POLITÉCNICA
"Ingeniamos el futuro"

CAMPUS
DE EXCELENCIA
INTERNACIONAL

Madrid Summer School on
Advanced Statistics and Data Mining

Module C9 :: Text Mining

4th July - 8th of July 2016

Florian Leitner
Data Catalytics
leitner@datacatytics.com

Applications of text mining and language processing

Text mining

(Web) Search engines

Information retrieval

Spam filtering

Document classification

Twitter brand monitoring

Opinion mining

Finding similar items (Amazon)

Content-based recommender systems

Event detection in e-mail

Information extraction

Language processing

Spelling correction

Statistical language modeling

Siri (Apple) and Google Now

Language understanding

Website translation (Google)

Machine translation

"Clippy" assistant (Microsoft)

Dialog systems

Watson in Jeopardy! (IBM)

Question answering

Ambiguity

Part-of-Speech tagging

The robot **wheels** out the iron.

Paraphrasing

Unemployment is on the rise.

vs

The economy is slumping.

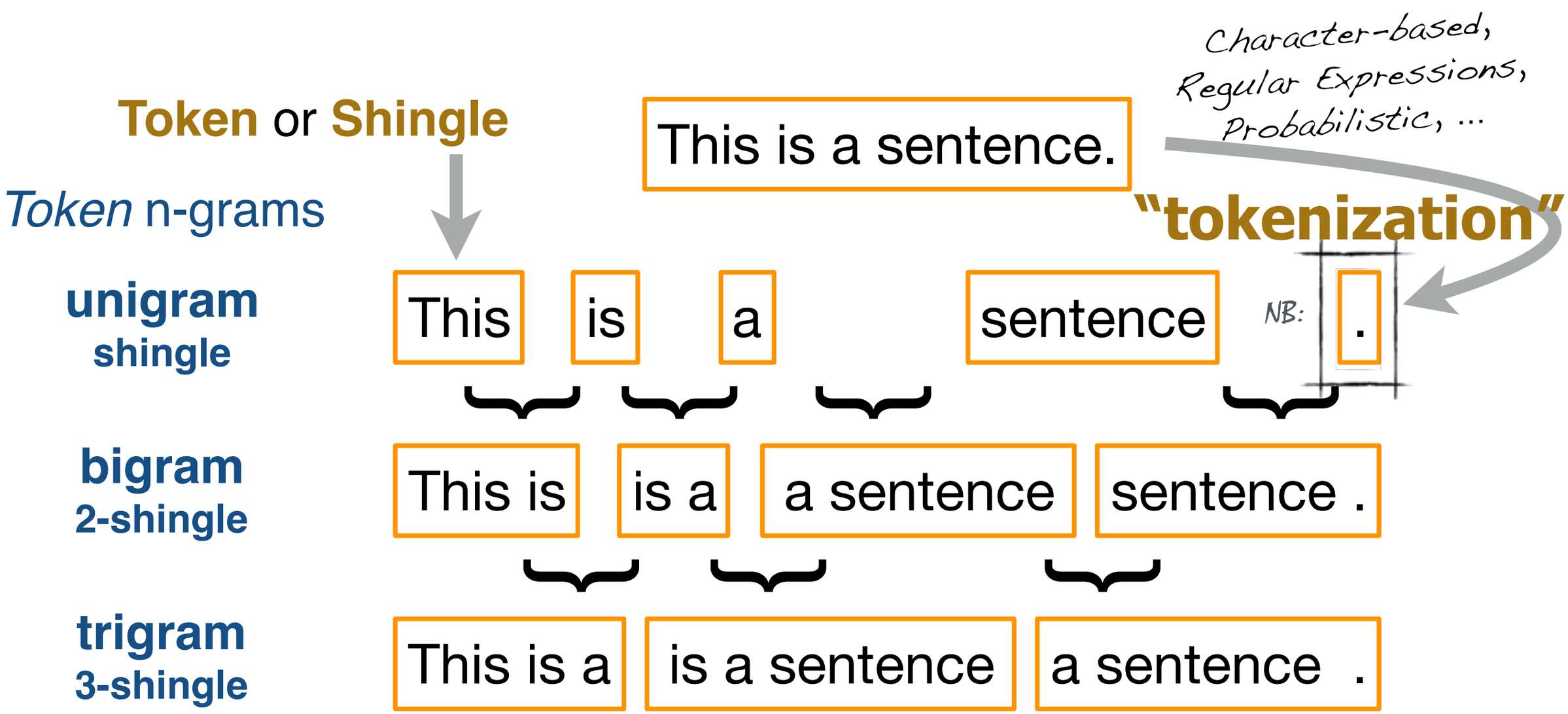
Anaphora resolution

Carl and Bob were fighting:
"**You** should shut up,"
Carl told **him**.

Named entity recognition

Is **Paris** really good for you?

Tokens and n-grams



Beware: the terms "shingle", "token" and "n-gram" are not used consistently...

Text Mining 1
Information Retrieval

Madrid Summer School on
Advanced Statistics and Data Mining

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

Converting tokens to numbers (part 1)

- **Tokenization** is the process of splitting text into words, punctuation, and symbols (aka. tokens).
- **Indexing** can refer to linking tokens to document and counting the frequencies of each token...
  - within a document: (token or) **term frequency TF**
  - number of documents with that token: **document frequency DF**
  - overall count in your document collection: **corpus frequency CF**
- The remaining question then is: how to make computations with these numbers?

The inverted index

factors, normalization (len[text]), probabilities, and n-grams

Text 1: He that not wills to the end neither
wills to the means.

Text 2: If the mountain will not go to Moses,
then Moses must go to the mountain.

tokens	Text 1	Text 2
end	1	0
go	0	2
he	1	0
if	0	1
means	1	0
Moses	0	2
mountain	0	2
must	0	1
not	1	1
that	1	0
the	2	2
then	0	1
to	2	2
will	2	1

unigrams	T1	T2	p(T1)	p(T2)
end	1	**0**	0.09	0.00
go	**0**	2	0.00	0.13
he	1	**0**	0.09	0.00
if	**0**	1	0.00	0.07
means	1	**0**	0.09	0.00
Moses	**0**	2	0.00	0.13
mountain	**0**	2	0.00	0.13
must	**0**	1	0.00	0.07
not	1	1	0.09	0.07
that	1	**0**	0.09	0.00
the	2	2	0.18	0.13
then	**0**	1	0.00	0.07
to	2	2	0.18	0.13
will	2	1	0.18	0.07
SUM	11	15	1.00	1.00

bigrams	Text 1	Text 2
end, neither	1	**0**
go, to	**0**	2
he, that	1	**0**
if, the	**0**	1
Moses, must	**0**	1
Moses, then	**0**	1
mountain, will	**0**	1
must, go	**0**	1
not, go	**0**	1
not, will	1	**0**
that, not	1	**0**
the, means	1	**0**
the, mountain	**0**	2
then, Moses	**0**	1
to, Moses	**0**	1
to, the	**2**	**1**
will, not	**0**	1
will, to	2	**0**

Document similarity

- Similarity measures
  - **Cosine similarity (of document/text vectors)**
  - **Correlation coefficients**
- Word vector normalization
  - **TF-IDF**
- Dimensionality reduction/clustering
  - **Locality sensitivity hashing**
  - **Latent semantic indexing**
  - **Latent Dirichlet allocation** (tomorrow)

Word vectors

Collections of vectorized texts: **Inverted index**

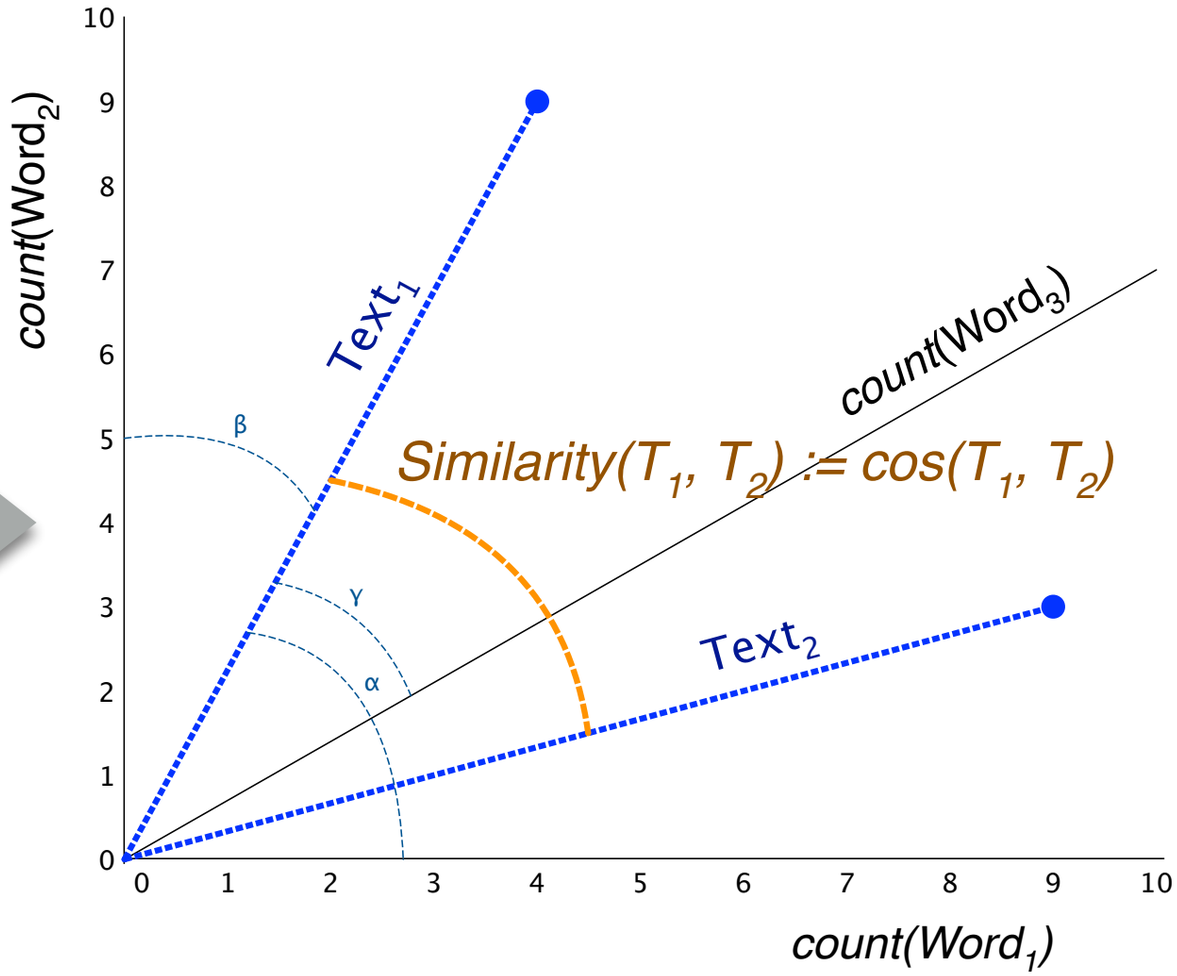
Text 1: He that not wills to the end neither wills to the means.
Text 2: If the mountain will not go to Moses, then Moses must go to the mountain.

each token/word is a dimension!

tokens	Text 1	Text 2
end	1	0
go	0	2
he	1	0
if	0	1
means	1	0
Moses	0	2
mountain	0	2
must	0	1
not	1	1
that	1	0
the	2	2
then	0	1
to	2	2
will	2	1

↑ word vector ↑ word vector

Comparing word vectors: **Cosine similarity**



Terrier Sphinx Xapian INDRI Lucene

Cosine similarity

$$sim(\vec{x},\vec{y}) = cos(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum x_i y_i}{\sqrt{\sum x_i^2}\sqrt{\sum y_i^2}}$$

can be dropped if using unit vectors ("length-normalized" a.k.a. "cosine normalization") only dot-product is now left: extremely efficient ways to compute on modern CPUs

- Define a similarity score between two document vectors (or the query vector in Information Retrieval)

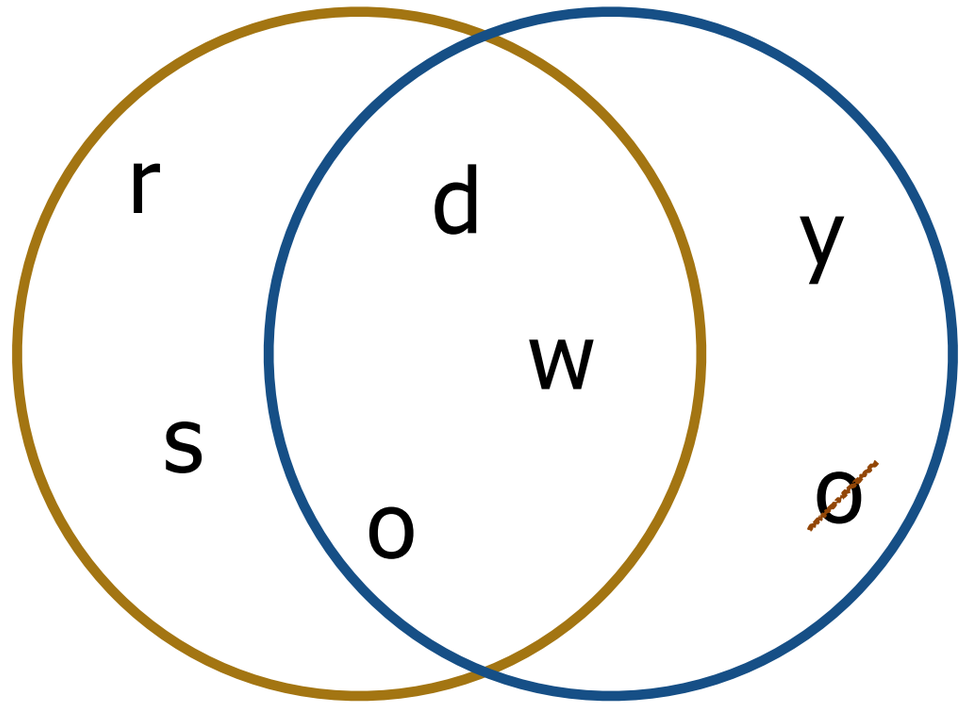
- **Euclidian** vector **distance** is **length dependent**
- The **cosine** [angle] between two vectors **is not**

$$sim(T_1,T_2) = \frac{1*1+2*2+2*2+2*1}{\sqrt{(5+3*4)} * \sqrt{(5+5*4)}} = 0.5336$$

tokens	Text 1	Text 2
end	1	0
go	0	2
he	1	0
if	0	1
means	1	0
Moses	0	2
mountain	0	2
must	0	1
not	1	1
that	1	0
the	2	2
then	0	1
to	2	2
will	2	1

Jaccard's [set] similarity

"words" vs "woody"



ABC
vs.
ABCABCABC

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{3}{7} = 0.43$$

6 0.5

Alternative similarity coefficients

- **Spearman's rank correlation coefficient** ρ (r[ho])
  - Ranking is done by term frequency (**TF**; count)
  - Critique: sensitive to ranking differences that are likely to occur with high-frequency words (e.g., "the", "a", …) ➡ use the **log** of the term count, rounded to two significant digits
    - NB that this is not relevant when only short documents (e.g. titles) with low TF counts are compared
- **Pearson's chi-square test** χ^2
  - Directly on the TFs (counts) - intuition:
    Are the TFs "random samples" from the same base distribution?
  - Usually, χ^2 should be preferred over ρ (Kilgarriff & Rose, 1998)
    - NB that both measures have no inherent normalization of document size
      - preprocessing might be necessary!

Term Frequency times Inverse Document Frequency (TF-IDF)

- **Motivation and background**
- The problem
  - **Frequent terms** contribute most to a document vector's direction, but **not all** terms are **relevant** ("the", "a", …).
- The goal
  - **Separate** important terms from frequent, but **irrelevant terms** in the collection.
- The idea
  - Frequent **terms** appearing **in all documents** tend to be less important **versus** frequent terms in just a **few documents**. → Zipf's Law!
  - Also **dampens** the effect of **topic-specific noun phrases** or an **author's bias** for a specific set of adjectives

Term Frequency times Inverse Document Frequency (TF-IDF)

- **tf.idf**(w) := tf(w) × idf(w)
  - tf: (document-specific) term frequency
  - idf: inverse (global) document frequency

- $\mathbf{tf_{natural}}(w) := \text{count}(w)$
  - $tf_{natural}$: n. of times term w occurs in a document
- $\mathbf{tf_{log}}(w) := \log(\text{count}(w) + 1)$
  - tf_{log}: the TF is smoothed by taking its log

- $\mathbf{idf_{natural}}(w) := N / \sum^{N} \{w_i > 0\}$
  - $idf_{natural}$: n. documents divided by n. documents in which term w occurs
- $\mathbf{idf_{log}}(w) := \log(N / \sum^{N} \{w_i > 0\})$
  - idf_{log}: the IDF is smoothed by taking its log
  - where N is the **number of documents**, w_i the **count of word** w in document i, and $\{w_i>0\}$ is 1 if document i has word w or 0 otherwise

TF-IDF in information retrieval

- Document vectors = tf_{log} → i.e. the doc. vectors do not use any IDF weighting (because its more efficient: the QV uses IDF, and that gets multiplied with the DV values)
- Query vector = tf_{log} idf_{log}
  - Terms are counted on each individual document & the query
- Cosine vector length normalization for TF-IDF scores:
  - Document W normalization

$$\sqrt{\sum_{w \in W} tf_{log}(w)^2}$$

  - Query Q normalization

$$\sqrt{\sum_{q \in Q} (tf_{log}(q) \times idf_{log}(q))^2}$$

- IDF is calculated over the indexed collection of all documents

TF-IDF query score: An example

	Collection		Query Q				Document D			Similarity
Term	df	idf_{log}	tf	tf_{log}	tf.idf	norm	tf	tf_{log}	tf.1	cos(Q,D)
best	3.5E+05	1.46	1	0.30	0.44	0.21	**0**	0.00	0.00	0.00
text	2.4E+03	3.62	1	0.30	1.09	0.53	10	1.04	0.06	0.03
mining	2.8E+02	4.55	1	0.30	1.37	0.67	8	0.95	0.06	0.04
tutorial	5.5E+03	3.26	1	0.30	0.98	0.48	3	0.60	0.04	0.02
data	9.2E+05	1.04	**0**	0.00	0.00	0.00	10	1.04	0.06	0.00
…	…	…	…	…	0.00	0.00	…	16.00	…	0.00
Sums	1.0E+07		4		2.05		~355	16.11		**0.09**

