UNIVERSITY OF WATERLOO

Big Data Infrastructure

CS 489/698 Big Data Infrastructure (Winter 2016)

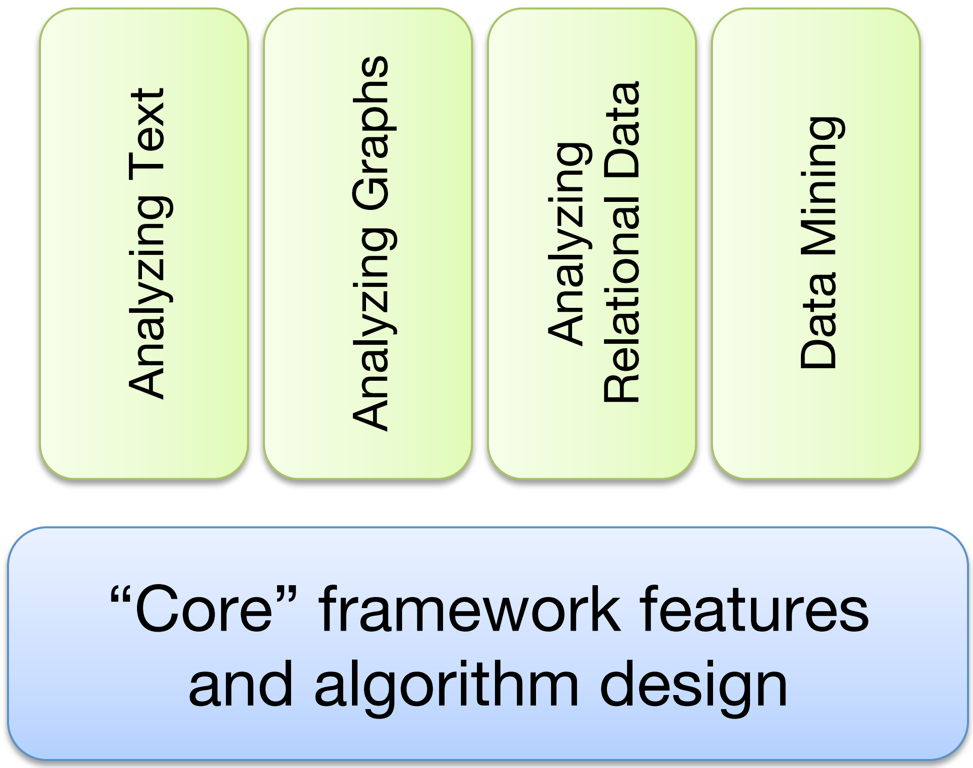
Week 4: Analyzing Text (1/2)

January 26, 2016

Jimmy Lin
David R. Cheriton School of Computer Science
University of Waterloo

These slides are available at http://lintool.github.io/bigdata-2016w/

Structure of the Course

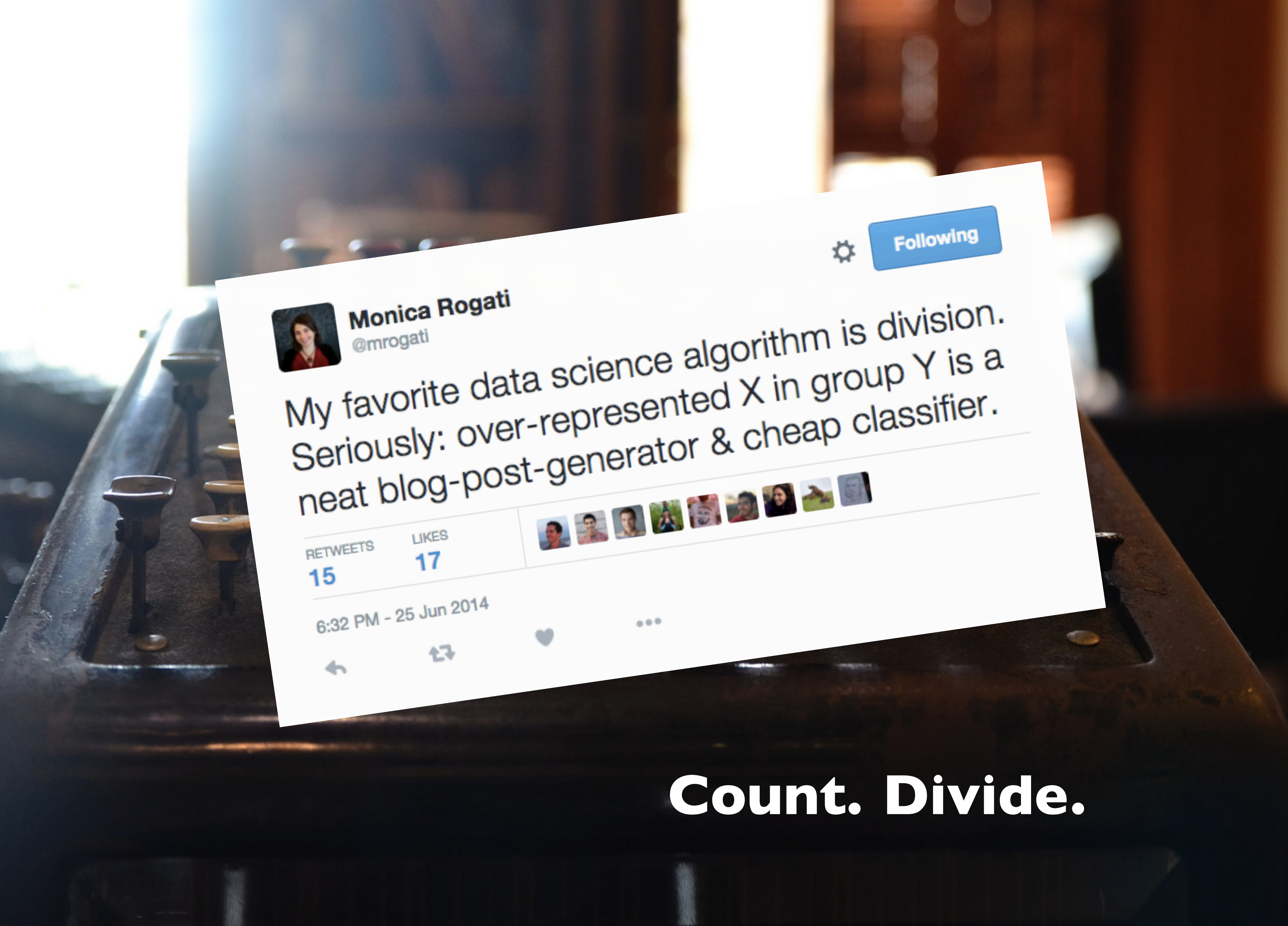


Source: http://www.flickr.com/photos/guvnah/7861418602/

Count.
(Efficiently)

```
class Mapper
    method Map(docid a, doc d)
        for all term t ∈ doc d do
            Emit(term t, count 1)

class Reducer
    method Reduce(term t, counts [c1, c2, ...])
        sum ← 0
        for all count c ∈ counts [c1, c2, ...] do
            sum ← sum + c
        Emit(term t, count s)
```



Source: http://www.flickr.com/photos/guvnah/7861418602/ https://twitter.com/mrogati/status/481927908802322433

Pairs. Stripes.

Seems pretty trivial…

More than a "toy problem"?
Answer: language models

Language Models

$$P(w_1, w_2, \ldots, w_T)$$

What are they?
How do we build them?
How are they useful?

Language Models

$$P(w_1, w_2, \ldots, w_T)$$

$$= P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\ldots P(w_T|w_1, \ldots, w_{T-1})$$

[chain rule]

Is this tractable?

Approximating Probabilities: *N*-Grams

Basic idea: limit history to fixed number of (*N* – 1) words (Markov Assumption)

$$P(w_k|w_1,\ldots,w_{k-1}) \approx P(w_k|w_{k-N+1},\ldots,w_{k-1})$$

N=1: Unigram Language Model

$$P(w_k|w_1,\ldots,w_{k-1}) \approx P(w_k)$$

$$\Rightarrow P(w_1,w_2,\ldots,w_T) \approx P(w_1)P(w_2)\ldots P(w_T)$$

Approximating Probabilities: *N*-Grams

Basic idea: limit history to fixed number of (*N* – 1) words (Markov Assumption)

$$P(w_k|w_1,\ldots,w_{k-1}) \approx P(w_k|w_{k-N+1},\ldots,w_{k-1})$$

N=2: Bigram Language Model

$$P(w_k|w_1,\ldots,w_{k-1}) \approx P(w_k|w_{k-1})$$

$$\Rightarrow P(w_1,w_2,\ldots,w_T) \approx P(w_1|<\mathrm{S}>)P(w_2|w_1)\ldots P(w_T|w_{T-1})$$

