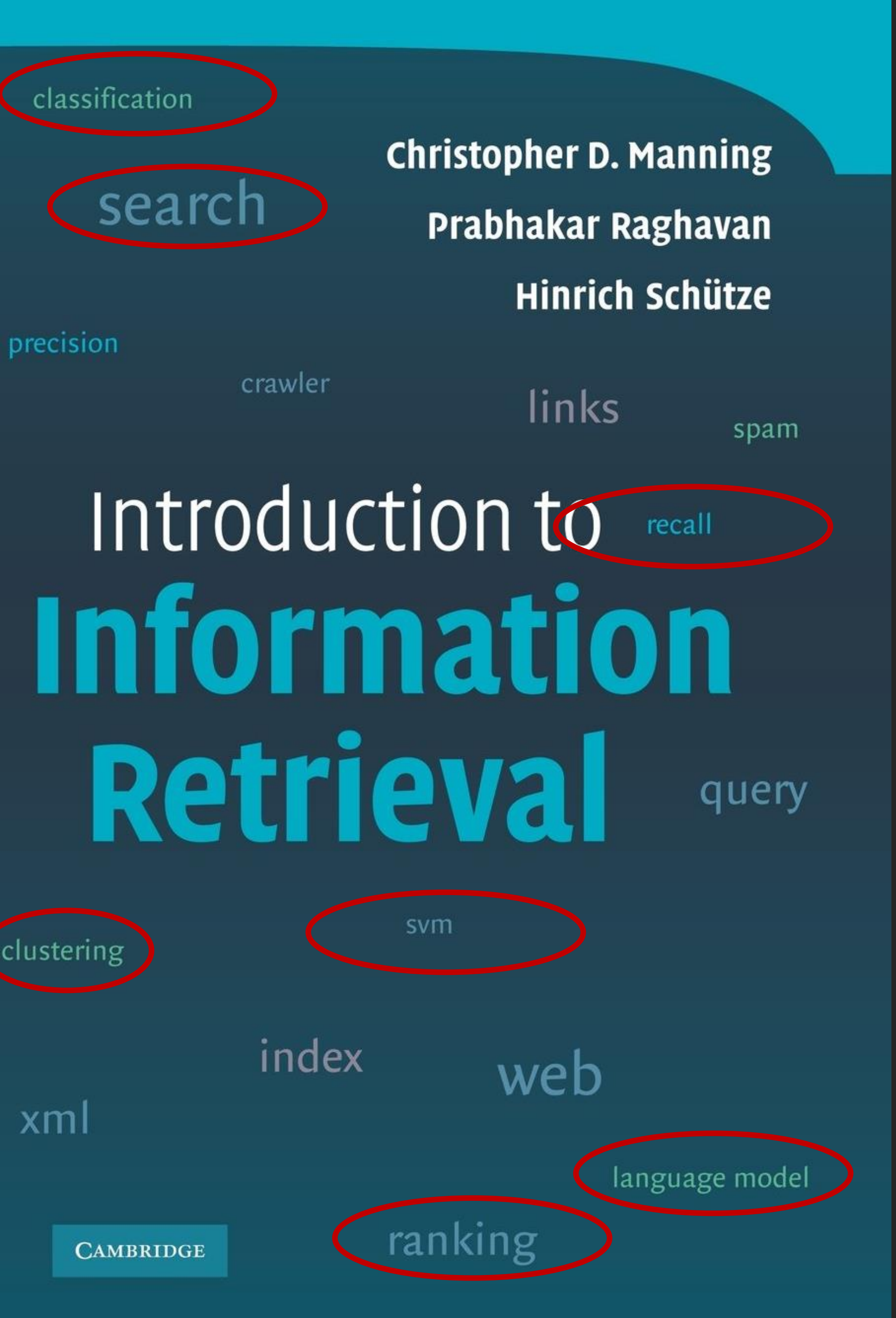
Information Retrieval and Question Answering

Nov 17. 2017

Content

- Information retrieval (IR)
    - Search
    - Rank
    - Learning to rank
    - Evaluate
- Question Answering (QA)
    - IR based QA
    - More