√ of ∑ of ² ÷ √ of ∑ of ² ÷

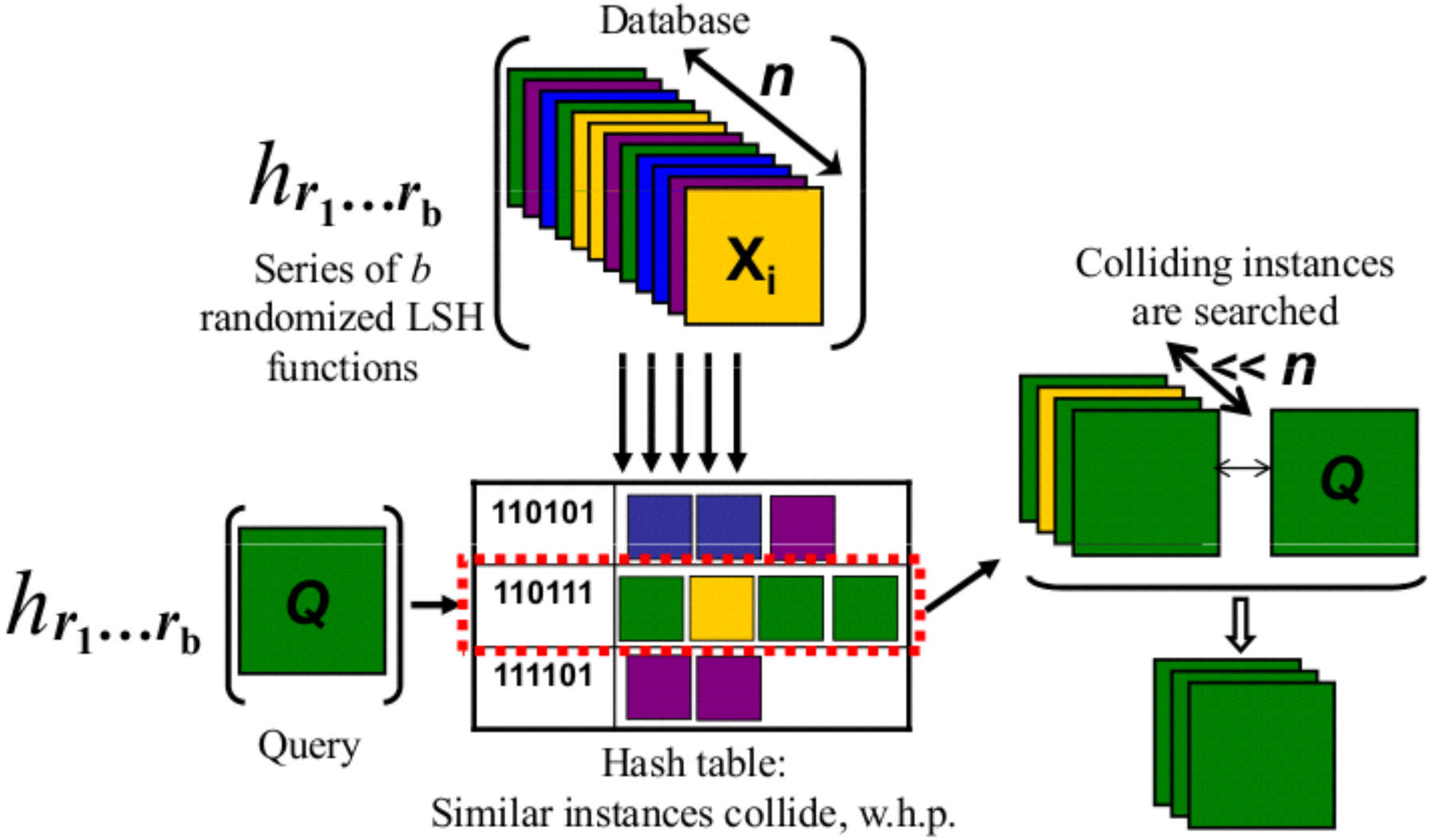
3 out of hundreds of unique words match (Jaccard < 0.03)

Example idea from: Manning et al. Introduction to Information Retrieval. 2009 free PDF!

Locality Sensitive Hashing (LSH) 1/2

- A **hashing** approach to group **near neighbors**.
- Map similar items (e.g., documents or words) into the same [hash] buckets.
- LSH "**maximizes**" (instead of minimizing) hash **collisions**.
- It therefore is a **dimensionality reduction** technique.
- For documents or words, **minhashing** can be used.
  - Approach from Rajaraman & Ullman, Mining of Massive Datasets, 2010
  - http://infolab.stanford.edu/~ullman/mmds/ch3a.pdf

Locality Sensitive Hashing (LSH) 2/2



M Vogiatzis. micvog.com 2013.

Minhash signatures (1/2)

Create a n-gram/k-shingle × document matrix (likely **very** sparse!):

{ **T**: shingle/n-gram in document
F: shingle/n-gram not in document

n-gram/shingle ID	x	D_1	D_2	D_3	D_4	h_1	h_2
	0	**T**	F	F	**T**	1	1
	1	F	F	**T**	F	2	4
	2	F	**T**	F	**T**	3	2
	3	**T**	F	**T**	**T**	4	0
	4	F	F	**T**	F	0	3

Step 1/2 - permuting the row order:

$h_1(x) = (x+1)\%n$

$h_2(x) = (3x+1)\%n$

with n=5 (rows)

hash-"permuted" row IDs

Lemma: Two docs will have the same first "true" shingle/n-gram when looking from top to bottom with a probability equal to their Jaccard (Set) Similarity.

a family of hash functions h_i

Idea: Create ~~sufficient **permutations**~~ of the row (shingle/n-gram) ordering so that the Jaccard Similarity can be approximated by comparing the number of coinciding vs. differing rows.

Minhash signatures (2/2)

shingle or n-gram ID

S	D_1	D_2	D_3	D_4	h_1	h_2
0	T	F	F	T	1	1
1	F	F	T	F	2	4
2	F	T	F	T	3	2
3	T	F	T	T	4	0
4	F	F	T	F	0	3

$h_1(x) = (x+1)\%n$

$h_2(x) = (3x+1)\%n$

from step one

n=5

Step 2/2 - generating the hash signature:

(vertical, per-document)

minhash signatures:

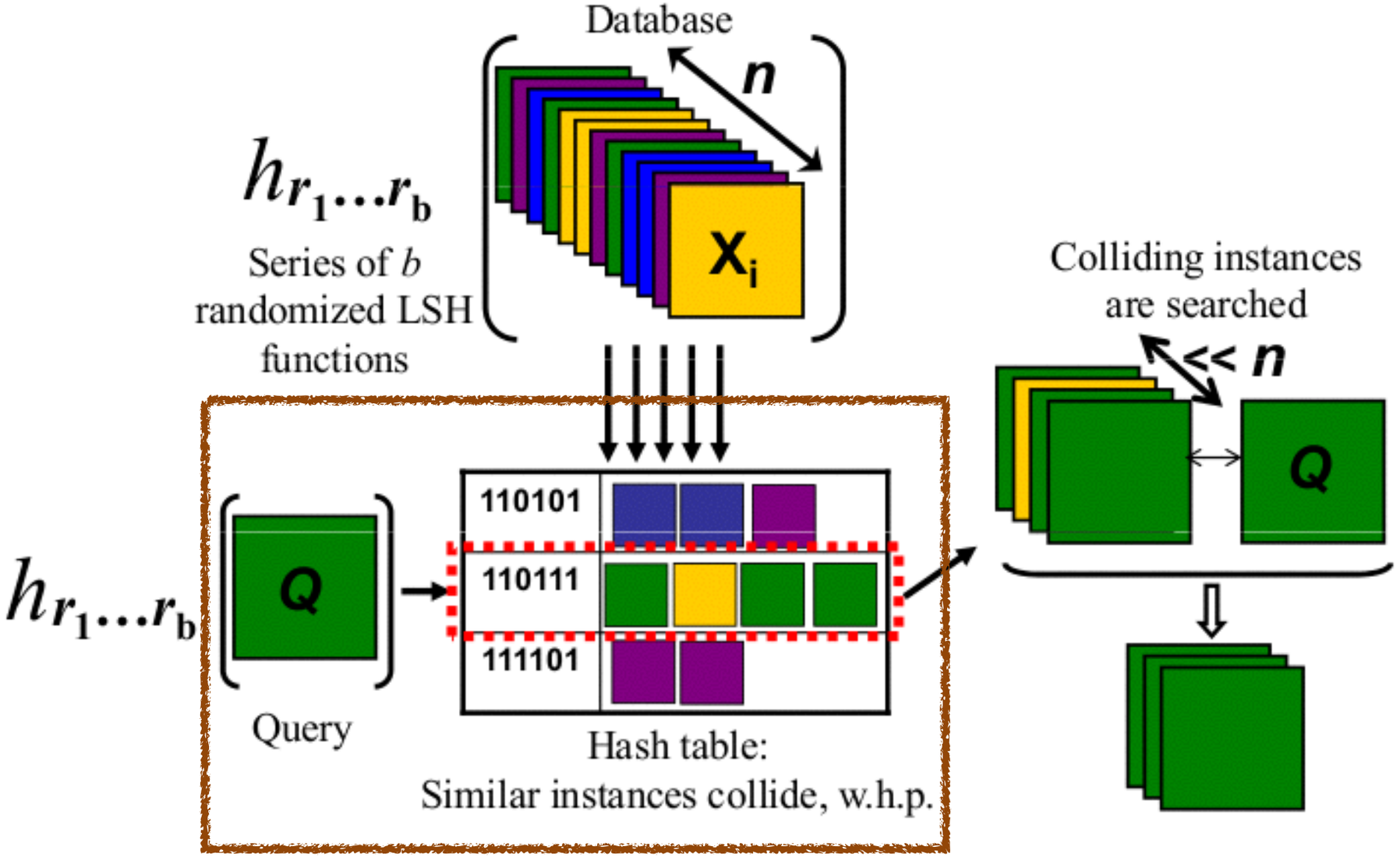
M	D_1	D_2	D_3	D_4
h_1	1	3	**0**	1
h_2	0	2	0	0

perfectly map-reduce-able and embarrassingly parallel!

```
init matrix M = ∞

for each shingle s:
 for each hash h:
  for each doc d:
   if S[d,s] and M[d,h] > h(s):
    M[d,h] = h(s)
```

Locality Sensitive Hashing (LSH) 2/2



M Vogiatzis. micvog.com 2013.

Banded locality sensitive minhashing

Bands		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	…
1	h_1	**1**	0	2	**1**	7	1	4	5	
	h_2	**1**	2	4	**1**	6	5	5	6	
	h_3	**0**	5	6	**0**	6	4	7	9	
2	h_5	4	0	8	8	**7**	6	5	**7**	
	h_1	7	7	0	8	**3**	8	7	**3**	
	h_4	8	9	0	7	**2**	4	8	**2**	
3	h_2	8	5	**4**	0	9	8	**4**	7	
	h_6	9	4	**3**	9	0	8	**3**	9	
	h_7	8	5	**8**	0	0	6	**8**	0	
…	…									

typical: r=10, b=30

Bands b $\propto$ $p_{agreement}$

Hashes/Band r $\propto 1/p_{agreement}$

(in at least one band) $p_{agreement} = \mathbf{1 - (1 - s^r)^b}$

s = Jaccard(A,B)

Union Find to connect Documents across Bands

	T_0	T_1	T_2	T_3	T_4	T_5	T_6	T_7	…
h_1	0	**1**	2	7	**1**	**1**	4	5	
h_2	2	**5**	4	6	**5**	**5**	5	6	
h_3	5	**0**	6	6	**0**	**0**	7	9	
h_5	4	0	**2**	8	8	**2**	5	5	
h_1	7	7	**1**	8	0	**1**	7	3	
h_6	8	9	**6**	7	0	**6**	8	8	
h_2	8	5	**4**	**4**	9	8	**4**	**4**	
h_5	9	4	**3**	**3**	0	1	**3**	**3**	
h_4	8	5	**8**	**8**	0	6	**8**	**8**	
…									

new "connection":
{ 2 5 }

Document clusters

{0}, {1 4 5}, {2 3 6 7} —*union (2,5)*→ {0}, {1 2 3 4 5 6 7}

POLITÉCNICA
"Ingeniamos el futuro"

CAMPUS
DE EXCELENCIA
INTERNACIONAL

Text Mining 2
Unsupervised Methods

Madrid Summer School on
Advanced Statistics and Data Mining

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

Sentence segmentation

- Sentences are **the** fundamental linguistic unit
  - Sentences are the boundaries or "constraints" for linguistic phenomena.
  - **Collocations** ["United Kingdom", "vice president"], idioms ["drop me a line"], phrases [e.g., the preposition phrase "of great fame"], clauses, statements, … all occur **within** a sentence.
- Rule/pattern-based segmentation
  - Segment sentences if the marker is followed by an upper-case letter
  - Works well for "clean text" (news articles, books, papers, …)
  - **Special cases**: abbreviations, digits, lower-case proper nouns (genes, "amnesty international", …), hyphens, quotation marks, …
- Supervised sentence boundary detection
  - Use some Markov model or a conditional random field to identify possible sentence segmentation tokens
  - Requires labeled examples (segmented sentences)

Punkt Sentence Tokenizer (PST) 1/2

- Unsupervised sentence boundary detection
- $P(\bullet|w_{-1}) > c_{cpc}$ Dr.
  - Determines if a marker ● is used as an **abbreviation** marker by comparing the **conditional probability** that the word w_{-1} before ● is followed by the marker against some (high) cutoff probability.
    - $P(\bullet|w_{-1}) = P(w_{-1}, \bullet) \div P(w_{-1})$
    - K&S set $c = 0.99$
- $P(w_{+1}|w_{-1}) > P(w_{+1})$ Mrs. Watson
  - Evaluates the likelihood that w_{-1} and w_{+1} surrounding the marker ● are more commonly collocated than would be expected by chance: ● is assumed an **abbreviation** marker ("not independent") if the LHS is greater than the RHS.
- $F_{length}(w) \times F_{markers}(w) \times F_{penalty}(w) \geq c_{abbr}$ U.S.A.
  - Evaluates if any of w's morphology (length of w w/o marker characters, number of periods inside w (e.g., ["U.S.A"]), penalized when w is not followed by a ●) makes it more likely that w is an abbreviation against some (low) cutoff.
- $F_{ortho}(w)$; $P_{sstarter}(w_{+1}|\bullet)$; … . Therefore
  - Orthography: lower-, upper-case or capitalized word after a probable ● or not
  - Sentence Starter: Probability that w is found after a ●

Punkt Sentence Tokenizer (PST) 2/2

- Unsupervised Multilingual Sentence Boundary Detection
  - Kiss & Strunk, MIT Press 2006.
  - Available from NLTK: `nltk.tokenize.punkt` (http://www.nltk.org/api/nltk.tokenize.html)
- PST is language agnostic
  - Requires that the language uses the sentence segmentation marker as an abbreviation marker
  - Otherwise, the problem PST solves is not present
- PST factors in word length
  - Abbreviations are relatively shorter than regular words
- PST takes "internal" markers into account
  - E.g., "U.S.A"
- Main weakness: long lists of abbreviations
  - E.g., author lists in citations
  - Can be fixed with a pattern-based post-processing strategy
- NB: a marker must be present
  - E.g., chats or fora

From syntactic to semantic similarity

Cosine Similarity, χ^2, Spearman's ρ, LSH, etc. all compare equal tokens.

But what if you are talking about "automobiles" and I am lazy, calling it a "car"?

We can solve this with Latent Semantic Indexing!

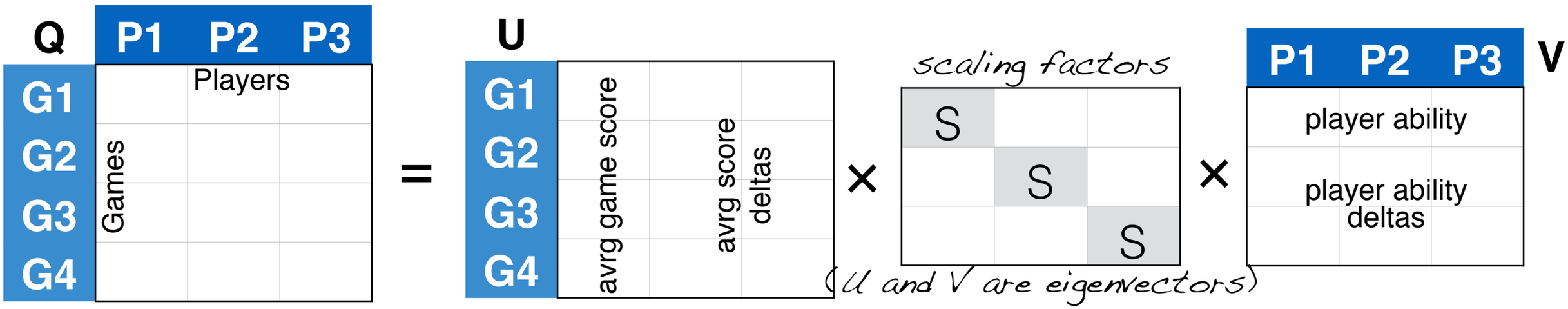
Latent Semantic Analysis (LSI 1/3)

- a.k.a. Latent Semantic **Indexing** (in Text Mining): **feature extraction** for **semantic inference**
- Linear algebra background
  - Singular value decomposition of a matrix Q: $\mathbf{Q} = \mathbf{U\Sigma V^T}$

    orthonormal factors of Q (QQ^T and Q^TQ)

    singular values: scaling factor

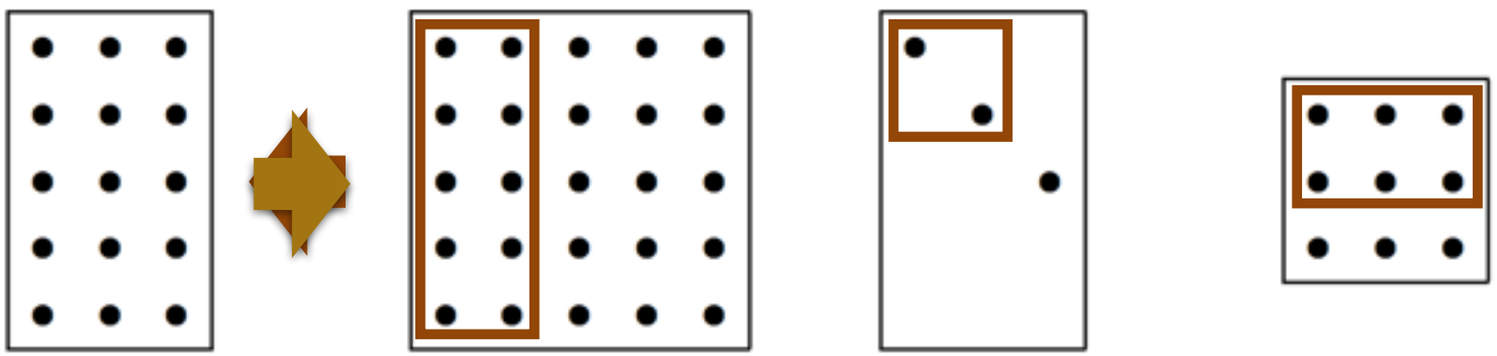
    the factors "predict" Q in terms of similarity (Frobenius norm) using as many factors as the lower dimension of Q