Approximating Probabilities: *N*-Grams

Basic idea: limit history to fixed number of (*N* – 1) words (Markov Assumption)

$$P(w_k|w_1,\dots,w_{k-1}) \approx P(w_k|w_{k-N+1},\dots,w_{k-1})$$

N=3: Trigram Language Model

$$P(w_k|w_1,\dots,w_{k-1}) \approx P(w_k|w_{k-2},w_{k-1})$$

$$\Rightarrow P(w_1,w_2,\dots,w_T) \approx P(w_1|<\mathrm{S}><\mathrm{S}>)\dots P(w_T|w_{T-2}w_{T-1})$$

Building *N*-Gram Language Models

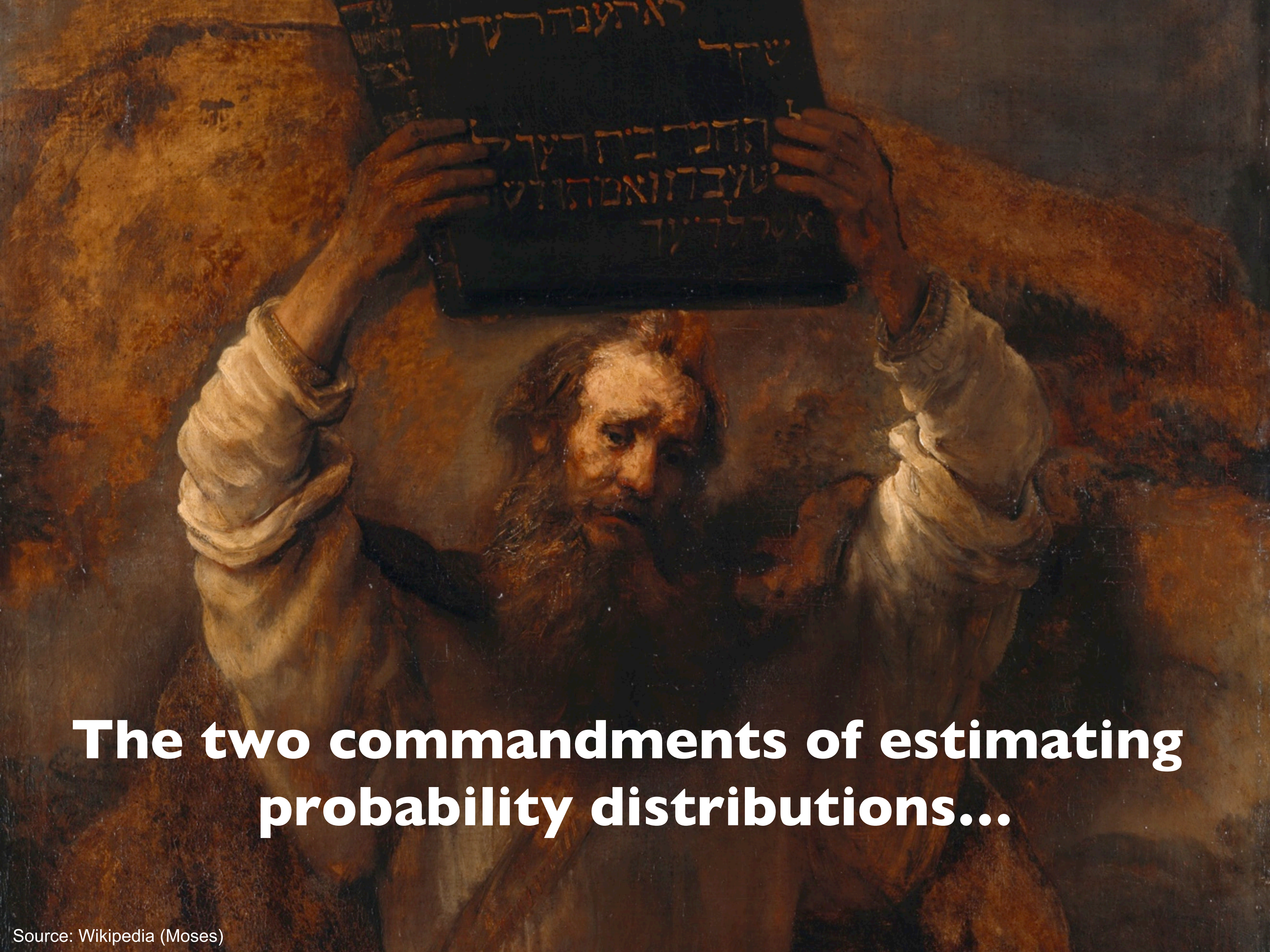
- Compute maximum likelihood estimates (MLE) for individual *n*-gram probabilities
  - Unigram: $P(w_i) = \frac{C(w_i)}{N}$
  - Bigram: $P(w_i, w_j) = \frac{C(w_i, w_j)}{N}$

$$P(w_j|w_i) = \frac{P(w_i, w_j)}{P(w_i)} = \frac{C(w_i, w_j)}{\sum_w C(w_i, w)} \stackrel{?}{=} \frac{C(w_i, w_j)}{C(w_i)}$$

  - Generalizes to higher-order *n*-grams
  - State of the art models use ~5-grams
- We already know how to do this in MapReduce!

Fancy way of saying: count + divide

Minor detail here…

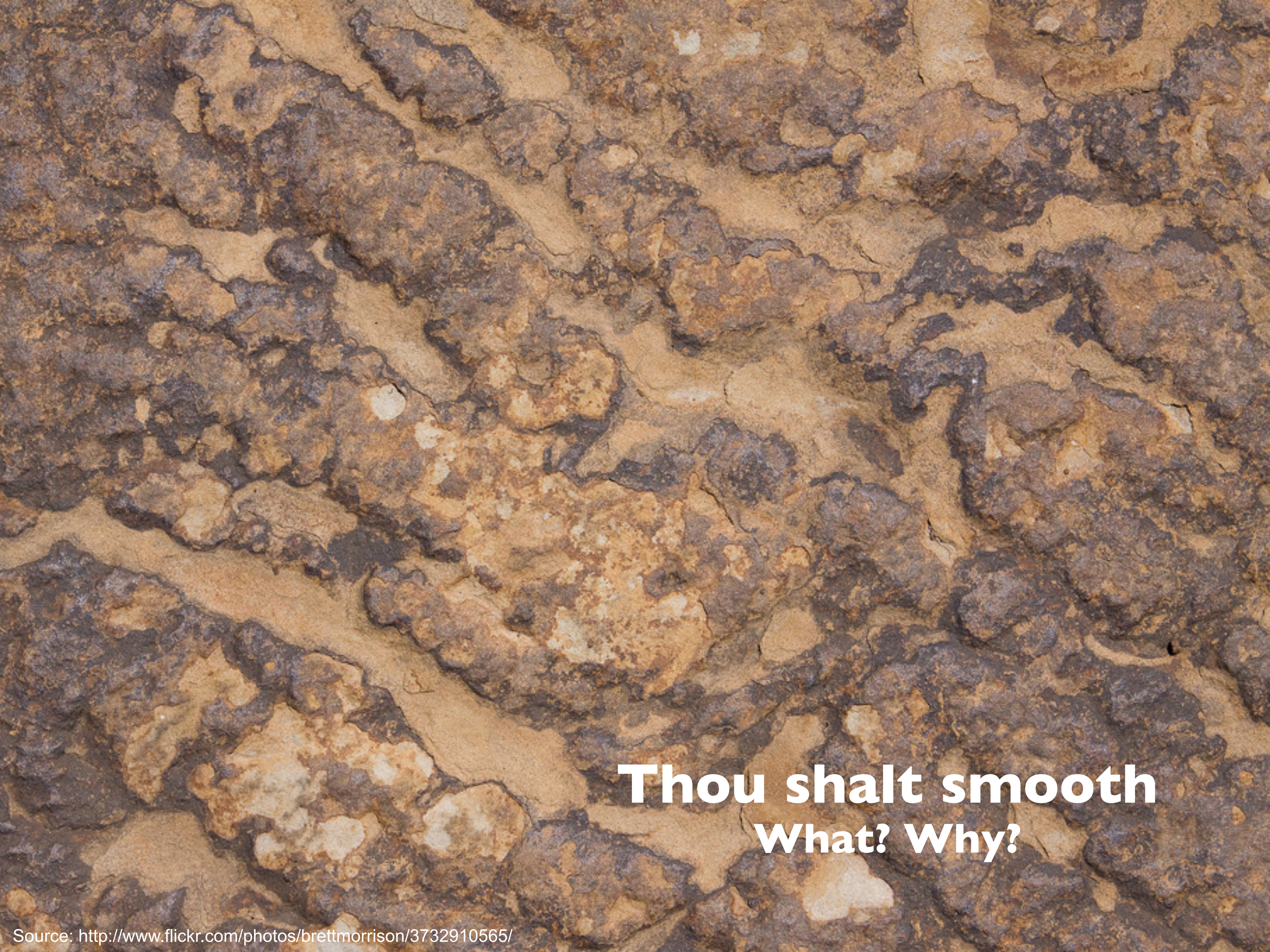


The two commandments of estimating probability distributions…

Source: Wikipedia (Moses)

Probabilities must sum up to one

Source: http://www.flickr.com/photos/37680518@N03/7746322384/



Thou shalt smooth

What? Why?

