.	.	O

**Begin-Inside-Outside**  
*of the (relevant) token*  
**B-I-O**  
**chunk encoding**

common alternatives:  
I-O  
I-E-O  
B-I-E-W-O

End token  
(unigram) Word

# Named Entity Recognition (NER)

Image Source: v@s3k [a GATE session; <http://vas3k.ru/blog/354/>]

The departure of Mr Hogan, who originally moved to British Midland as service director from Hertz International in 1997, surprised aviation analysts, as it was believed that he had been brought into the senior executive team of the airline, as part of the group's management succession planning.

He played a leading role in the strategic planning for the rebranding of the airline as BMI in preparation for its entry this year into the scheduled long haul market with the launch of services from Manchester to the US.

BMI has taken on the costs of entry into the North Atlantic market at an unfortunate time, as airlines in North America are facing the toughest conditions for 20 years with many carriers plunging into loss.

BMI, in which Lufthansa of Germany and SAS Scandinavian Airlines each own stakes of 20 per cent, suffered a 26 per cent fall in pre-tax profits last year from £11.1m (\$15.7m) to £8.2m on a turnover that grew 16.5 per cent to £739.2m.

In the first six months this year it is understood that passenger volumes have fallen by around two per cent. The share of available seats filled, the load factor, has declined by around two percentage points, but this has been offset by a strong increase in yields, or average fare levels, by more than ten per cent.

How much training  
data do I need?  
“corpora list”

**Date**

**Location**

**Money**

**Organization**

**Percentage**

**Person**

- ➔ **Conditional Random Field**
- ➔ Ensemble Methods; +SVM, HMM, MEMM, ... ➔ `pyensemble`  
NB these are **corpus-based** approaches (supervised)  
CoNLL03: <http://www.cnts.ua.ac.be/conll2003/ner/>

# Gazetteers:

## Dictionary matching

- Finding all tokens that match dictionary entries

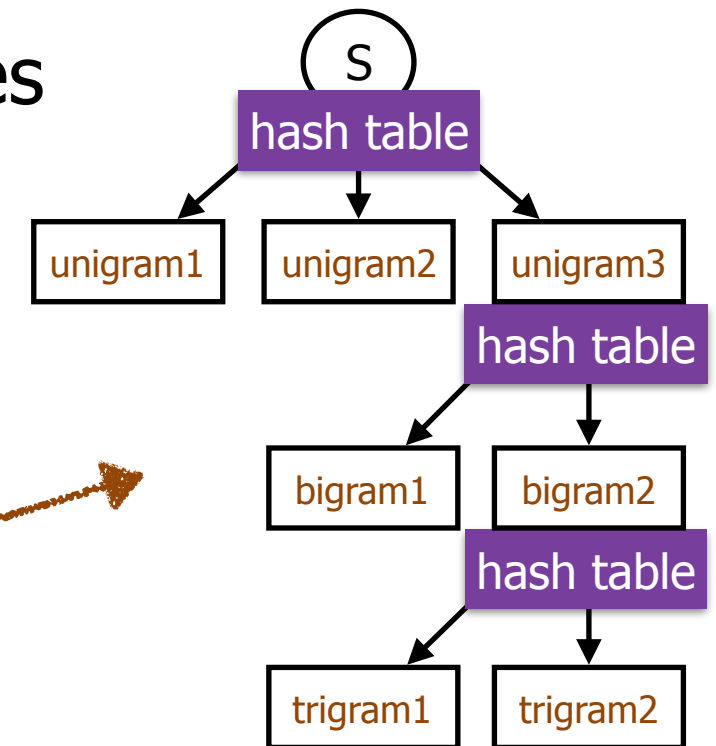
- hash table lookups: constant complexity -  $O(1)$

- **Exact**, single token matches

- regular hash table lookup (e.g., MURMUR3)

- Exact, multiple tokens

- **prefix trie** of (hashed) tokens



- **Approximate**, single tokens *e.g. Jaro-Winkler similarity*

- use some string metric - but do not compare all n-to-m cases...

- **Approximate**, multiple tokens

- various "fuzzy" sequence matching algos

*efficient/traditional approach:  
character n-gram similarity  
(e.g. databases)*