- SVD in text mining
  - Inverted index = doc. eigenvectors × singular values × term eigenvectors

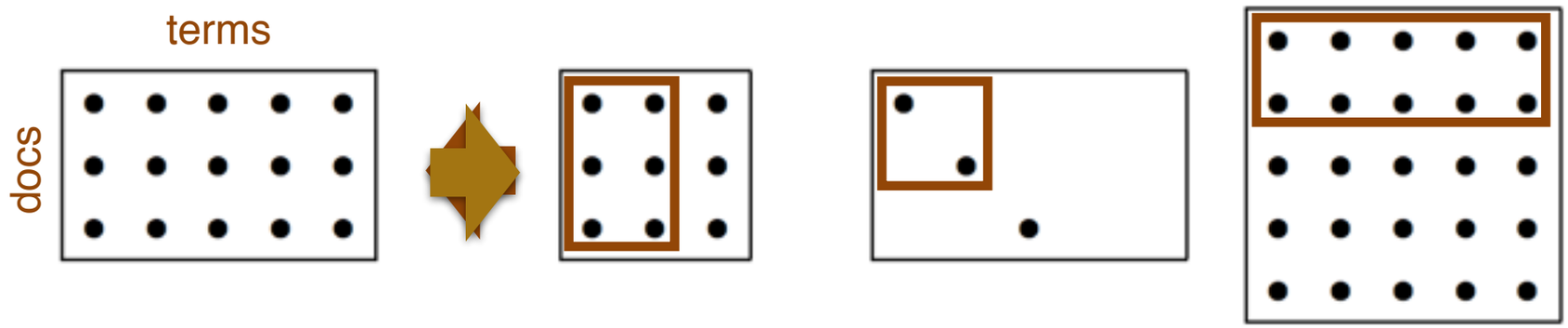
Latent Semantic Analysis (LSI 2/3)

C = $\hat{F}$eat. extraction by selecting only the largest n eigenvalues



- Inverted index = doc. eigenvectors × singular values × term eigenvectors

$$C = U \Sigma V^T$$



- Image taken from: Manning et al. An Introduction to IR. 2009

Latent Semantic Analysis (LSI 3/3)

c1: *Human* machine *interface* for ABC *computer* applications
c2: A *survey* of *user* opinion of *computer system response time*
c3: The *EPS user interface* management *system*
c4: *System* and *human system* engineering testing of *EPS*
c5: Relation of *user* perceived *response time* to error measurement

m1: The generation of random, binary, ordered *trees*
m2: The intersection *graph* of paths in *trees*
m3: *Graph minors* IV: Widths of *trees* and well-quasi-ordering
m4: *Graph minors*: A *survey*

C

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
computer	1	1	0	0	0	0	0	0	0
user	0	1	1	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
time	0	1	0	0	1	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1

[Spearman's] rho(human, user) = -0.38
rho(human, minors) = -0.29

$\hat{C}$

	c1	c2	c3	c4	c5	m1	m2	m3	m4
human	0.16	0.40	0.38	0.47	0.18	-0.05	-0.12	-0.16	-0.09
interface	0.14	0.37	0.33	0.40	0.16	-0.03	-0.07	-0.10	-0.04
computer	0.15	0.51	0.36	0.41	0.24	0.02	0.06	0.09	0.12
user	0.26	0.84	0.61	0.70	0.39	0.03	0.08	0.12	0.19
system	0.45	1.23	1.05	1.27	0.56	-0.07	-0.15	-0.21	-0.05
response	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
time	0.16	0.58	0.38	0.42	0.28	0.06	0.13	0.19	0.22
EPS	0.22	0.55	0.51	0.63	0.24	-0.07	-0.14	-0.20	-0.11
survey	0.10	0.53	0.23	0.21	0.27	0.14	0.31	0.44	0.42
trees	-0.06	0.23	-0.14	-0.27	0.14	0.24	0.55	0.77	0.66
graph	-0.06	0.34	-0.15	-0.30	0.20	0.31	0.69	0.98	0.85
minors	-0.04	0.25	-0.10	-0.21	0.15	0.22	0.50	0.71	0.62

top 2 dim

test # dim to use via synonyms or missing words

From: Landauer et al. An Introduction to LSA. 1998

rho(human, user) = **0.94**
rho(human, minors) = **-0.83**

Principal Component vs. Latent Semantic Analysis

best Frobenius norm: minimize “std. dev.” of matrix
best affine subspace: minimize dimensions while maintaing the form

- **LSA** seeks for the **best linear subspace** in **Frobenius norm**, while **PCA** aims for the **best affine linear subspace**.
- **LSA** (**can**) **use** TF-IDF weighting as **preprocessing** step.
- **PCA requires the** (square) **covariance matrix** of the original matrix as its first step and therefore can only compute term-term or doc-doc similarities.
- **PCA matrices are more dense** (zeros occur only when true independence is detected).

Text Summarization

Russian defense minister Ivanov called on Sunday for the creation of a global front for combating terrorism.

- Extractive summarization
  - Select the most informative sentences.
  - Order sentences (or leave in order).

Ivanov called for a global front combating terrorism.

- Abstractive summarization
  - Generate new text given the input document.
  - Very unique but still rather experimental.

Russia calls for a joint effort against terrorism.

Extractive Summarization with TextRank

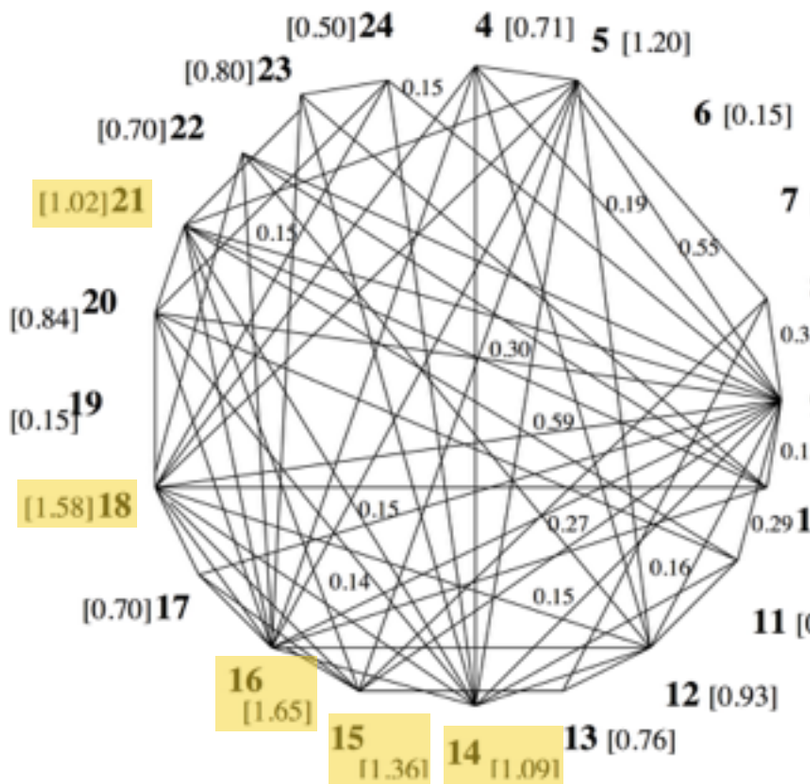
tokens, n-grams or whole sentences

1. Collect all text shingles (→ graph vertices) from the input document[s].

2. Quantify relation strength (→ edges) between those shingles from their context (**co-occurrence**) or content (**TF-IDF**).

3. Iterate a graph ranking algorithm (**PageRank**) to convergence.

4. Sort the vertices on their final score to identify the most informative shingles.



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

TextRank Summarization with Okapi-BM25 Ranking

2. Quantify relation strength (→ edges) between those shingles from their content.

"classical" TF-IDF

$$TFIDF(D_n, Q) = \sum_i^{|Q|} TF(q_i, D_n) \times IDF(q_i)$$

$$TF(q_i, D_n) = log(|q_i \in D_n|)$$

Okapi BM25 "TF modification"

$$BM25(D_n, Q) = \sum_i^{|Q|} Okapi(q_i, D_n) \times IDF(q_i)$$

$$Okapi(q_i, D_n) = \frac{TF(q_i, D_n)(k+1)}{TF(q_i, D_n) + k(1 - b + b\frac{|D_n|}{mean(|D|)})}$$

Main difference: the Okapi function flattens out much faster than a log-scaled Term Frequency function (alone).

https://en.wikipedia.org/wiki/Okapi_BM25

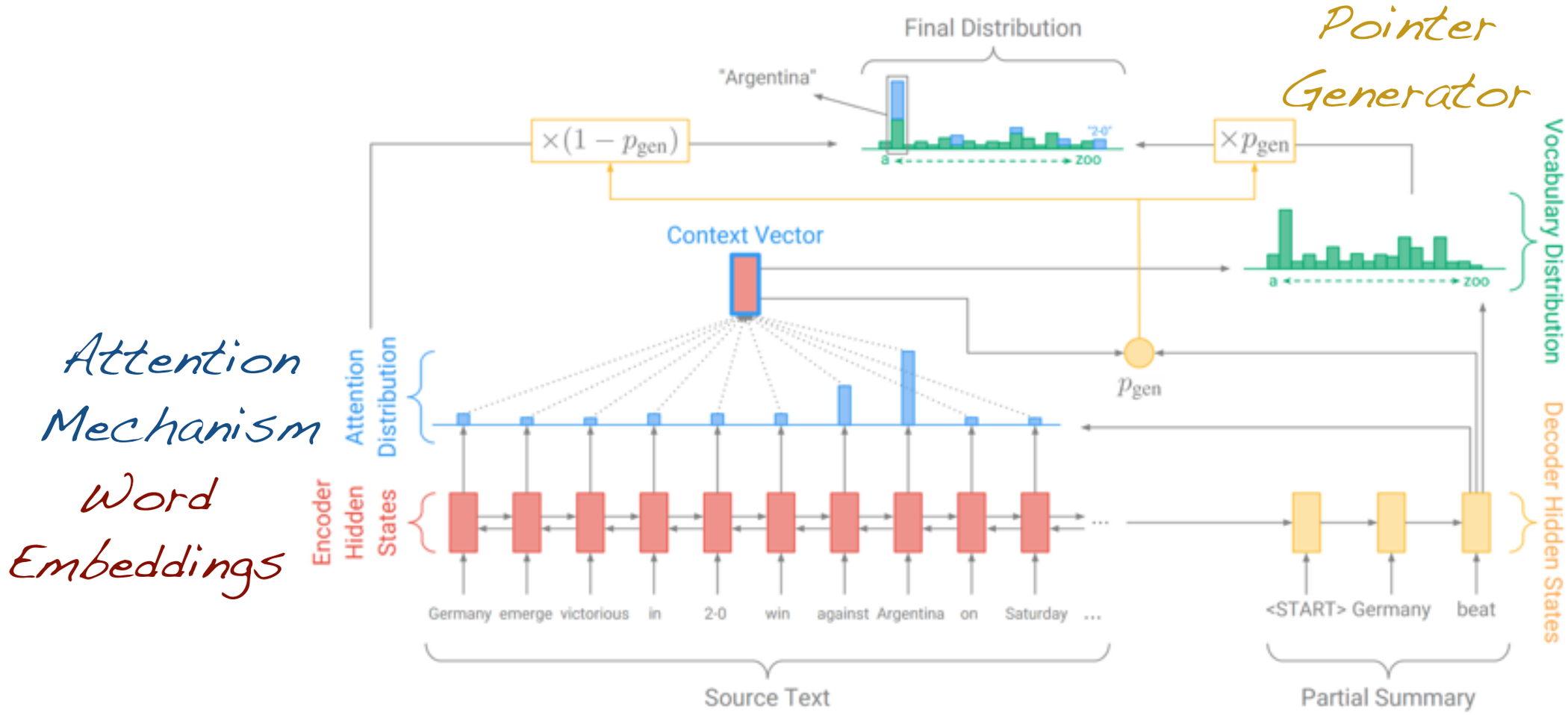
Barrios, F., López, F., Argerich, L., and Wachenchauzer, R. (2016).
Variations of the similarity function of TextRank for automated summarization.

LexRank vs. TextRank

- Published simultaneously in 2004 by two independent groups
- Both are based on the same idea (graph similarity ranking)
- **LexRank** is part of a larger supervised summarization system ("MEAD") that uses features like sentence position and length.
- **LexRank** additionally covered a multi-document summarization approach (requiring post-processing; "CSIS")
- The **TextRank** authors expanded their work to keyword extraction

Erkan, G., and Radev, D.R. (2004). LexRank: Graph-based Lexical Centrality as Salience in Text Summarization.

Abstractive Summarization with Recurrent Neural Networks



Generates new text using the full sentence context (attention mechanism) from the current text (word embeddings), while at the same time it can copy facts/words (pointer generator) over to the new text.

See, A., Liu, P.J., and Manning, C.D. (2017). Get To The Point: Summarization with Pointer-Generator Networks.

A first look at probabilistic graphical models

- Latent Dirichlet Allocation: LDA
  - Blei, Ng, and Jordan. Journal of Machine Learning Research 2003
  - For assigning "topics" to "documents" i.e., for text classification
  - An **unsupervised**, **generative** model

Latent Dirichlet Allocation (LDA 1/3)

- Intuition for LDA
  - From: Edwin Chen. Introduction to LDA. 2011
  - "Document Collection"

• I like to eat broccoli and bananas. • I ate a banana and spinach smoothie for breakfast.	➡ Topic A
• Chinchillas and kittens are cute. • My sister adopted a kitten yesterday.	➡ Topic B
• Look at this cute hamster munching on a piece of broccoli.	➡ Topic 0.6A + 0.4B

Topic A: 30% broccoli, 15% bananas, 10% breakfast, 10% munching, …

Topic B: 20% chinchillas, 20% kittens, 20% cute, 15% hamster, …

The Dirichlet process

A Dirichlet process is like drawing from an (infinite) "bag of dice" (with finite faces).

- A Dirichlet is a [possibly continuos] **distribution over** [discrete/multinomial] **distributions** (probability **masses**).

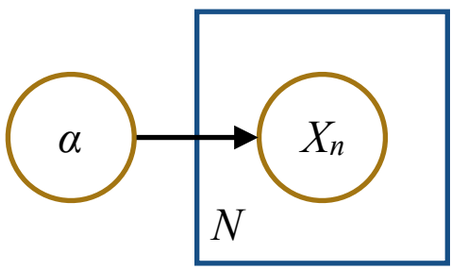
Gamma function -> a "continuous" factorial [!]

$$D(\boldsymbol{\theta}, \boldsymbol{\alpha}) = \frac{\Gamma(\sum \alpha_i)}{\prod \Gamma(\alpha_i)} \prod \theta_i^{\alpha_i - 1}$$

$\boldsymbol{\alpha}$ Dirichlet prior: $\forall\ \alpha_i \in \boldsymbol{\alpha}: \alpha_i > 0$

$\sum \theta_i = 1$; a Probability Mass Function

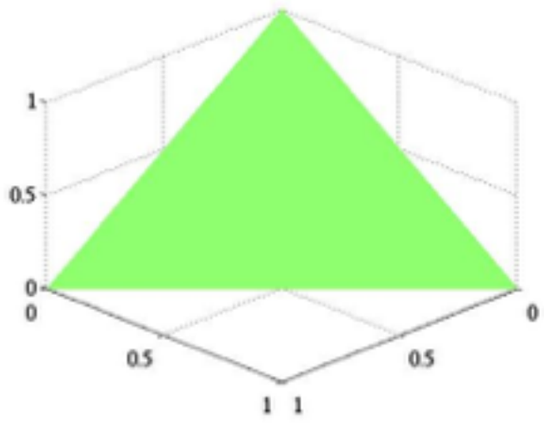
- The **Dirichlet Process samples** multiple independent, discrete **distributions** θ_i with repetition from $\boldsymbol{\theta}$ ("statistical clustering").