Source: http://www.flickr.com/photos/brettmorrison/3732910565/

Source: https://www.flickr.com/photos/avlxyz/6898001012/

$$P({\color{green}\bullet}) > P({\color{red}\bullet})$$

$$P({\color{green}\bullet}\ \ \bullet) \;?\; P({\color{red}\bullet}\ \ \bullet)$$

Example: Bigram Language Model

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

Training Corpus

$P(\text{ I} \mid \text{<s>}) = 2/3 = 0.67$
$P(\text{ am} \mid \text{I}) = 2/3 = 0.67$
$P(\text{ </s>} \mid \text{Sam}) = 1/2 = 0.50$
...

$P(\text{ Sam} \mid \text{<s>}) = 1/3 = 0.33$
$P(\text{ do} \mid \text{I}) = 1/3 = 0.33$
$P(\text{ Sam} \mid \text{am}) = 1/2 = 0.50$

Bigram Probability Estimates

Note: We don't ever cross sentence boundaries

Data Sparsity

P(I | <s>) = 2/3 = 0.67
P(am | I) = 2/3 = 0.67
P(</s> | Sam)= 1/2 = 0.50
...

P(Sam | <s>) = 1/3 = 0.33
P(do | I) = 1/3 = 0.33
P(Sam | am) = 1/2 = 0.50

Bigram Probability Estimates

P(I like ham)

= P(I | <s>) P(like | I) P(ham | like) P(</s> | ham)

= 0

Why is this bad?

Issue: Sparsity!

Thou shalt smooth!

- Zeros are bad for any statistical estimator
  - Need better estimators because MLEs give us a lot of zeros
  - A distribution without zeros is “smoother”
- The Robin Hood Philosophy: Take from the rich (seen *n*-grams) and give to the poor (unseen *n*-grams)
  - And thus also called discounting
  - Make sure you still have a valid probability distribution!
- Lots of techniques:
  - Laplace, Good-Turing, Katz backoff, Jelinek-Mercer
  - Kneser-Ney represents best practice

Laplace Smoothing

- Simplest and oldest smoothing technique
- Just add 1 to all n-gram counts including the unseen ones
- So, what do the revised estimates look like?

Learn fancy words
for simple ideas!

Laplace Smoothing

Unigrams

$$P_{MLE}(w_i) = \frac{C(w_i)}{N} \longrightarrow P_{LAP}(w_i) = \frac{C(w_i)+1}{N+V}$$

Bigrams

$$P_{MLE}(w_i, w_j) = \frac{C(w_i, w_j)}{N} \longrightarrow P_{LAP}(w_i, w_j) = \frac{C(w_i, w_j)+1}{N+V^2}$$

Careful, don't confuse the N's!

$$P_{LAP}(w_j|w_i) = \frac{P_{LAP}(w_i, w_j)}{P_{LAP}(w_i)} = \frac{C(w_i, w_j)+1}{C(w_i)+V}$$

What if we don't know V?

Jelinek-Mercer Smoothing: Interpolation

- Mix a trigram model with bigram and unigram models to offset sparsity
- Mix = Weighted Linear Combination

$$P(w_k|w_{k-2}w_{k-1}) =$$

$$\lambda_1 P(w_k|w_{k-2}w_{k-1}) + \lambda_2 P(w_k|w_{k-1}) + \lambda_3 P(w_k)$$

$$0 <= \lambda_i <= 1 \qquad \sum_i \lambda_i = 1$$

Kneser-Ney Smoothing

- Kneser-Ney: Interpolate discounted model with a special "continuation" unigram model
  - Based on appearance of unigrams in different contexts
  - Excellent performance, state of the art

$$P_{KN}(w_k|w_{k-1}) = \frac{C(w_{k-1}w_k) - D}{C(w_{k-1})} + \beta(w_k)P_{CONT}(w_k)$$

$$P_{CONT}(w_i) = \frac{N(\bullet\ w_i)}{\sum_{w'} N(\bullet\ w')}$$

$N(\bullet\ w_i)$ = number of different contexts w_i has appeared in

Kneser-Ney Smoothing: Intuition

- I can’t see without my ____________
- “San Francisco” occurs a lot
- I can’t see without my Francisco?

Stupid Backoff

- Let's break all the rules:

$$S(w_i|w_{i-k+1}^{i-1}) = \begin{cases} \frac{f(w_{i-k+1}^{i})}{f(w_{i-k+1}^{i-1})} & \text{if } f(w_{i-k+1}^{i}) > 0 \\ \alpha S(w_i|w_{i-k+2}^{i-1}) & \text{otherwise} \end{cases}$$

$$S(w_i) = \frac{f(w_i)}{N}$$

- But throw *lots* of data at the problem!

Source: Brants et al. (EMNLP 2007)

Stupid Backoff Implementation: Pairs!

- Straightforward approach: count each order separately

A B	← remember this value	
A B C		S(C|A B) = f(A B C)/f(A B)
A B D		S(D|A B) = f(A B D)/f(A B)
A B E		S(E|A B) = f(A B E)/f(A B)
…		…

- More clever approach: count *all* orders together

A B	← remember this value
A B C	← remember this value
A B C P	
A B C Q	
A B D	← remember this value
A B D X	
A B D Y	
…	

Stupid Backoff: Additional Optimizations

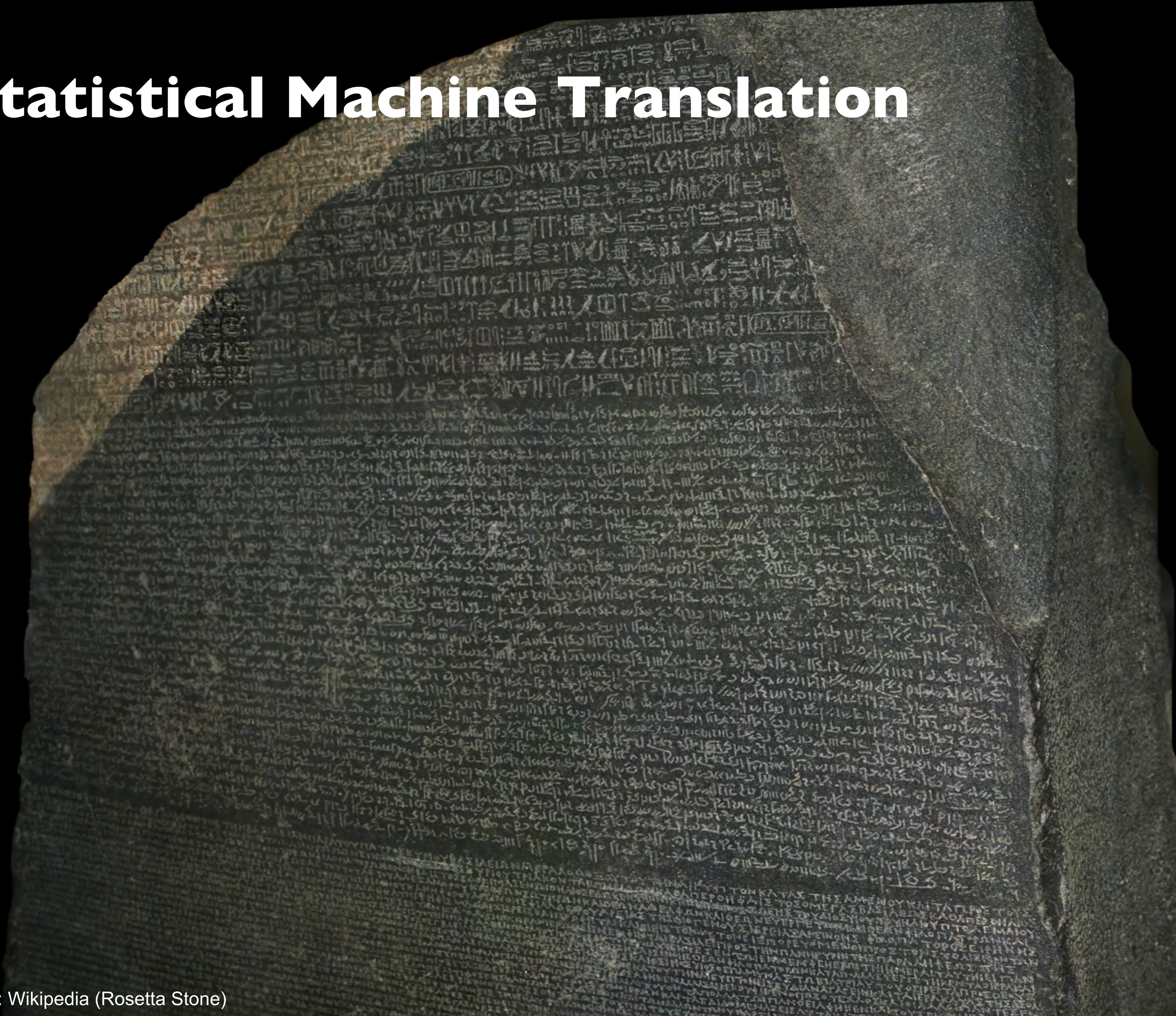
- Replace strings with integers
  - Assign ids based on frequency (better compression using vbyte)
- Partition by bigram for better load balancing
  - Replicate all unigram counts