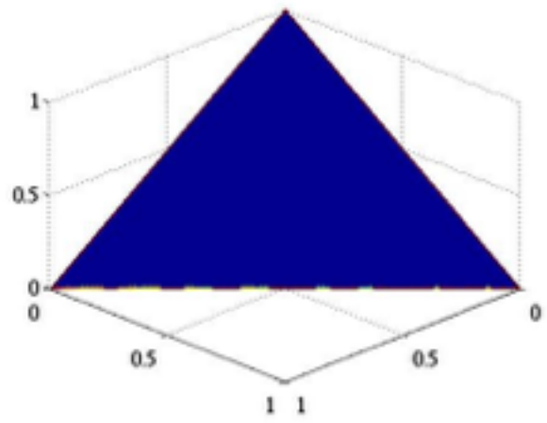
1. Draw a new distribution X from $D(\boldsymbol{\theta}, \boldsymbol{\alpha})$
2. With probability $\alpha \div (\alpha + n - 1)$ draw a new X
   With probability $n \div (\alpha + n - 1)$, (re-)sample an X_i from X

The Dirichlet prior α

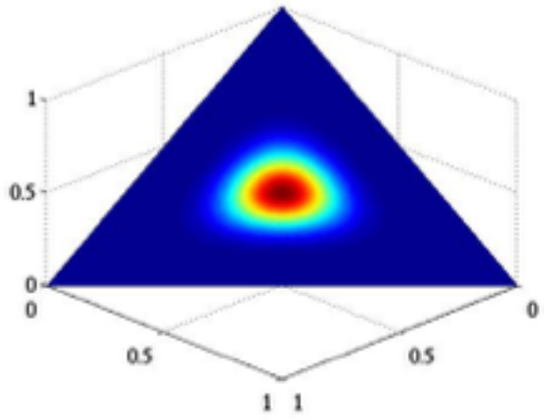
"density plots over the probability simplex in R3"



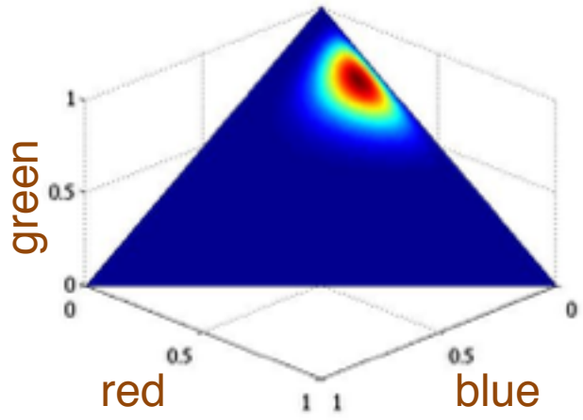
$\alpha = [1, 1, 1]$



$\alpha = [.1, .1, .1]$

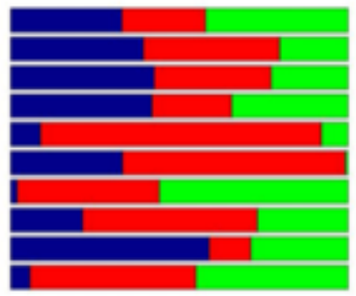


$\alpha = [10, 10, 10]$

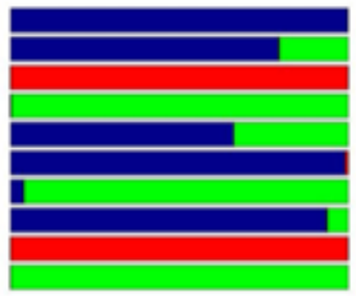


$\alpha = [2, 5, 15]$

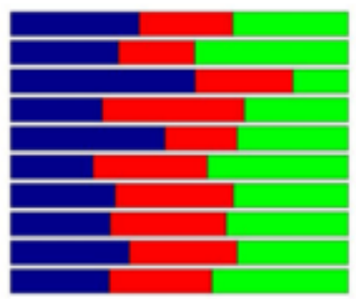
Documents and topic distributions (N=3)



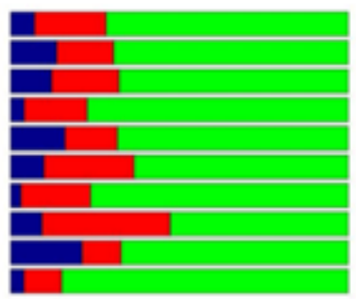
$\alpha = (1, 1, 1)$



$\alpha = (0.1, 0.1, 0.1)$



$\alpha = (10, 10, 10)$



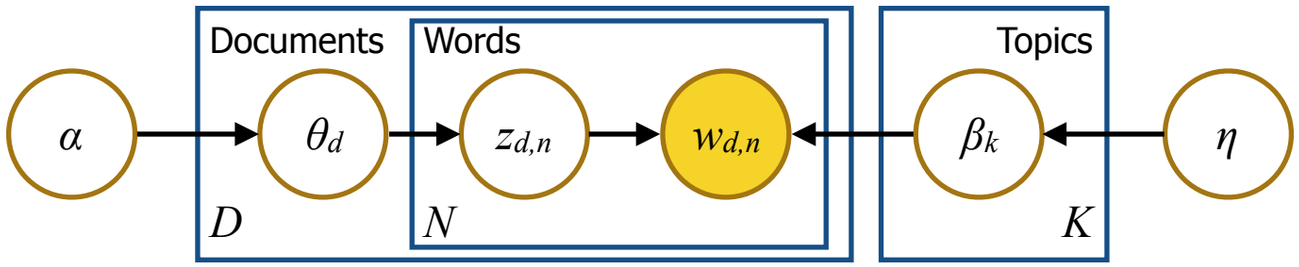
$\alpha = (2, 5, 15)$

→ equal, =1 ➡ uniform distribution
→ equal, <1 ➡ marginal distrib. (”choose few“)
→ equal, >1 ➡ symmetric, mono-modal distrib.
→ not equal, >1 ➡ non-symmetric distribution

Frigyik et al. Introduction to the Dirichlet Distribution and Related Processes. 2010

Latent Dirichlet Allocation (LDA 2/3)

A Document-Topic is the assignment of a Document to some Topic.
A Word-Topic is the assignment of a (non-unique!) Word (in a Document) to some Topic



Joint Probability

P(Document-T.)

P(Word | Topics, Word-T.)

$$P(B, \Theta, Z, W) = \left(\prod_k^K P(\beta_k|\eta)\right)\left(\prod_d^D P(\theta_d|\alpha)\prod_n^N P(z_{d,n}|\theta_d)P(w_{d,n}|\beta_{1:K}, z_{d,n})\right)$$

P(Topic)

P(Word-T. | Document-T.)

- α - per-document Dirichlet prior
- θ_d - topic distribution of document d
- $z_{d,n}$ - word-topic assignments
- $w_{d,n}$ - **observed** words
- β_k - word distrib. of topic k
- η - per-topic Dirichlet prior

dampens the topic-specific score of terms assigned to many topics

$$termscore_{k,n} = \hat{\beta}_{k,n} log \frac{\hat{\beta}_{k,n}}{\left(\prod_j^K \hat{\beta}_{j,n}\right)^{1/K}}$$

What Topics is a Word assigned to?

Latent Dirichlet Allocation (LDA 3/3)

- LDA sampling/inference in a nutshell
  - Initialization: Choose K, the number of Topics, and randomly assign one out of the K Topics to each of the N Words in each of the D Documents.
    - The **same word** can have different Topics **at different positions** in the Document.
  - Calculate the posterior probability that Topic t generated Word w.
    - Then, for each Topic and for each Word in each Document:

1. Compute P(Word-Topic | Document): the proportion of [Words assigned to] Topic t in Document d
2. Compute P(Word | Topics, Word-Topic): the probability a Word w is assigned a Topic t (using the general distribution of Topics and the Document-specific distribution of [Word-] Topics)
   - Note that a Word can be assigned a different Topic each time it appears in a Document.
3. Given the prior probabilities of a Document's Topics and that of Topics in general, reassign P(Topic | Word) = P(Word-Topic | Document) * P(Word | Topics, Word-Topic)

- Repeat until P(Topic | Word) stabilizes (e.g., Collapsed Gibbs sampling)
- Better: Use collapsed **variational inference** (i.e., combining Variational Bayes)

Teh, Newman, Welling (2006). A Collapsed Variational Bayes Inference Algorithm for LDA

Text Mining 3
Representation Learning

Madrid Summer School on
Advanced Statistics and Data Mining

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

Representation learning

- a **transformation of raw data** to a representation that can be effectively exploited in machine learning tasks
- **obviates feature engineering** (manually developing a representation to use for the classifier)
- many feature learning techniques do not required labeled data (i.e., are fully **unsupervised**)

Word representations

A trivial approach is to use a token's string itself as the representation; Numerically encode that leads us to a sparse, **one-hot** vector:

$$\text{"}tutorial\text{"} := [0\ 0\ 0\ 0\ \ldots 0\ 0\ 0\ 0\ \mathbf{1}\ 0\ 0\ 0\ 0\ \ldots 0\ 0\ 0]$$

Problem: every such vector v is orthogonal to all others, so:

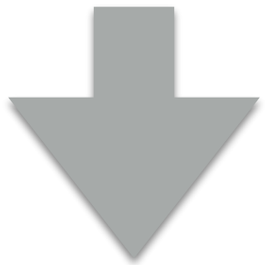
$$\mathbf{v}_1^T \cdot \mathbf{v}_2 = \mathbf{0}$$

In other words, there is no notion of similarity between those vectors.

Therefore, the goal of word representations is to [numerically] **quantify the similarity of related words**.

From one-hot encoding to word embeddings

```
fun   = [1.0, 0.0, …, 0.0, 0.0, …, 0.0]
enjoy = [0.0, 0.0, …, 1.0, 0.0, …, 0.0]
like  = [0.0, 0.0, …, 0.0, 0.0, …, 1.0]
```

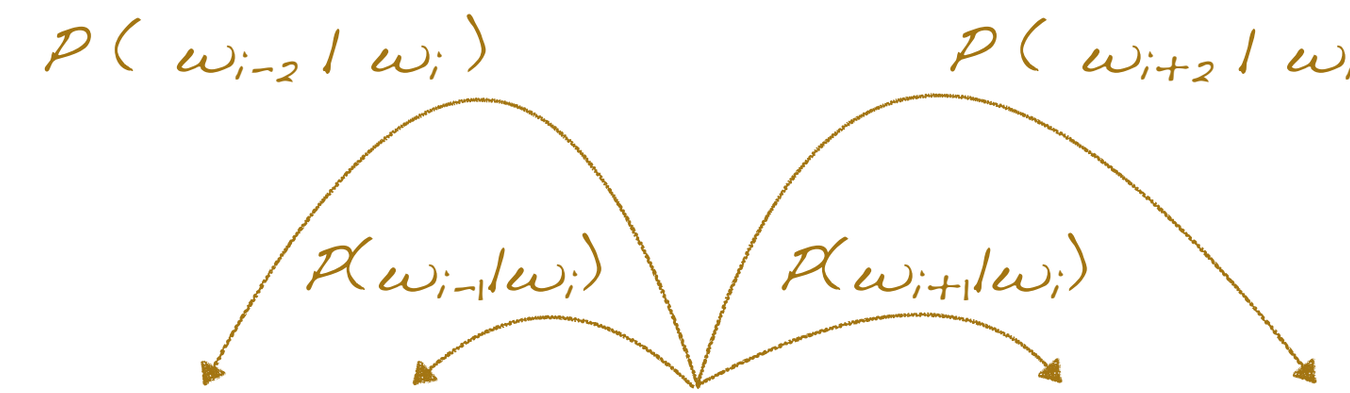


```
fun   = [0.6, 0.0, …, 0.3, 0.0, …, 0.1]
enjoy = [0.4, 0.0, …, 0.5, 0.0, …, 0.1]
like  = [0.2, 0.0, …, 0.2, 0.0, …, 0.6]
```

"You shall know a word by the company it keeps"

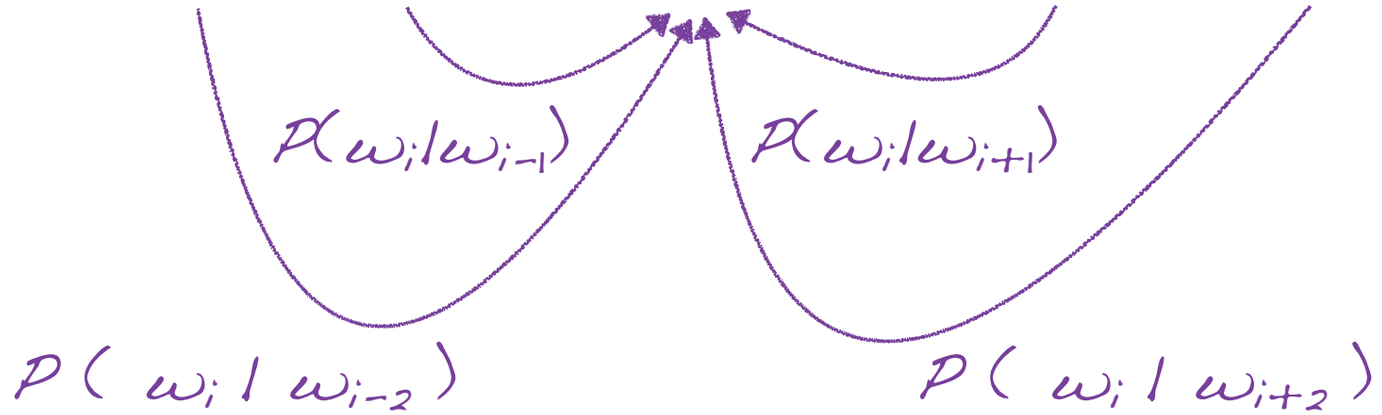
- J. R. Firth, **1957**:11
  (so the idea of word embeddings is definitely not new...)
- That is, the **context** (the surrounding words) of a word is dependent on the word itself; Put it slightly differently: a word "dictates" the possible words you can find in its surrounding.

Predicting the surrounding of words (and vice versa)



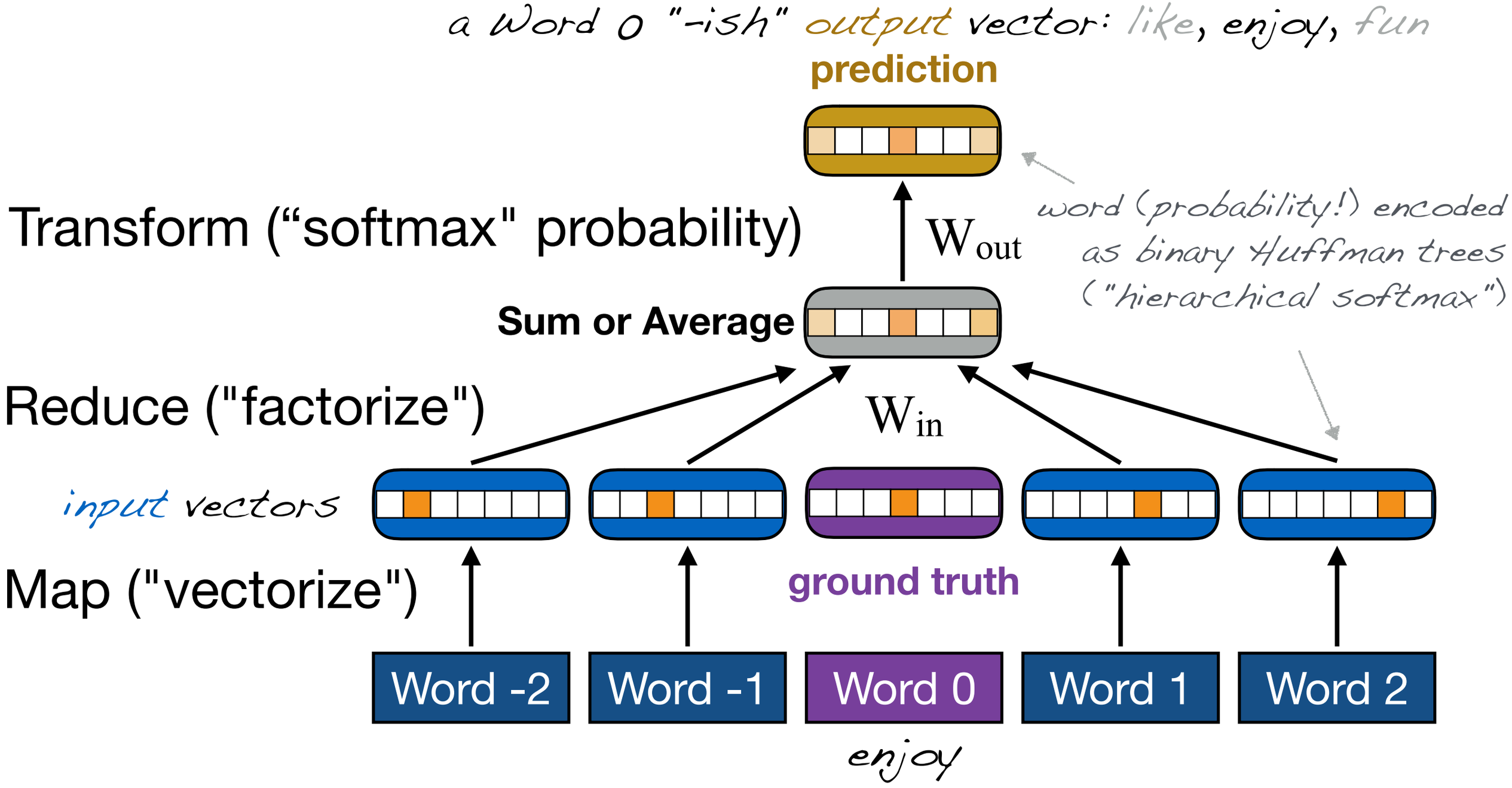
(skip-gram)

... today we learned something new ...



(CBOW)

Word embeddings with neural networks (CBOW model)



Where are the final word embeddings (vectors)?

$$\text{Embeddings} := \text{ll}\ W_{\text{in}}\ \text{ll}_2$$

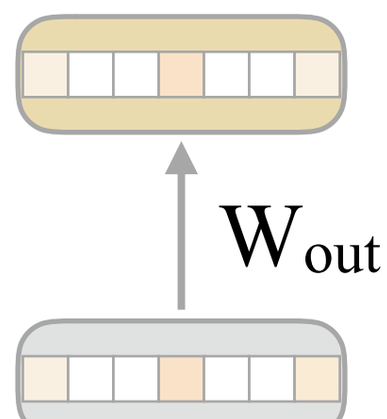
(Mikolov 2013, NAACL-HLT, word2vec)

$$\text{Embeddings} := \text{ll}\ W_{\text{in}} + W_{\text{out}}\ \text{ll}_2$$

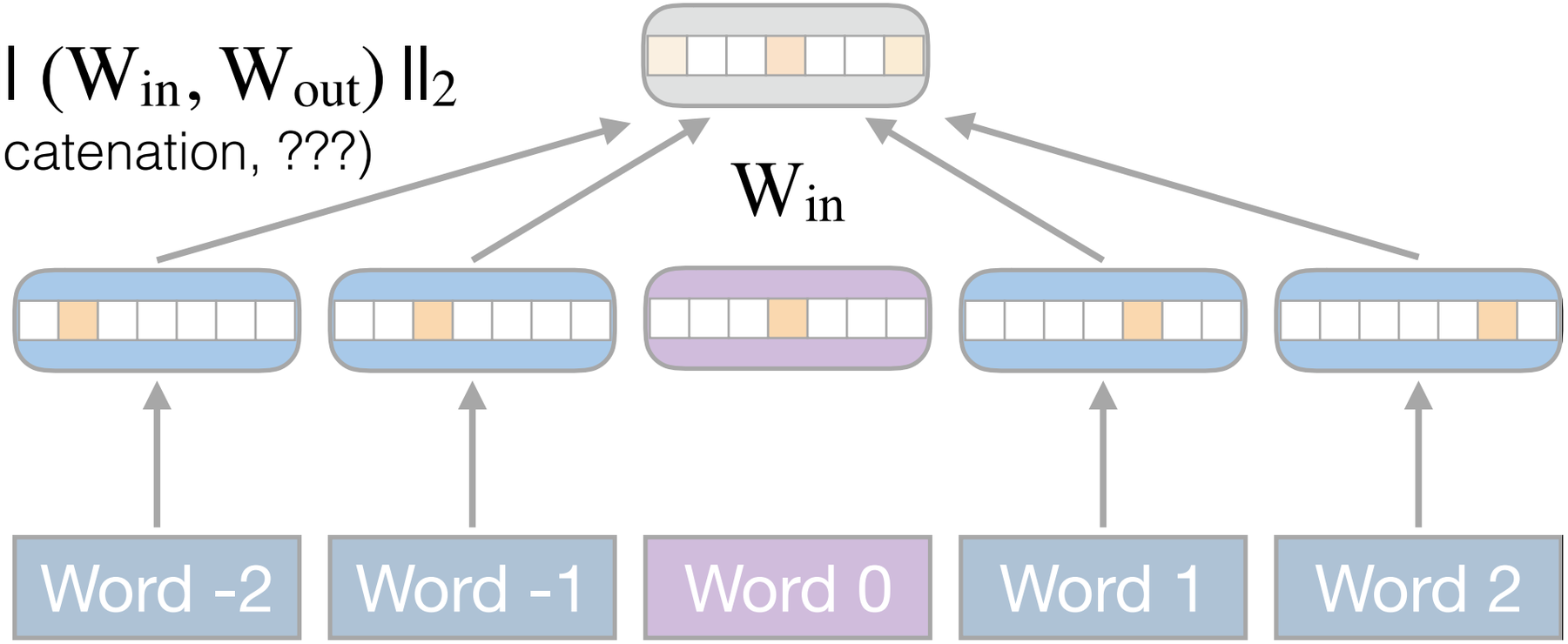
(Pennington 2014, EMNLP, Glove)

$$\text{Embeddings} := \text{ll}\ (W_{\text{in}}, W_{\text{out}})\ \text{ll}_2$$

(in+out vector concatenation, ???)



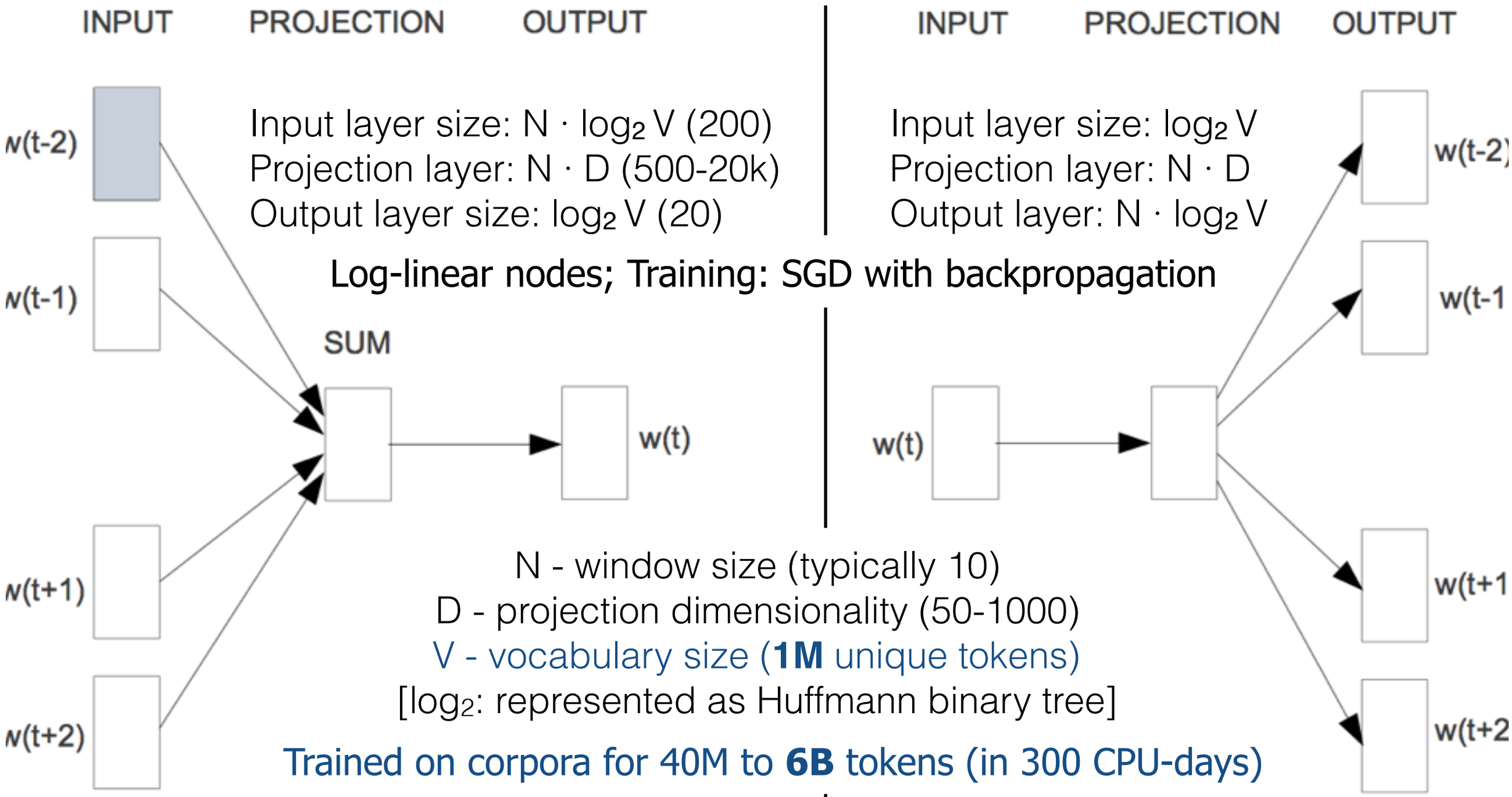
$W \rightarrow$ n. words x n. hidden units



NB: softmax CBOW is easier to understand, but skip-gram with negative subsampling (SGNS) performs better

Neural network models of language

word2vec - Thomas Mikolov et al. - Google - 2013



Input layer size: $N \cdot \log_2 V$ (200)
Projection layer: $N \cdot D$ (500-20k)
Output layer size: $\log_2 V$ (20)

Input layer size: $\log_2 V$
Projection layer: $N \cdot D$
Output layer: $N \cdot \log_2 V$

Log-linear nodes; Training: SGD with backpropagation

N - window size (typically 10)
D - projection dimensionality (50-1000)
V - vocabulary size (**1M** unique tokens)
[$\log_2$: represented as Huffmann binary tree]

Trained on corpora for 40M to **6B** tokens (in 300 CPU-days)

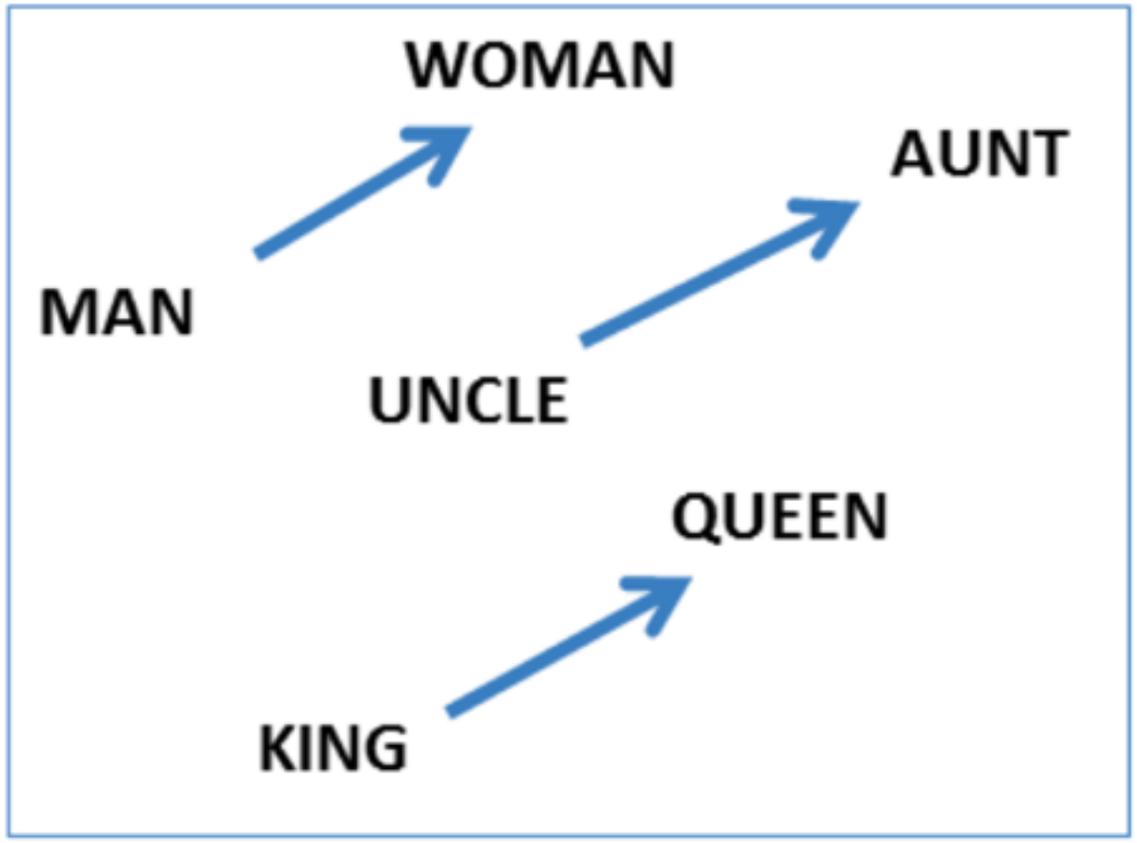
CBOW

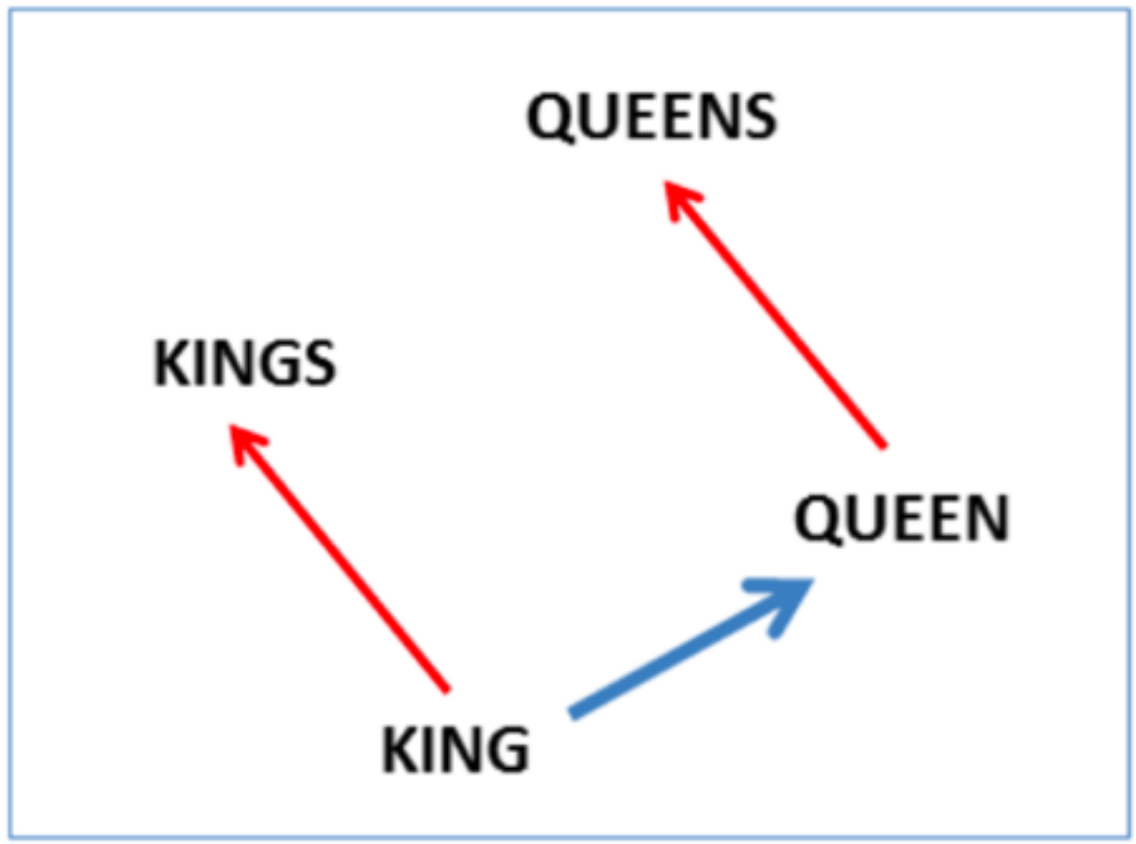
predict a word from its surrounding

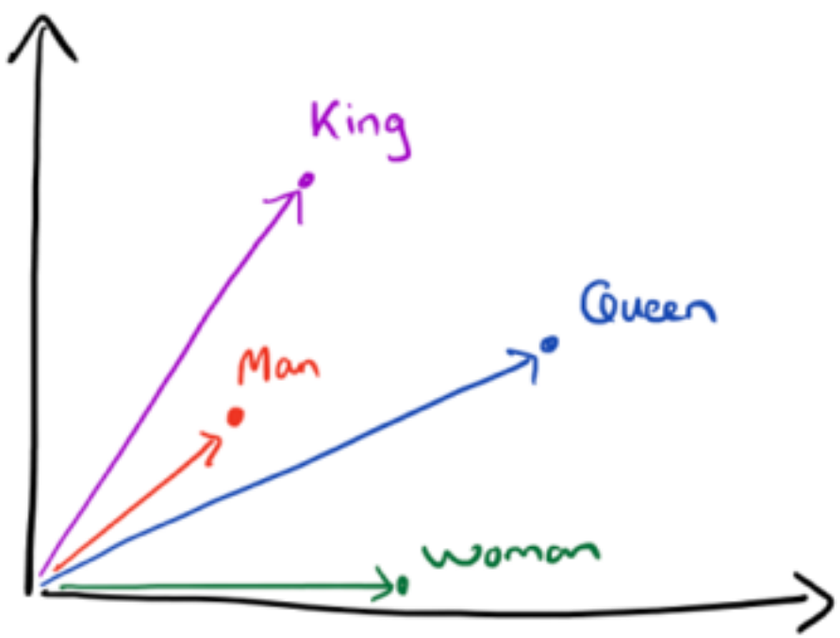
Skip-gram

predict the surrounding of a word

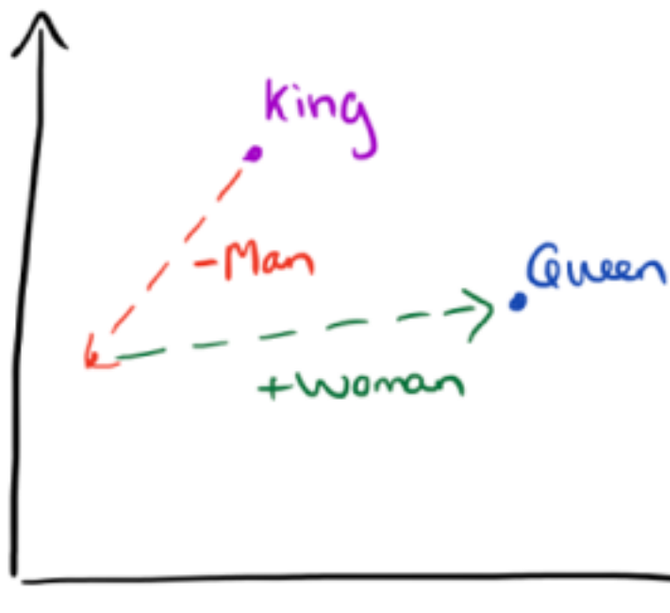
King - Man + Woman = ?