State of the art smoothing (less data)

vs. Count and divide (more data)

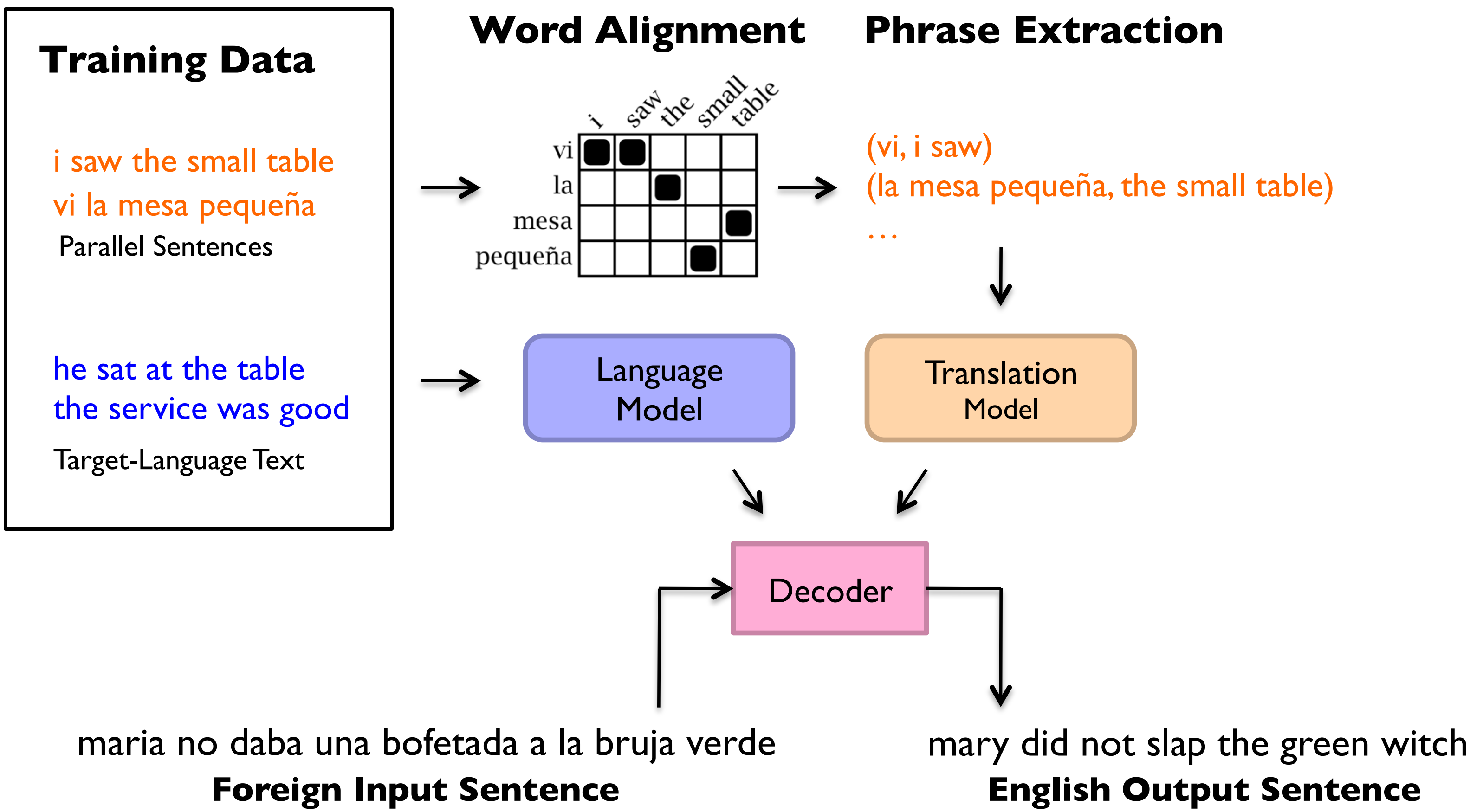
Source: Wikipedia (Boxing)

Statistical Machine Translation



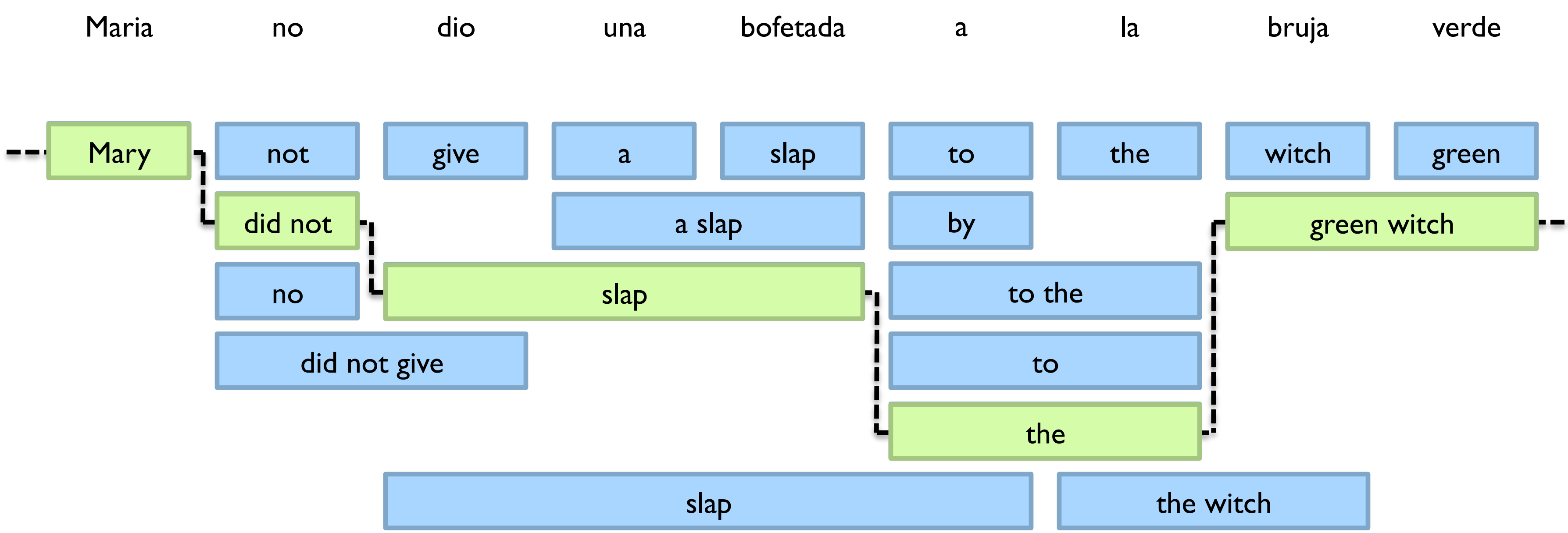
Source: Wikipedia (Rosetta Stone)

Statistical Machine Translation



$$\hat{e}_1^I = \arg\max_{e_1^I}\left[P(e_1^I \mid f_1^J)\right] = \arg\max_{e_1^I}\left[P(e_1^I)P(f_1^J \mid e_1^I)\right]$$