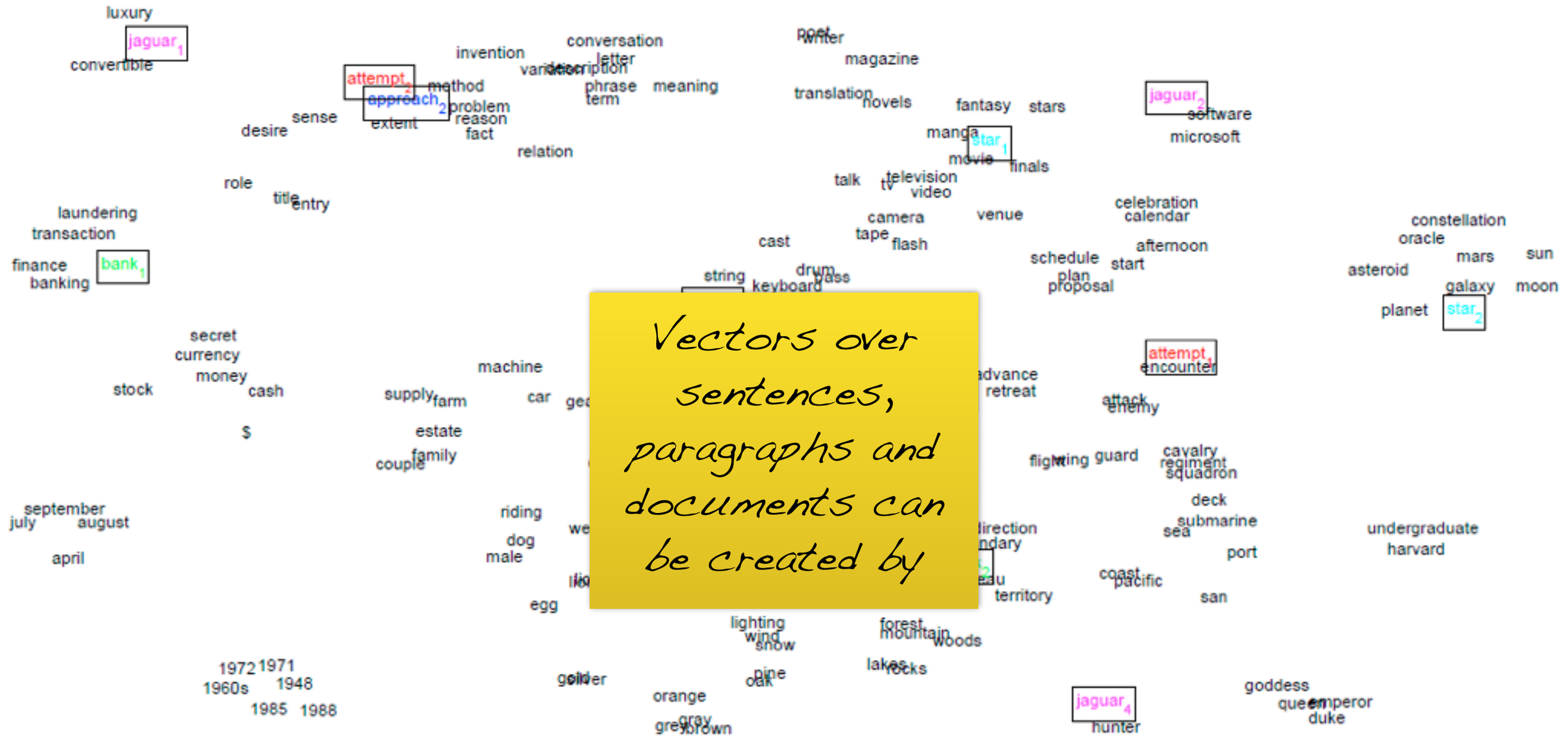


Word Vectors



Vector Composition

word2vec: co-occurrence probs.
GloVe: ratio of co-occ. probabilities

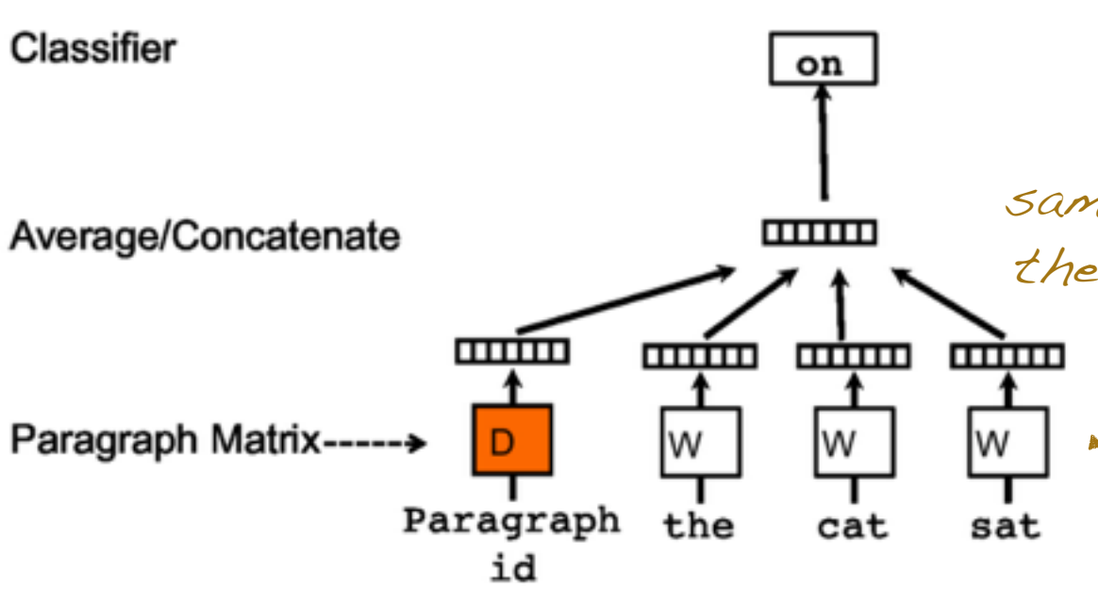


Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. ICLR Workshop.

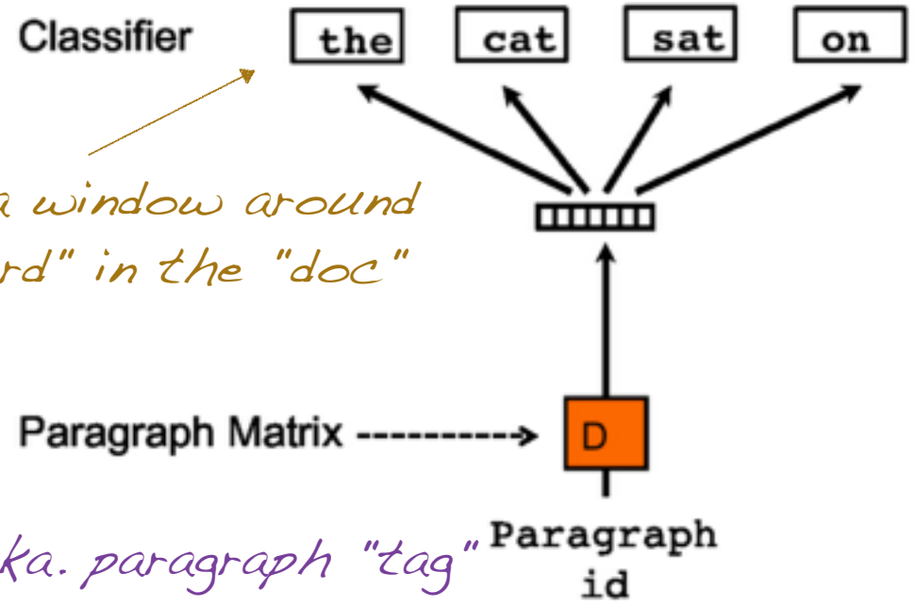
Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. In Proceedings of the Empiricial Methods in Natural Language Processing (pp. 1532–1543).

Text embeddings with Paragraph Vectors (doc2vec)

a "doc" is some piece of text: a sentence, a tweet, a paragraph, or even a whole document



sampled from a window around the "target word" in the "doc"



aka. paragraph "tag"

PV-Distributed Memory (DM)
Predict the target word from the paragraph vector and the doc's words (from a window over the doc centered at the target word).

Distributed BOW-PV
Predict the doc's words (from some window over the doc) from the paragraph vector.

Le's & Mikolov's recommendation: train both models (using concatenation) and combine them.

Le, Q., and Mikolov, T. (2014). Distributed Representations of Sentences and Documents. 2014

Paragraph Vectors (doc2vec)

- Base idea is the same as word embeddings
  - c.f. CBOW/SGNS models
- But the **paragraph vector** D **needs to be inferred** when using this model ("in production")
  - i.e., you **predict** the embedding
  - c.f. **looking up** the embedding vector for words
- D is a **tag** for each doc
  - used as memory for that doc during training
  - typically just a unique integer per doc

Out-of-vocabulary (OOV) words: character n-grams

Problem: no embedding for words not seen during training

Solution: instead learn the embeddings of a word's n-grams

split each word into its **character** n-grams (typically, $\mathrm{n} = [3, 6]$; and just use the word "as is" for tokens with character lengths < 4)

learn to embed the n-grams, with the target embedding being the average over the predicted n-gram embeddings

fastText: Joulin et al., 2016, arXiv (Facebook)

Cheap Solution: bucket all words into a fixed-size hash-table (smaller than the actual vocabulary) and allow for collisions (also known as the "**hashing trick**")

Statistical models of language and polysemy

- Polysemous words have multiple meanings (e.g., “bank”).
  - This is a real problem in scientific texts because polysemy is frequent.
- One idea: Create **context vectors** for each sense of a word (vector).
  - MSSG - Neelakantan et al. - 2015
- Caveat: Performance isn’t much better than for the skip-gram model by Mikolov et al., while training is ~5x slower.
- Simpler approach (partial solution only): use collocations
  - Either train the embeddings over the merged collocations (tomorrow's lesson), or [also] use bigrams as your embedding inputs (vs. of the [unigram] tokens)

Word embeddings: Applications in TM & NLP

- Opinion mining (Maas et al., 2011)
- Paraphrase detection (Socher et al., 2011)
- Chunking (Turian et al., 2010; Dhillon and Ungar, 2011)
- Named entity recognition (Neelakantan and Collins, 2014; Passos et al., 2014; Turian et al., 2010)
- Dependency parsing (Bansal et al., 2014)

POLITÉCNICA
"Ingeniamos el futuro"

CAMPUS
DE EXCELENCIA
INTERNACIONAL

Text Mining 4
Information Extraction

Madrid Summer School on
Advanced Statistics and Data Mining

Florian Leitner
Data Catalytics, S.L.
leitner@datacatytics.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, its 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 σ^2)
  - has problems with more frequent words and is typically only used to **rank** collocations (rather than detect them)
- **χ^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$_1$ word$_2$", with counts c and total words N:

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

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

- Asymptotically χ^2-distributed (explaining the "-2" multiplier)
- Can use the χ^2 statistic to find the critical values for λ
  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

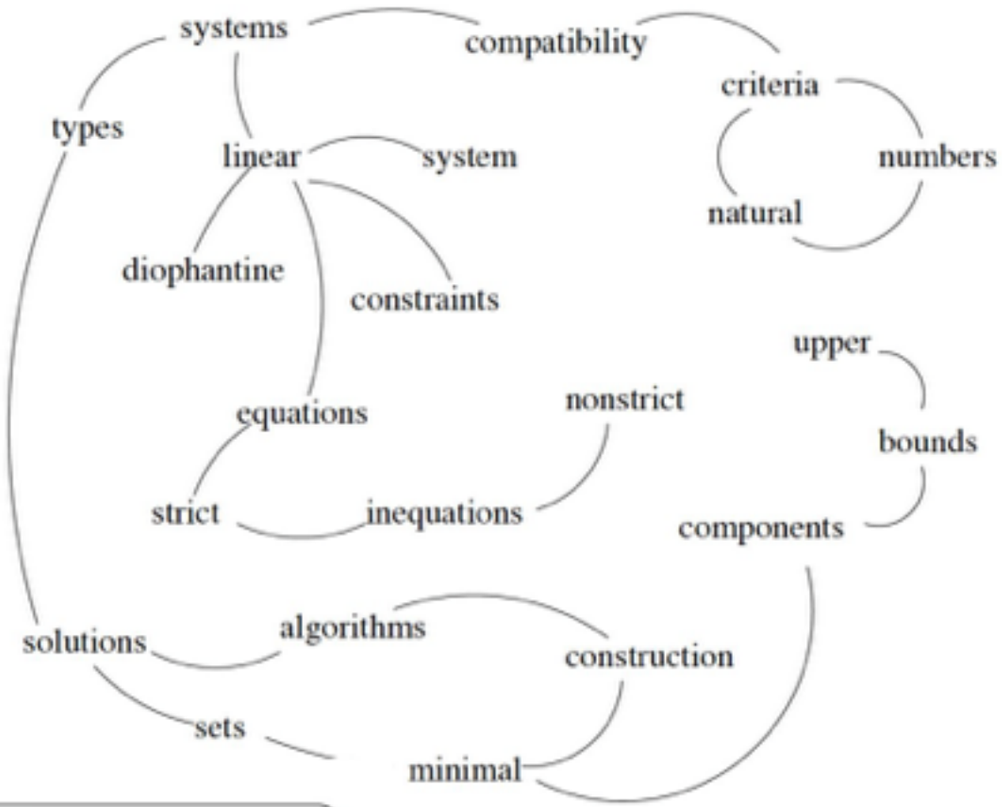
- Select only terms that are valid noun and verb phrases
  - NB: not all collocations are noun phrases, e.g. "to make up".
  - Advantage: Can be applied after any statistical selection method.
  - Disadvantage: Requires tagging tokens with their part-of-speech.
    - i.e., this is a fully supervised 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 $w_1, \ldots, w_n$
- Probability of **next word** w_n in a sentence is: $P(w_n | w_1, \ldots, w_{n-1})$
  - a **conditional probability**
- The probability of the **whole sentence** is: $P(W) = P(w_1, \ldots, w_n)$
  - 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…" ➡

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

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

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

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

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

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

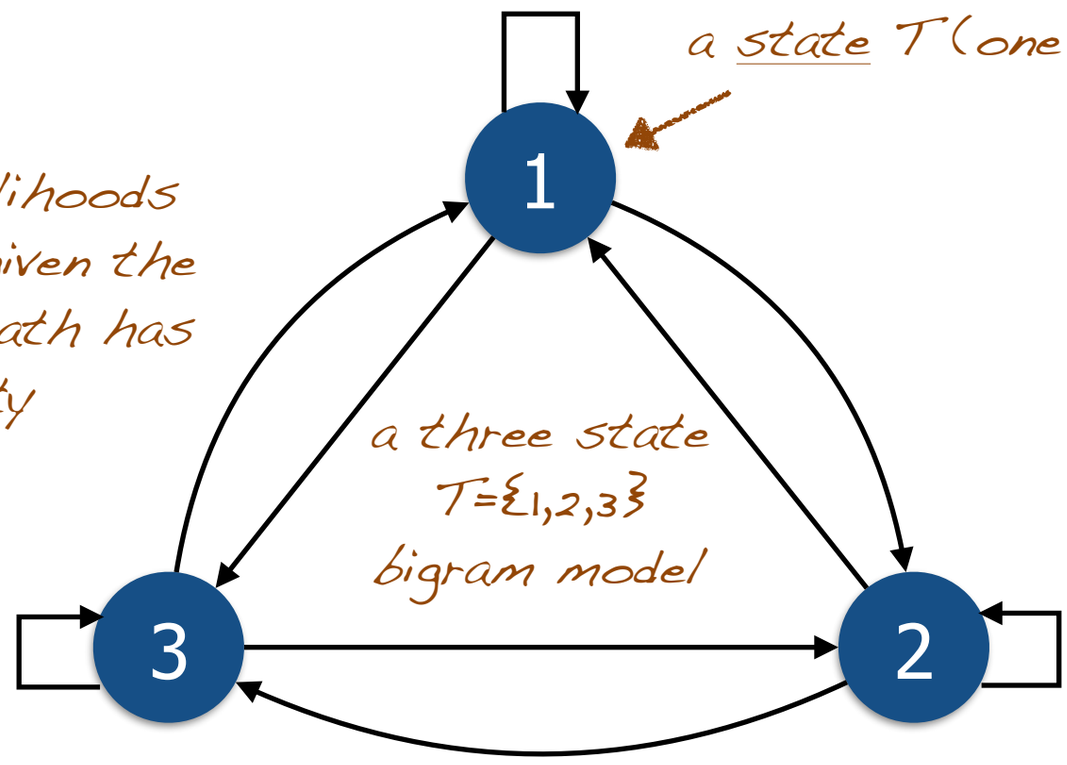
$$P(\text{many} \mid \text{this, is, a, long, sentence, with}) \times$$

….

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

The Markov property

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

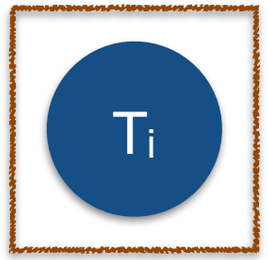


A. A. Markov (1886).

Modeling the stochastic process of "generating words" assuming it is Markovian

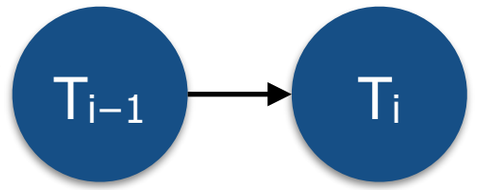
$$\prod P(w_i|w_1^{i-1}) \approx \prod P(w_i|w_{i-k}^{i-1})$$

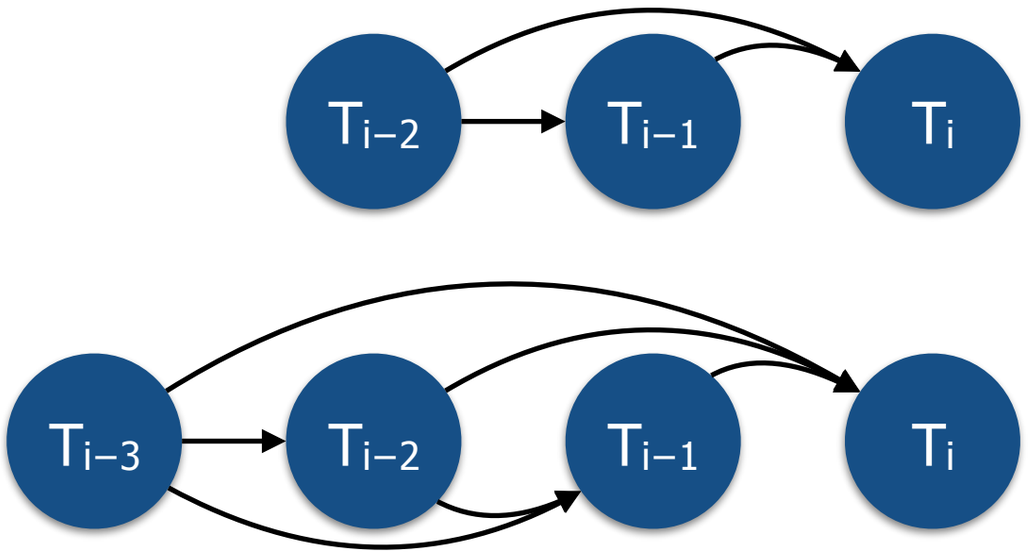
stochastic process: "Unigram Model"



- 1st Order Markov Chains
  - Bigram Model, k=1, P(be | this)
- 2nd Order Markov Chains
  - Trigram Model, k=2, P(long | this, be)
- 3rd Order Markov Chains
  - Quadrigram Model, k=3, P(sentence | this, be, 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:

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

N = total word count

- Bigrams:

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

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

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

k = n-gram size - 1

- **Language Model:**

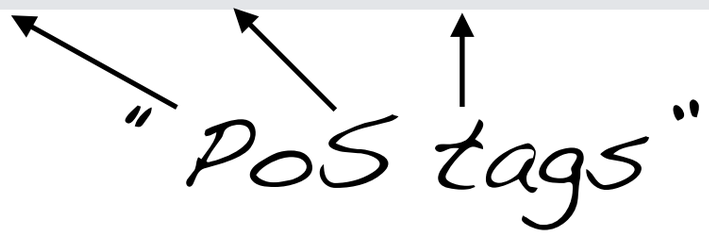
$$P(W) = \prod P(w_i|w_{i-k}^{i-1}) =$$

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

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

The Parts of Speech (PoS)

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



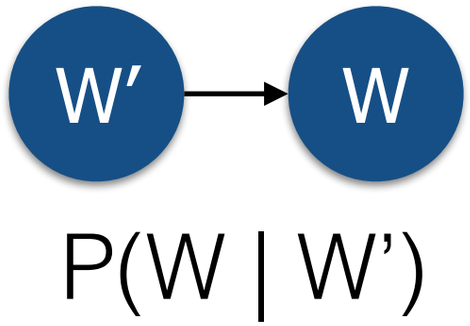
- The Parts of Speech:
  - noun: NN, verb: VB, adjective: JJ, adverb: RB, preposition: IN, personal pronoun: PRP, …
  - e.g. the full **Penn Treebank PoS tagset** contains 48 tags:
  - 34 grammatical tags (i.e., "real" parts-of-speech) for words
  - one for cardinal numbers ("CD"; i.e., a series of digits)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