Translation as a Tiling Problem



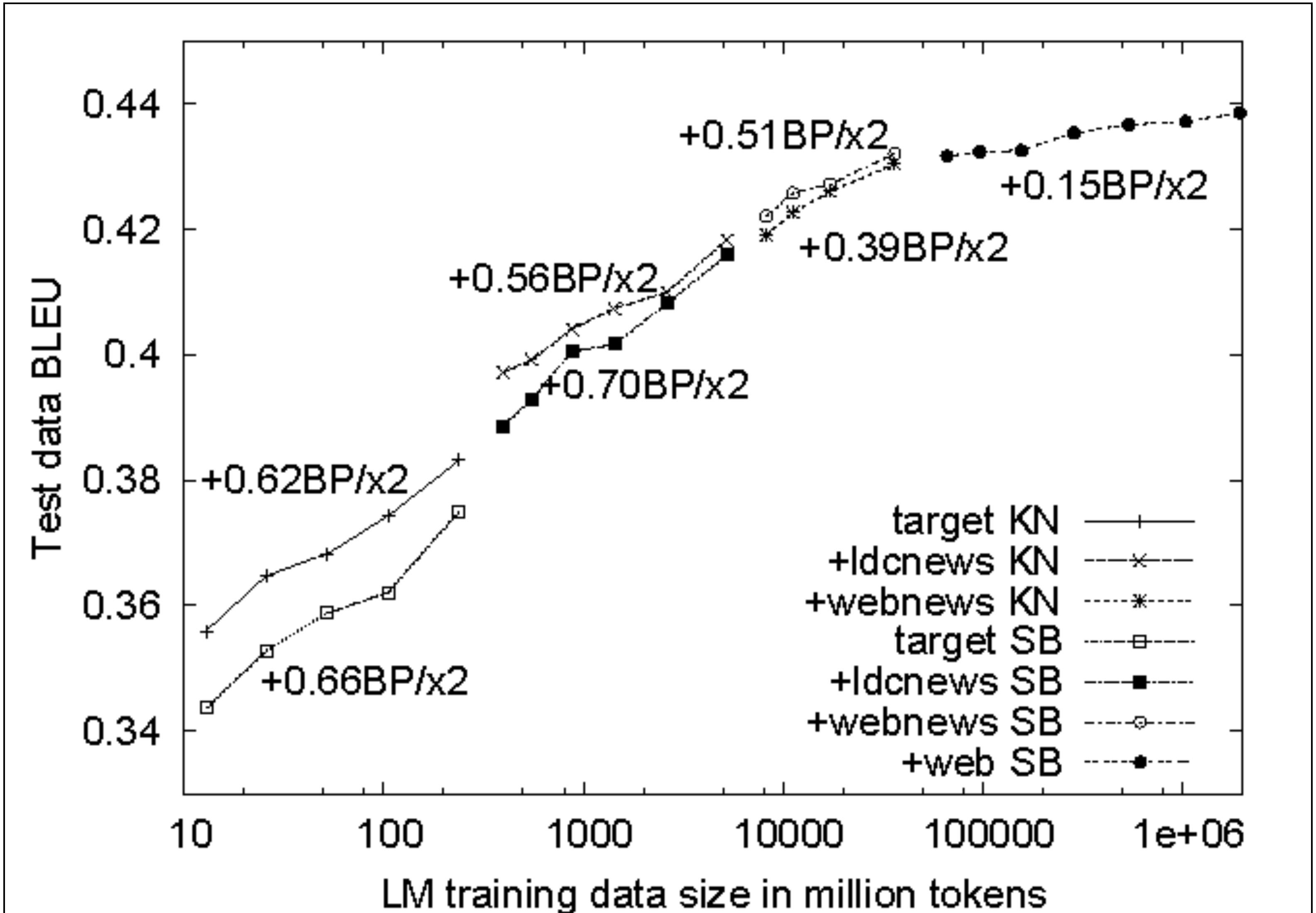
$$\hat{e}_1^I = \arg\max_{e_1^I}\left[P(e_1^I \mid f_1^J)\right] = \arg\max_{e_1^I}\left[P(e_1^I)P(f_1^J \mid e_1^I)\right]$$

Results: Running Time

	target	*webnews*	*web*
# tokens	237M	31G	1.8T
vocab size	200k	5M	16M
# n-grams	257M	21G	300G
LM size (SB)	2G	89G	1.8T
time (SB)	20 min	8 hours	1 day
time (KN)	2.5 hours	2 days	–
# machines	100	400	1500

Source: Brants et al. (EMNLP 2007)

Results: Translation Quality



Source: Brants et al. (EMNLP 2007)

What’s actually going on?

$$P(e|f) = \frac{P(e) \cdot P(f|e)}{P(f)}$$

$$\hat{e} = \arg\max_e P(e)P(f|e)$$

Source: http://www.flickr.com/photos/johnmueller/3814846567/in/pool-56226199@N00/

$$P(e|f) = \frac{P(e) \cdot P(f|e)}{P(f)}$$

$$\hat{e} = \arg\max_{e} P(e)P(f|e)$$

Source: http://www.flickr.com/photos/johnmueller/3814846567/in/pool-56226199@N00/

$$P(e|f) = \frac{P(e) \cdot P(f|e)}{P(f)}$$

$$\hat{e} = \arg\max_{e} P(e)P(f|e)$$

Source: http://www.flickr.com/photos/johnmueller/3814846567/in/pool-56226199@N00/

Source: http://www.flickr.com/photos/guvnah/7861418602/

First, nomenclature…

- Search and information retrieval (IR)
  - Focus on textual information (= text/document retrieval)
  - Other possibilities include image, video, music, …
- What do we search?
  - Generically, “collections”
  - Less-frequently used, “corpora”
- What do we find?
  - Generically, “documents”
  - Even though we may be referring to web pages, PDFs, PowerPoint slides, paragraphs, etc.

The Central Problem in Search

Searcher

Author

Concepts

Query Terms

"tragic love story"

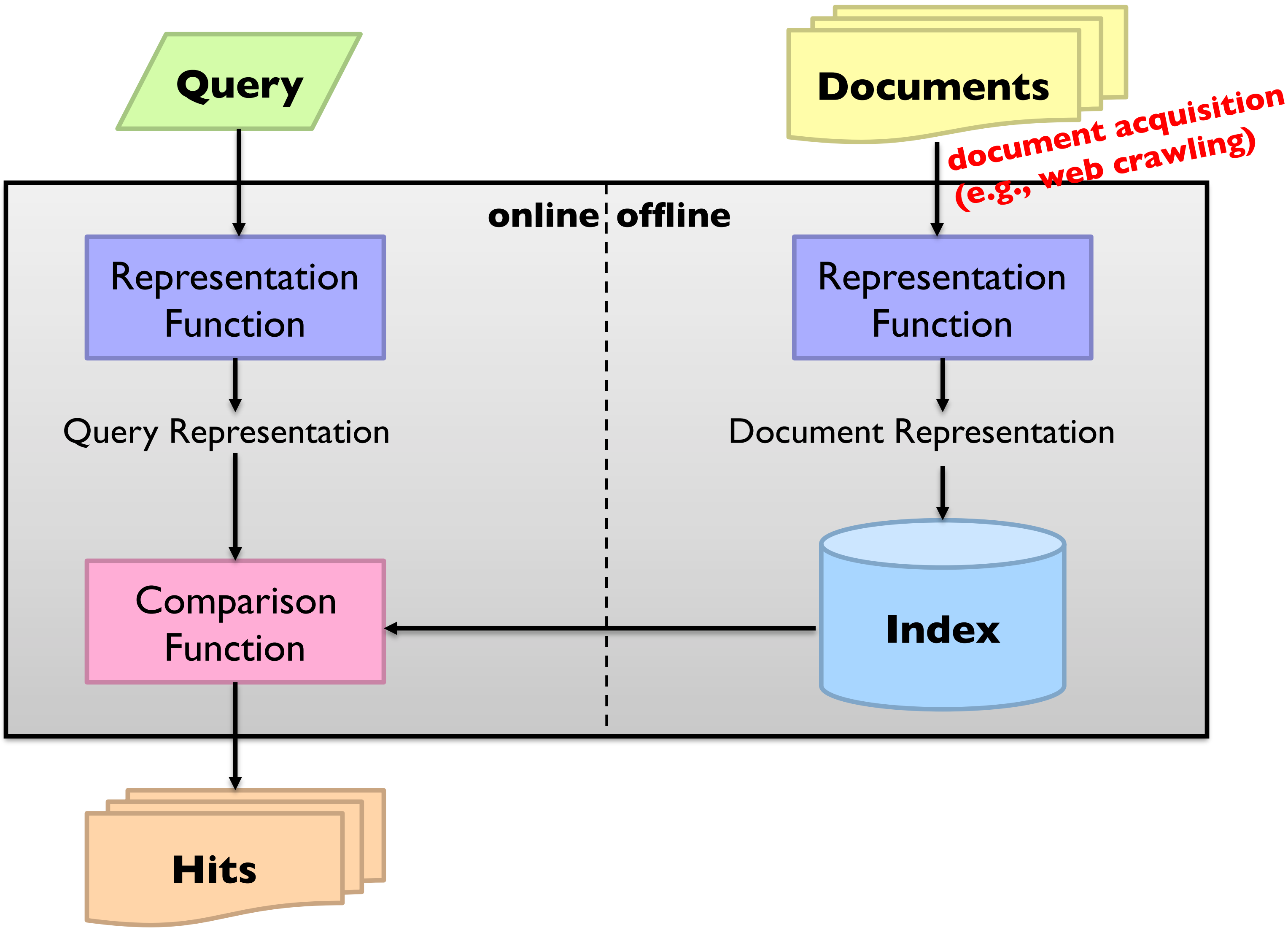
Concepts

Document Terms

"fateful star-crossed romance"

Do these represent the same concepts?

Abstract IR Architecture



How do we represent text?

- Remember: computers don't "understand" anything!
- "Bag of words"
  - Treat all the words in a document as index terms
  - Assign a "weight" to each term based on "importance" (or, in simplest case, presence/absence of word)
  - Disregard order, structure, meaning, etc. of the words
  - Simple, yet effective!
- Assumptions
  - Term occurrence is independent
  - Document relevance is independent
  - "Words" are well-defined

What’s a word?

天主教教宗若望保祿二世因感冒再度住進醫院。
這是他今年第二度因同樣的病因住院。

وقال مارك ريجيف - الناطق باسم
الخارجية الإسرائيلية - إن شارون قبل
الدعوة وسيقوم للمرة الأولى بزيارة
تونس، التي كانت لفترة طويلة المقر
الرسمي لمنظمة التحرير الفلسطينية بعد خروجها من لبنان عام 1982.

Выступая в Мещанском суде Москвы экс-глава ЮКОСа
заявил не совершал ничего противозаконного, в чем
обвиняет его генпрокуратура России.

भारत सरकार ने आर्थिक सर्वेक्षण में वित्तीय वर्ष 2005-06 में सात फ़ीसदी
विकास दर हासिल करने का आकलन किया है और कर सुधार पर ज़ोर दिया है

日米連合で台頭中国に対処…アーミテージ前副長官提言

조재영 기자= 서울시는 25일 이명박 시장이 `행정중심복합도시" 건설안에 대해 `
군대라도 동원해 막고싶은 심정"이라고 말했다는 일부 언론의 보도를 부인했다.

Sample Document

McDonald's slims down spuds

Fast-food chain to reduce certain types of fat in its french fries with new cooking oil.

NEW YORK (CNN/Money) - McDonald's Corp. is cutting the amount of "bad" fat in its french fries nearly in half, the fast-food chain said Tuesday as it moves to make all its fried menu items healthier.

But does that mean the popular shoestring fries won't taste the same? The company says no. "It's a win-win for our customers because they are getting the same great french-fry taste along with an even healthier nutrition profile," said Mike Roberts, president of McDonald's USA.

But others are not so sure. McDonald's will not specifically discuss the kind of oil it plans to use, but at least one nutrition expert says playing with the formula could mean a different taste.

Shares of Oak Brook, Ill.-based McDonald's (MCD: down $0.54 to $23.22, Research, Estimates) were lower Tuesday afternoon. It was unclear Tuesday whether competitors Burger King and Wendy's International (WEN: down $0.80 to $34.91, Research, Estimates) would follow suit. Neither company could immediately be reached for comment.

…

"Bag of Words"

14 × McDonalds

12 × fat

11 × fries

8 × new

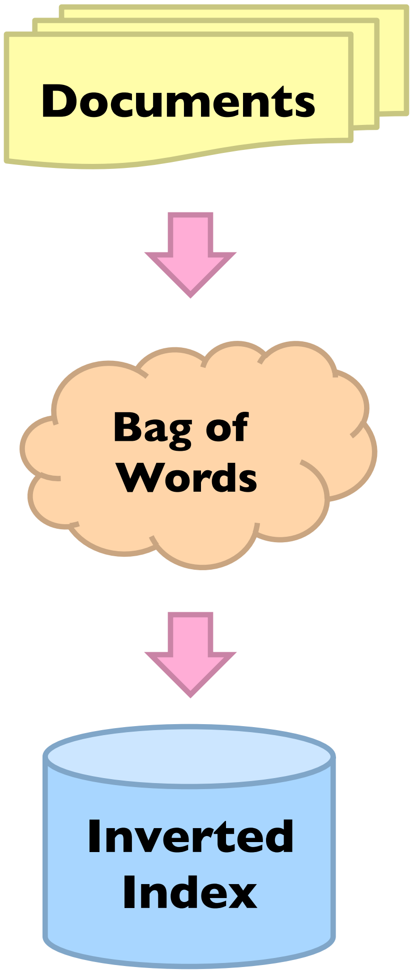
7 × french

6 × company, said, nutrition

5 × food, oil, percent, reduce, taste, Tuesday

…

Counting Words…



case folding, tokenization, stopword removal, stemming

~~syntax~~, ~~semantics~~, ~~word knowledge~~, etc.

Doc 1
one fish, two fish

Doc 2
red fish, blue fish

Doc 3
cat in the hat

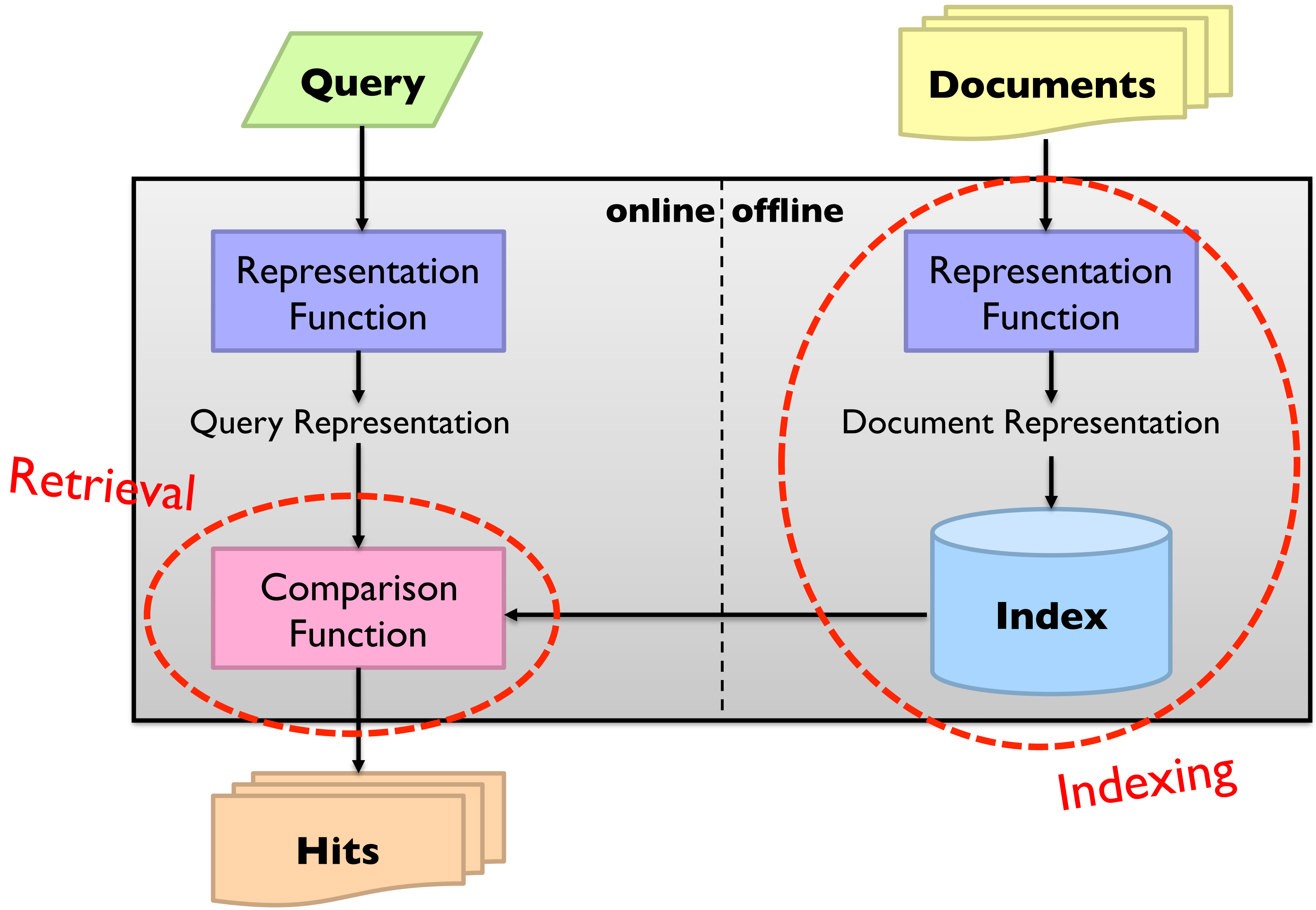
Doc 4
green eggs and ham

	1	2	3	4
blue		1		
cat			1	
egg				1
fish	1	1		
green				1
ham				1
hat			1	
one	1			
red		1		
two	1			

What goes in each cell?

- boolean
- count
- positions

Abstract IR Architecture



Doc 1
one fish, two fish

Doc 2
red fish, blue fish

Doc 3
cat in the hat

Doc 4
green eggs and ham

	1	2	3	4
blue		1		
cat			1	
egg				1
fish	1	1		
green				1
ham				1
hat			1	
one	1			
red		1		
two	1			