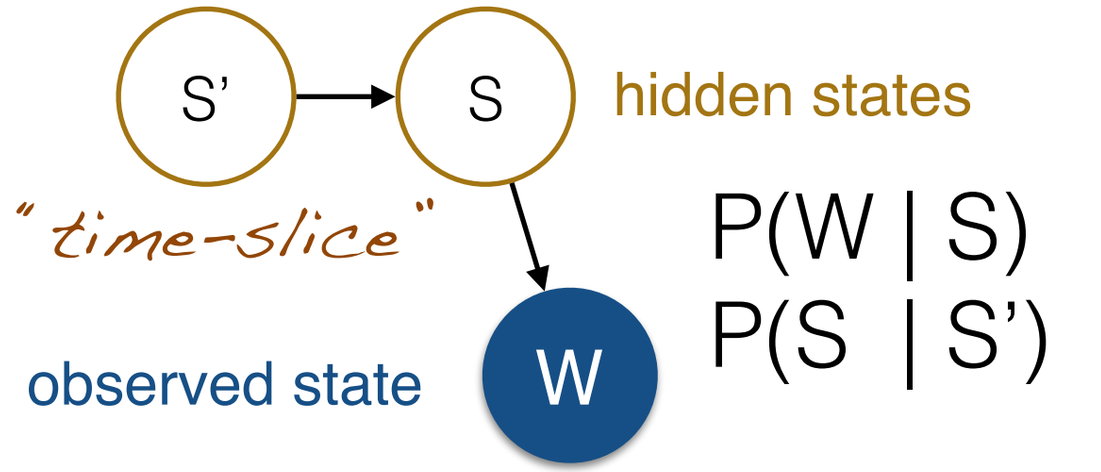
The Parts of Speech (2/2)

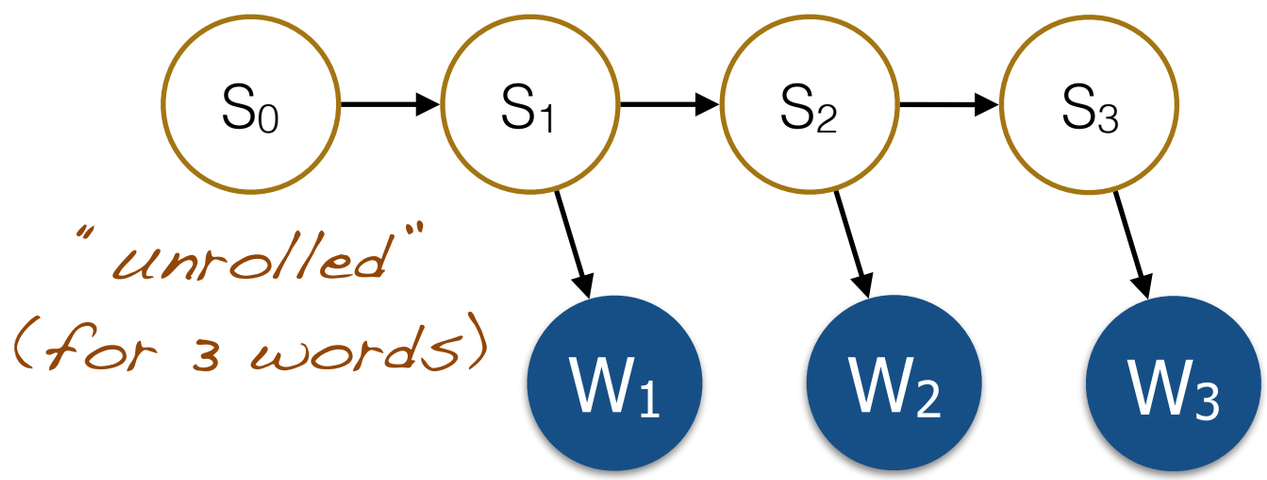
I	ate	the	pizza	with	green	peppers	.
PRP	VB	DT	NN	IN	JJ	NN	.

- 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)

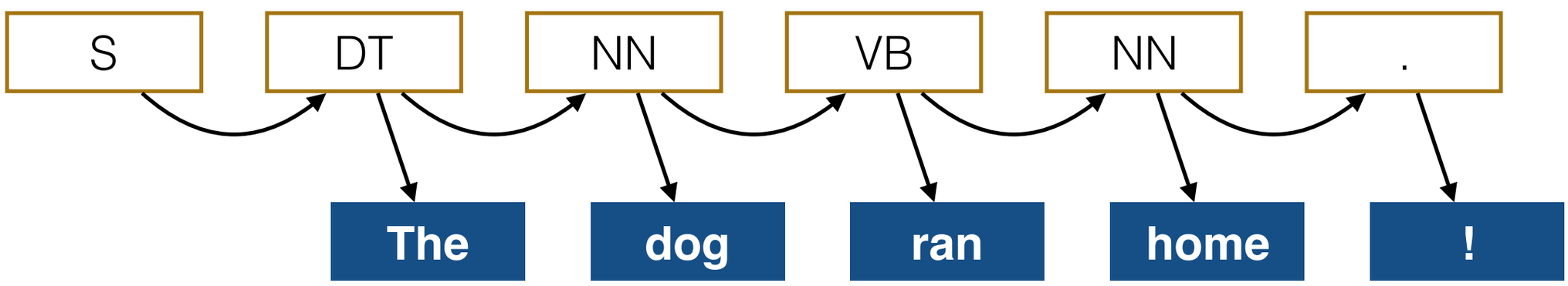






A language-based intuition for HMMs

- 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

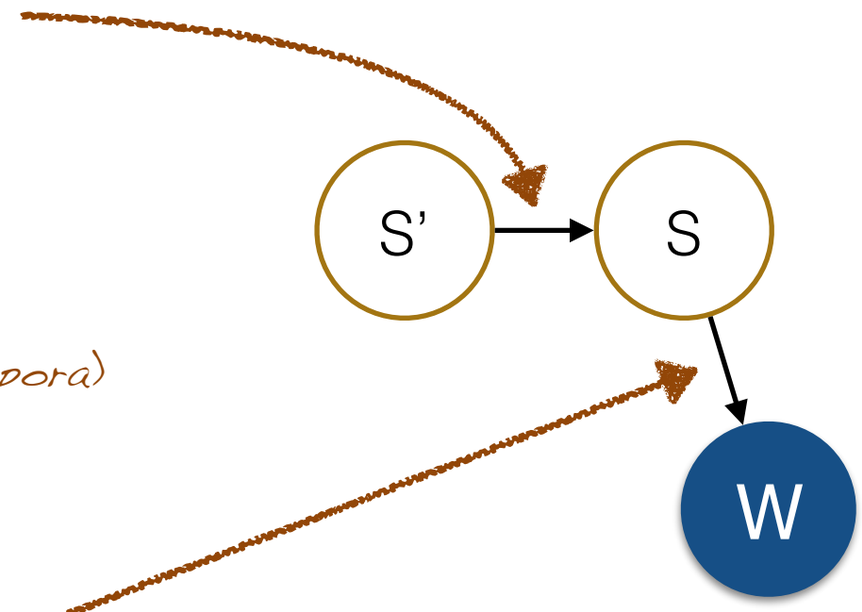
P(s | s')	DT	NN	VB	…
DT	0.03	0.7	0	
NN	0	0	0.5	
VB	0	0.5	0.2	
…				

(measured as discrete factor tables from annotated PoS corpora)

P(w | s)	$word_1$	$word_2$	$word_3$	…
DT	0.3	0	0	
NN	0.0001	0.002	0	
VB	0	0	0.001	
…				

underflow danger ➡ use "log Ps"!

Transition Matrix



Observation Matrix

very sparse (W is large)
➡ Smoothing!

Three limitations of HMMs

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

Example issue: trigrams (long-range dependencies!)

"unsolved" (CRF)

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

Example issue: word morphology (inflection, declension!)

MEMM

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

Example issue: position in sentence

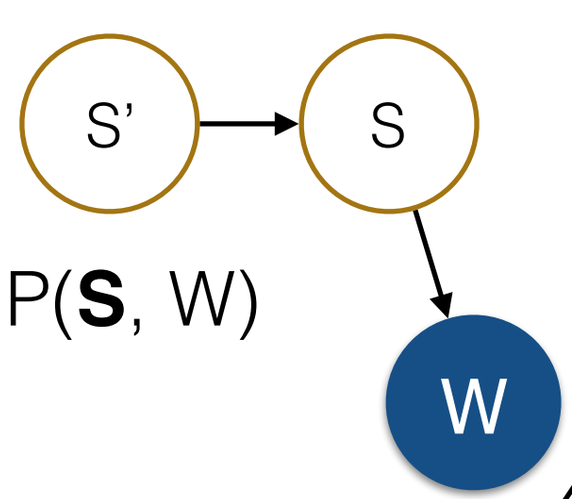
(label bias problem, see next!)

CRF

From generative to discriminative sequence models

Hidden Markov Model

(first oder version)

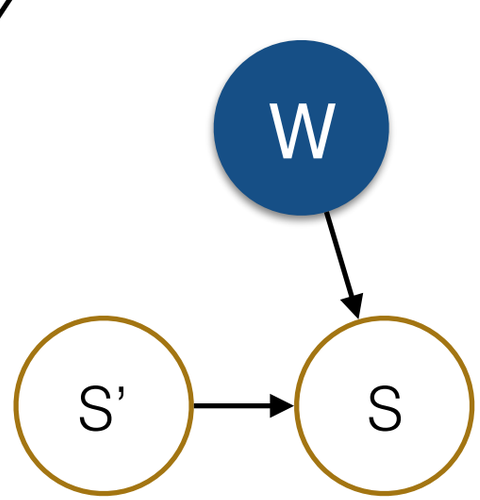


P(**S**, W)

generative model; lower bias is beneficial for small training sets

Conditional Random Field

(linear chain version)

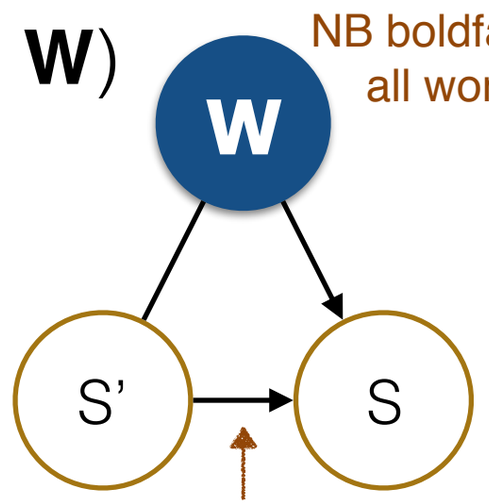


P(S | S', W)

Maximum Entropy Markov Model

(first oder version)

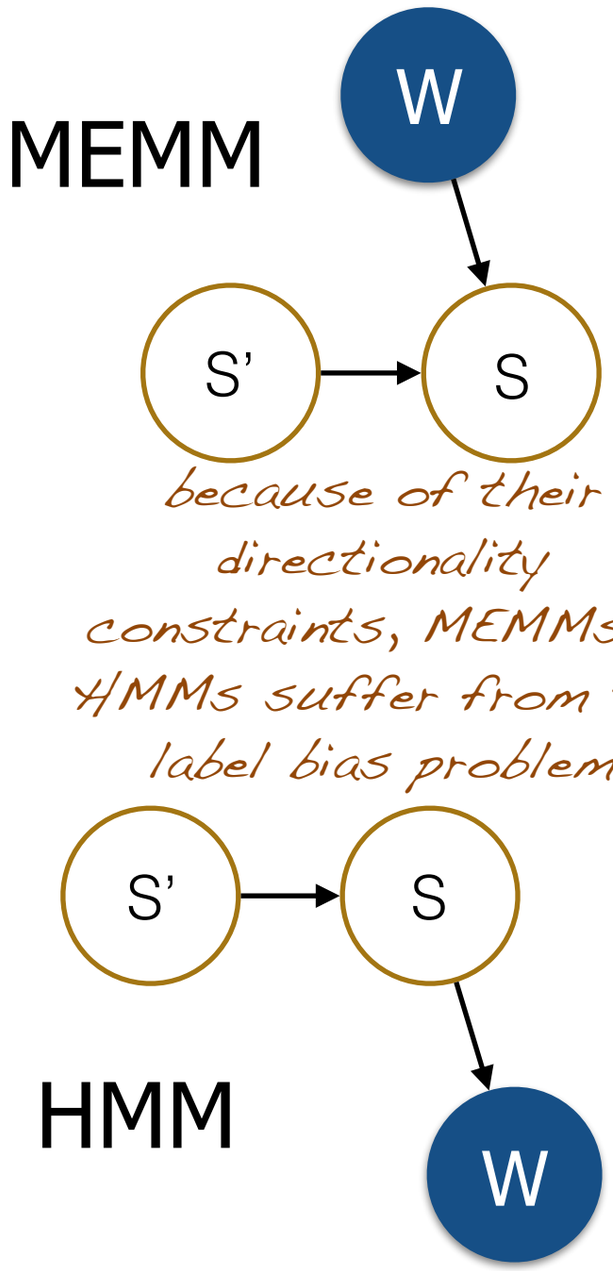
P(S | S', **W**)



NB boldface W: all words!

this "clique" makes CRFs expensive to compute

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** (20th 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**
  - don’t say you’re in a man**’s** world…

not a contraction: possessive s

- **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 (2nd 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

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	VBN	see	O
in	IN	in	O
monocytes	NNS	monocyte	B-cell
.	.	.	O

chunk

noun-phrase (chunk)

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) **W**ord

Named Entity Recognition (NER)

How much training data do I need? "corpora list"

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.

Date
Location
Money
Organization
Percentage
Person

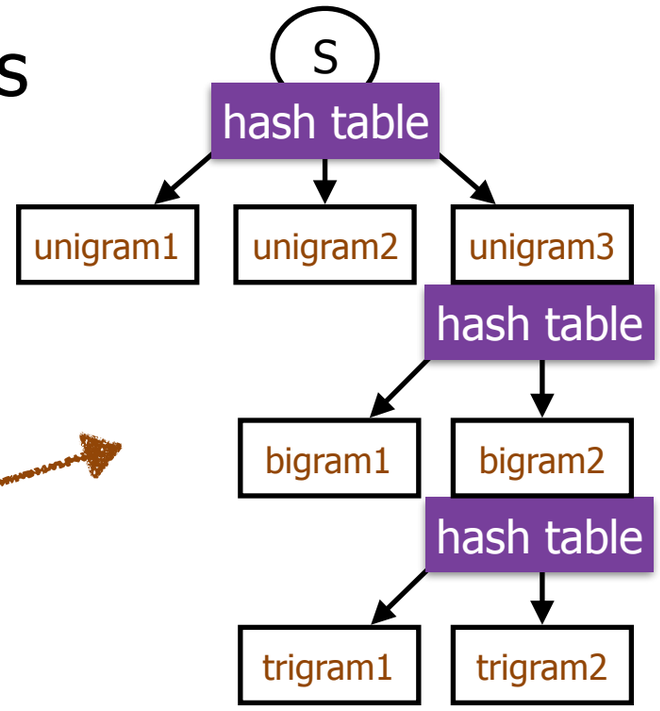
- ➡ Conditional Random Field
- ➡ Ensemble Methods; +SVM, HMM, MEMM, … ➡ `pyensemble`

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)

POLITÉCNICA
"Ingeniamos el futuro"

CAMPUS
DE EXCELENCIA
INTERNACIONAL

Text Mining 5
Language Processing

Madrid Summer School on
Advanced Statistics and Data Mining

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

Evaluation metrics for classification tasks

Evaluations should answer questions like:

How to measure a change to an approach?

Did adding a feature improve or decrease performance?

Is the approach good at locating the relevant pieces or good at excluding the irrelevant bits?

How do two or more different methods compare?