Indexing: building this structure

Retrieval: manipulating this structure

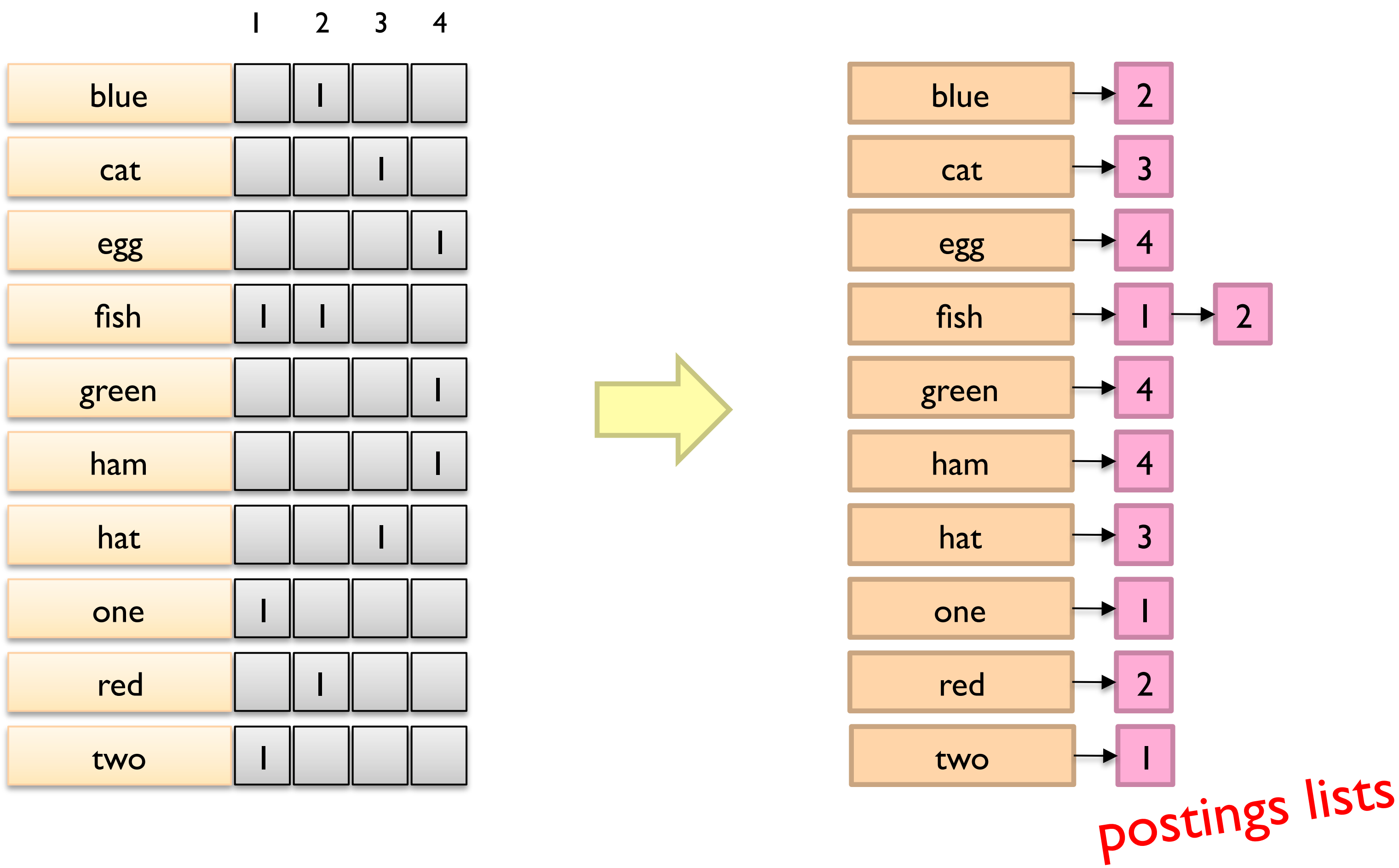
Where have we seen this before?

Doc 1
one fish, two fish

Doc 2
red fish, blue fish

Doc 3
cat in the hat

Doc 4
green eggs and ham



Indexing: Performance Analysis

- Fundamentally, a large sorting problem
  - Terms usually fit in memory
  - Postings usually don't
- How is it done on a single machine?
- How can it be done with MapReduce?
- First, let's characterize the problem size:
  - Size of vocabulary
  - Size of postings

Vocabulary Size: Heaps' Law

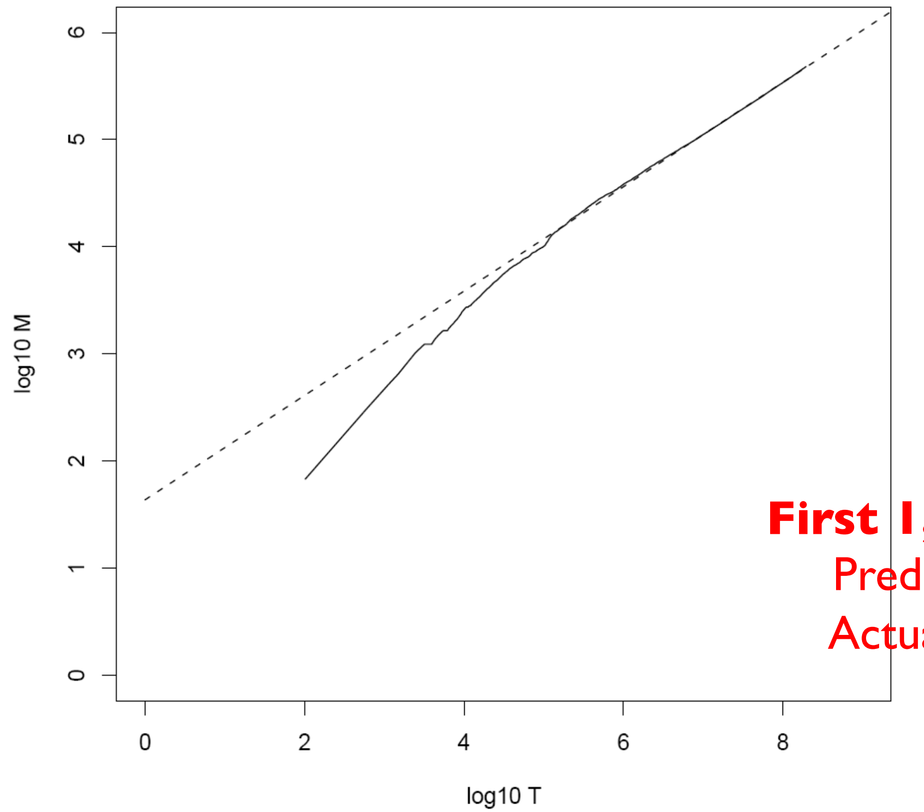
$$M = kT^b$$

M is vocabulary size
T is collection size (number of documents)
k and *b* are constants

Typically, *k* is between 30 and 100, *b* is between 0.4 and 0.6

- Heaps' Law: linear in log-log space
- Vocabulary size grows unbounded!

Heaps' Law for RCV1



k = 44
b = 0.49

First 1,000,020 terms:
Predicted = 38,323
Actual = 38,365

Reuters-RCV1 collection: 806,791 newswire documents (Aug 20, 1996-August 19, 1997)

Manning, Raghavan, Schütze, Introduction to Information Retrieval (2008)

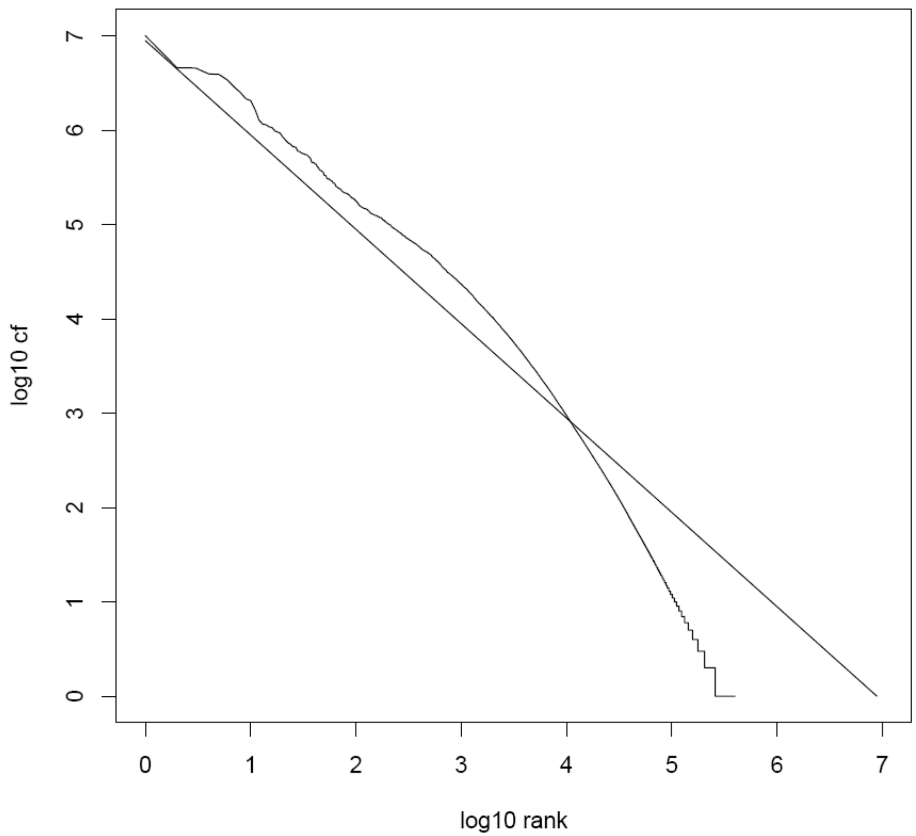
Postings Size: Zipf's Law

$$\mathrm{cf}_i = \frac{c}{i}$$

cf is the collection frequency of *i*-th common term
c is a constant

- Zipf's Law: (also) linear in log-log space
  - Specific case of Power Law distributions
- In other words:
  - A few elements occur very frequently
  - Many elements occur very infrequently

Zipf's Law for RCV1



Fit isn't that good…
but good enough!