Essential evaluation metrics: Accuracy, F-Measure, MCC Score

Patient→ Doctor↓	has cancer	is healthy
diagnose cancer	TP	FP
detects nothing	FN	TN

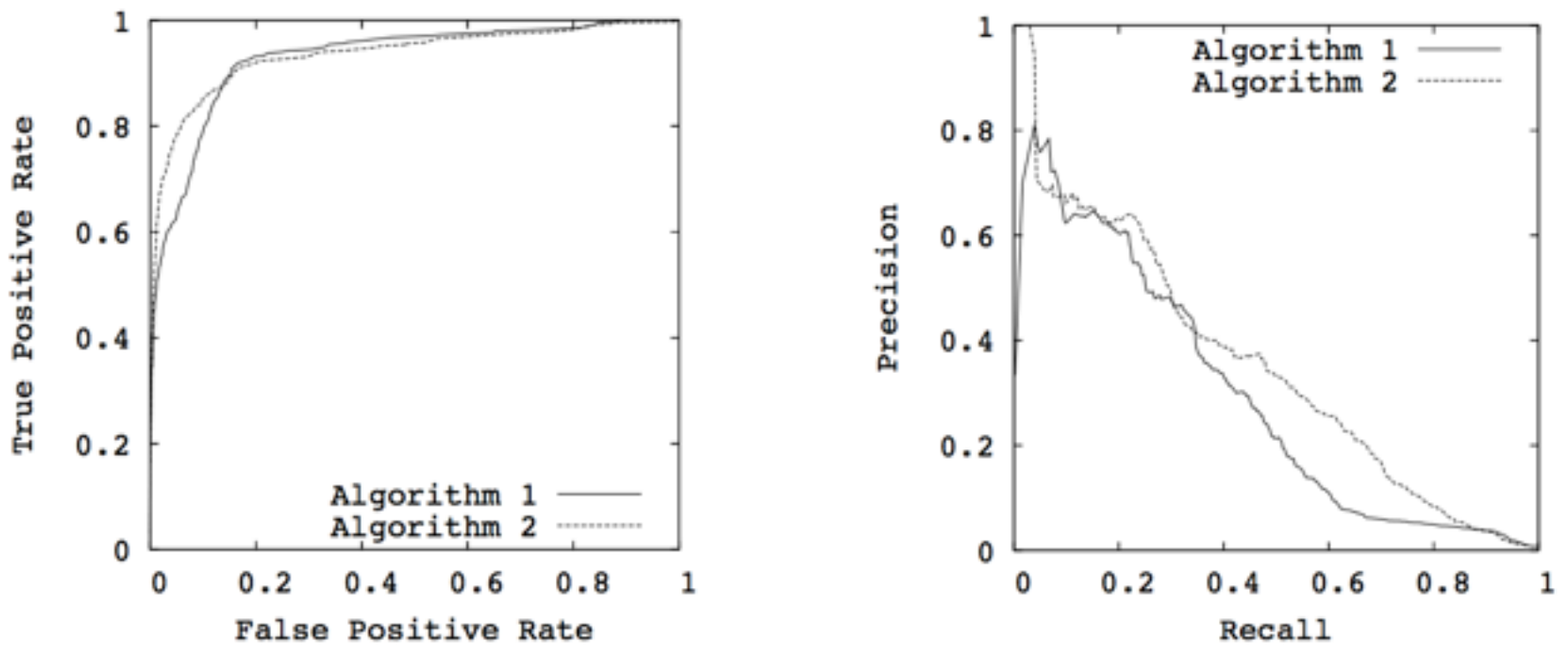
- **Precision** (P)
  - correct hits [TP] ÷ all hits [TP + FP]
- **Recall** (R; **Sensitivity**, TPR)
  - correct hits [TP] ÷ true cases [TP + FN]
- **Specificity** (True Negative Rate)
  - correct misses [TN] ÷ negative cases [FP + TN]

NB: no result order

- **Accuracy**
  - correct classifications [TP + TN] ÷ all cases [TP + TN + FN + FP])
  - highly **sensitive to** class **imbalance**
- **F-Measure** (F-Score)
  - the harmonic mean between P & R
    = 2 TP ÷ (2 TP + FP + FN)
    = (2 P R) ÷ (P + R)
  - does **not** require a **TN** count
- **MCC Score** (Mathew's Correlation Coefficient)
  - χ^2**-based**: (TP TN - FP FN) ÷ sqrt[(TP+FP)(TP+FN)(TN+FP)(TN+FN)]
  - **robust against** class **imbalance**

Ranked evaluation results: AUC ROC and PR

Area Under the Curve
Receiver-Operator Characteristic
Precision-Recall

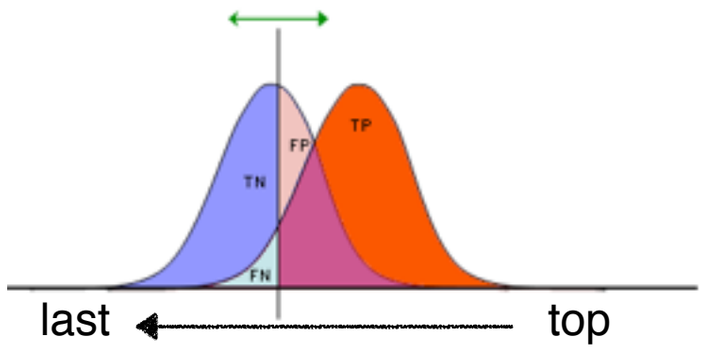


Davis & Goadrich.
ICML 2006

TPR / Recall (aka. Sensitivity)
TP ÷ (TP + FN)

FPR (not Specificity!)
FP ÷ (TN + FP)

Precision
TP ÷ (TP + FP)



TP	FP
FN	TN
1	1

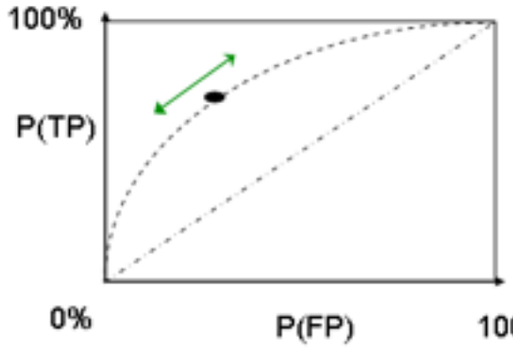


Image Source: WikiMedia Commons, kku ("kakau", eddie)

To ROC or to PR?

Curve I:
10 hits in the top 10, and 10 hits spread over the next 1500 results.

AUC ROC
0.813

Results: 20 T ≪ 1980 N

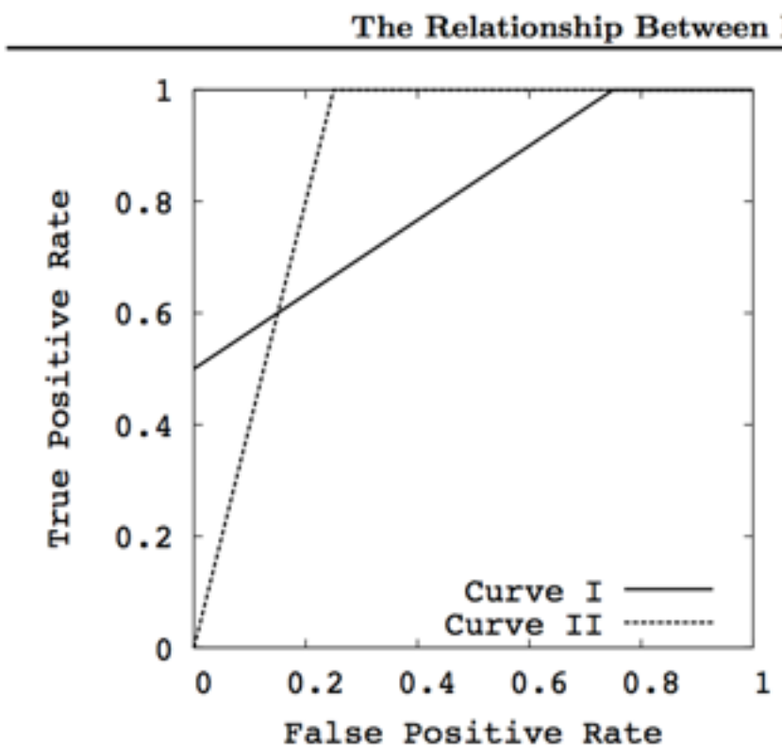


Figure 11. Comparing AUC-ROC for Two Algorithms

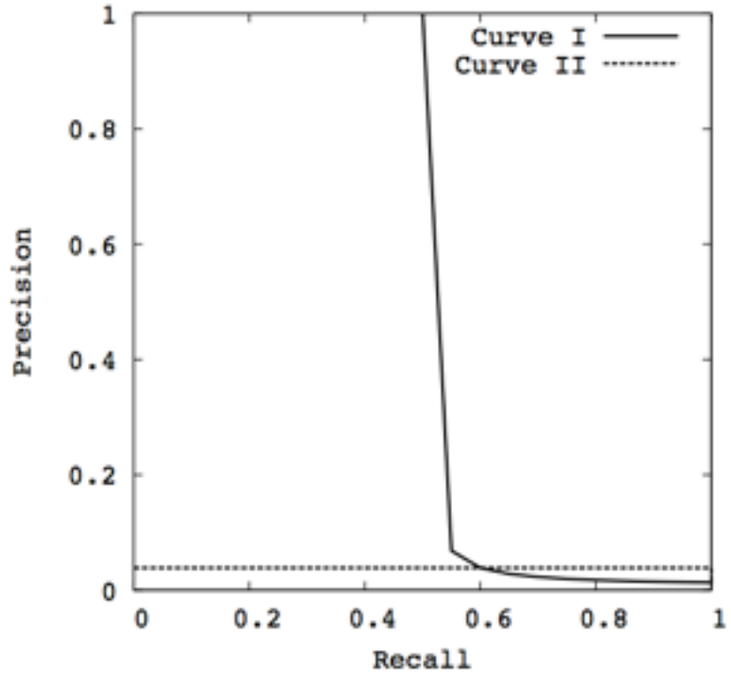


Figure 12. Comparing AUC-PR for Two Algorithms

Curve II:
Hits spread evenly over the first 500 results.

AUC ROC
0.875

"An algorithm which optimizes the area under the ROC curve is not guaranteed to optimize the area under the PR curve."

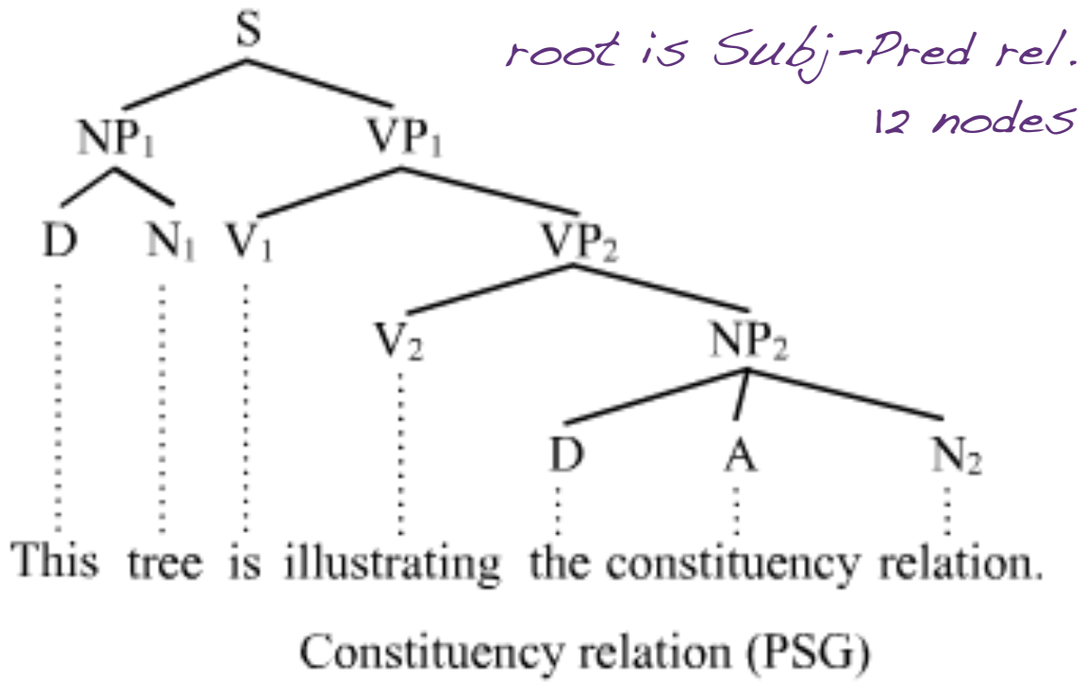
Davis & Goadrich, 2006

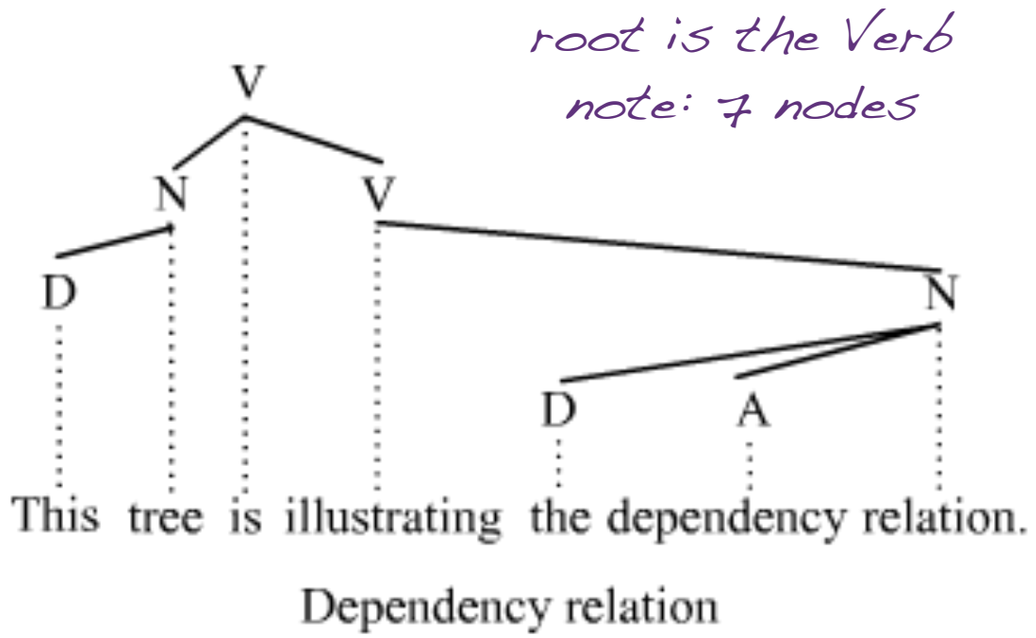
- Davis & Goadrich. The Relationship Between PR and ROC Curves. ICML 2006
- Landgrebe et al. Precision-recall operating characteristic (P-ROC) curves in imprecise environments. Pattern Recognition 2006
- Hanczar et al. Small-Sample Precision of ROC-Related Estimates. Bioinformatics 2010

→ Use (AUC) PR for [imbalanced] ranking scenarios!

Detecting grammatical (sentence) structure

Phrase-structure (aka. **constituency**) vs. **dependency** grammars





https://en.wikipedia.org/wiki/Phrase_structure_grammar

P-S Grammars: **Chomsky**; Dependency Grammars: **Tesnière**

Dependency relations can be annotated with a linear-time parser.

note the one-to-many constituency vs. the one-to-one dependency relations

Tesnière's dependency relations (1959)

He ate pizza with anchovies. He ate pizza with friends.

ate(he, pizza with anchovies)
~~ate(he, with anchovies)~~

Relationships

ate(he, pizza)
ate(he, with friends)

NB: Dependencies cannot capture **phrasal structure** (subject, object, verb phrase, etc.), and in particular, **word order**.

which can be a benefit: some languages have a free word order, e.g. Turkish or Czech
reminder: clauses and collocations are special phrasal structures

Dependency parsing 1/2

- Transition-based, arc-standard, shift-reduce, greedy parsing.
- The default approach to dependency parsing today is O(n).

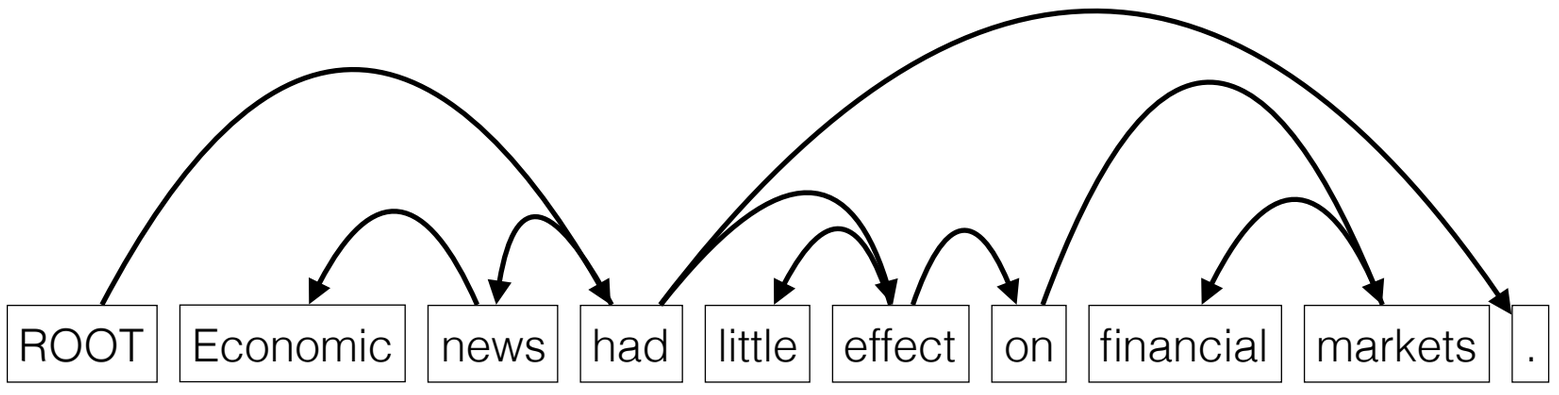
  - **Transition-based**: Move from one token to the next.
  - **Arc-standard**: assign arcs when the dependent token (at the arrowhead) is fully resolved (common alternative: arc-eager → assign the arcs immediately).
  - **Shift-reduce**: A stack of words and a stream buffer: either shift next word from the buffer to the stack or reduce a word from the stack by "arcing".
  - **Greedy**: Make locally optimal transitions (assume independence of arcs).

A shift-reduce parse

Buffer: Economic | news | had | little | effect | on | financial | markets | .

Stack: . | markets | financial | on | effect | little | had | news | Economic | ROOT

Shift: Economic	Shift: financial
Shift: news	Shift: markets
Left-arc: Economic←news	Left-arc: financial←markets
Shift: had	Right-arc: on→markets
Left-arc: news←had	Right-arc: effect→on
Shift: little	Right-arc: had→effect
Shift: effect	Shift: .
Left-arc: little←effect	Right-arc: had→.
Shift: on	Right-arc: ROOT→had



Dependency Parsing. Kübler et al., 2009

Dependency parsing 2/2

- (Arc-standard) Transitions: **shift** or **reduce** (left-arc, right-arc)
- Transitions are chosen using some classifier
  - Maximum entropy classifier, support vector machine, single-layer perceptron, perceptron with one hidden layer (→ Stanford parser, 2014 edition, SpaCy v1), more complex deep nets (→ Google's SyntaxNet, SpaCy v2)
- Main issues:
  - Few large, well annotated training corpora ("dependency **treebanks**"). Biomedical domain: GENIA; Newswire: WSJ, Prague, Penn, …
  - **Non-projective** trees (i.e., trees with arcs crossing each other; common in a number of other languages, e.g. German) with arcs that have to be drawn between nodes that are not adjacent on the stack.

Four approaches to relationship extraction

- **Co-mention window**
  - E.g.: if ORG and LOC entity within same sentence and no more than x tokens in between, treat the pair as a hit.
  - Low precision, high recall; trivial, many false positives.
- **Dependency parsing**
  - If a path covering certain nodes (e.g. prepositions like "in/IN" or predicates [~verbs]) connects two entities, extract that pair.
  - Balanced precision and recall, computationally expensive.
- **Pattern extraction** (over the seq. tags)
  - e.g.: <ORG>+ <IN> <LOC>+ (IN: preposition)
  - High precision, low recall; cumbersome, but very common.
  - Pattern **learning** can help.
- **Machine Learning**
  - Features for sentences with entities and some classifier (e.g., SVM, neural net, MaxEnt, Bayesian net, …)
  - Highly variable milages.

token-distance, num. of tokens between the entities, tokens before/after them, etc.

… but loads of fun in your speaker's opinion :)


Our Text Analytics software uses the most advanced NLP and Machine Learning!
But is it Gluten Free?
MBA Rule #1:
Always Counter Buzz Words with Buzz Words
@TomHCAnderson
tomhcanderson.com

The End.