Reuters-RCV1 collection: 806,791 newswire documents (Aug 20, 1996-August 19, 1997)

Manning, Raghavan, Schütze, Introduction to Information Retrieval (2008)

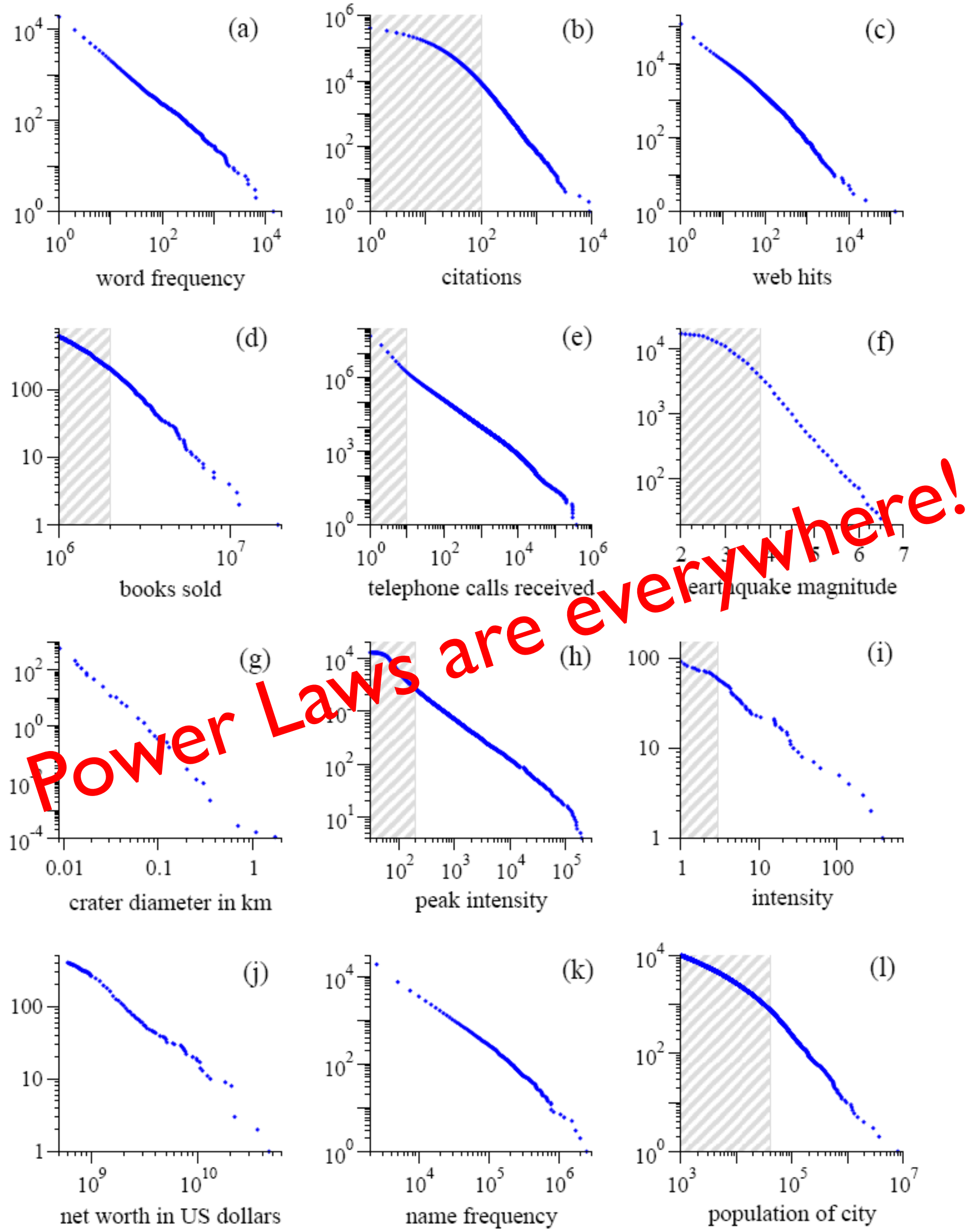
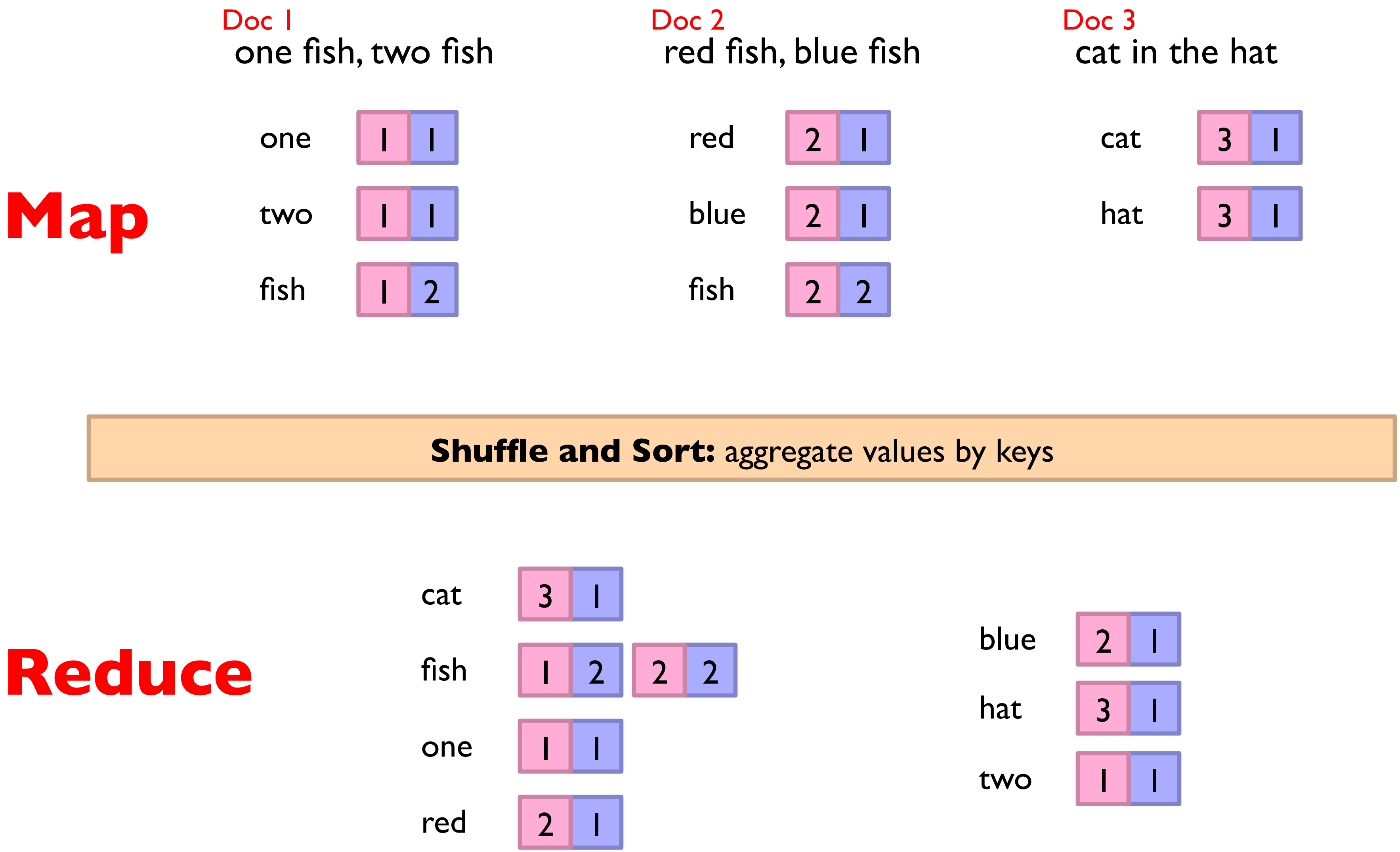


Figure from: Newman, M. E. J. (2005) “Power laws, Pareto distributions and Zipf's law.” Contemporary Physics 46:323–351.

MapReduce: Index Construction

- Map over all documents
  - Emit *term* as key, (*docno*, *tf)* as value
  - Emit other information as necessary (e.g., term position)
- Sort/shuffle: group postings by term
- Reduce
  - Gather and sort the postings (e.g., by *docno* or *tf*)
  - Write postings to disk
- MapReduce does all the heavy lifting!

Inverted Indexing with MapReduce



Inverted Indexing: Pseudo-Code

```
class Mapper
    method Map(docid n, doc d)
        H ← new AssociativeArray                 ▷ histogram to hold term frequencies
        for all term t ∈ doc d do     ▷ processes the doc, e.g., tokenization and stopword removal
            H{t} ← H{t} + 1
        for all term t ∈ H do
            Emit(term t, posting ⟨n, H{t}⟩)                        ▷ emits individual postings

class Reducer
    method Reduce(term t, postings [⟨n1, f1⟩ . . .])
        P ← new List
        for all ⟨n, f⟩ ∈ postings [⟨n1, f1⟩ . . .] do
            P.Append(⟨n, f⟩)                                      ▷ appends postings unsorted
        P.Sort()                                                    ▷ sorts for compression
        Emit(term t, postingsList P)
```

What's the problem?

Stay tuned…

Source: Wikipedia (Japanese rock garden)