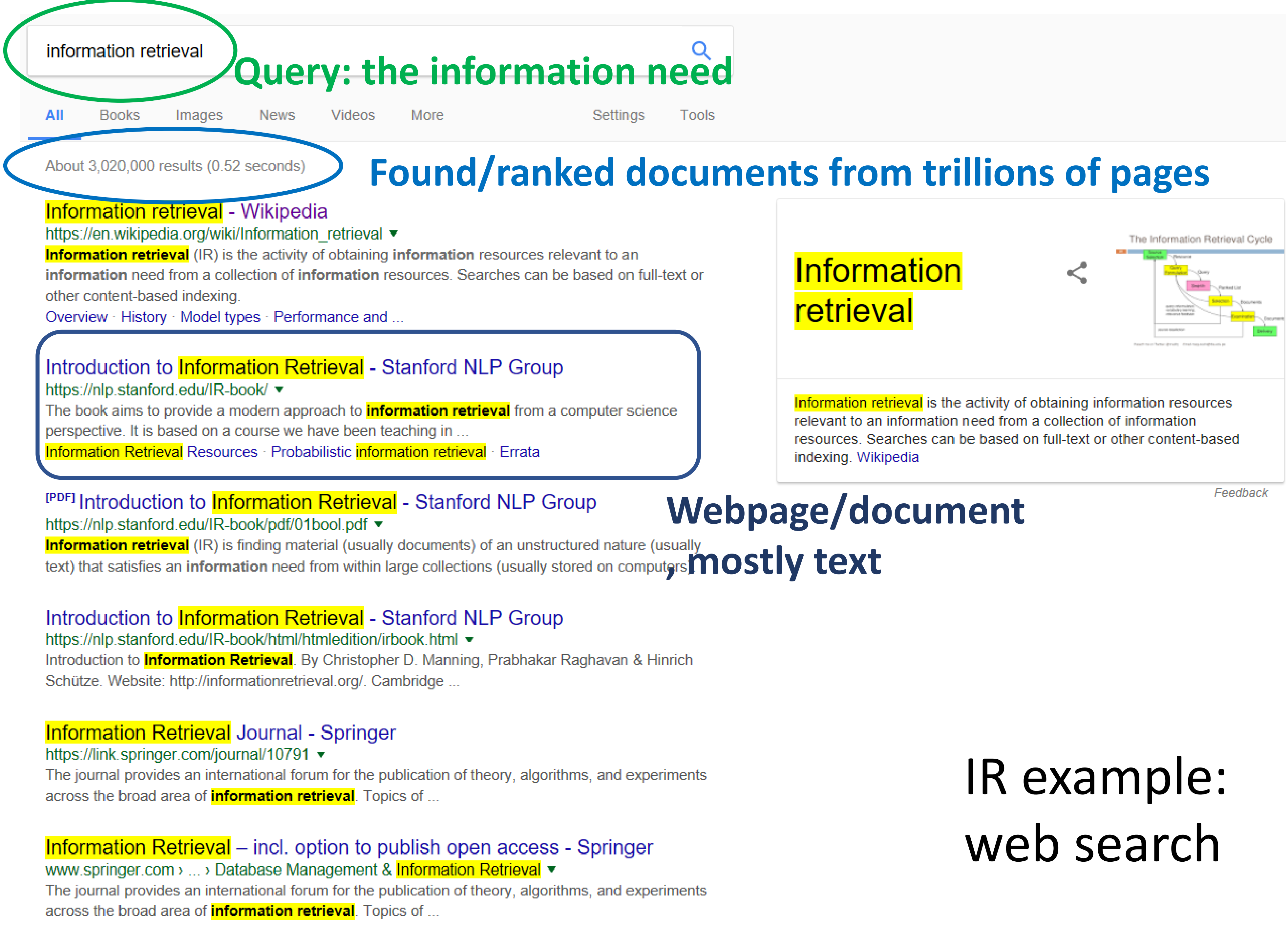
Information Retrieval

Model thinking in application problems

Definition of IR

- **Information retrieval** (**IR**) is finding material (usually documents) of an unstructured nature (usually text, contrary to database tables) that satisfies an information need from within large collections (usually stored on computers).
  - **Task**: finding something
  - **Target**: information need
  - **Nature**: unstructured data
  - **Evaluation**: satisfies

reference: chapter 1. *introduction to IR.*
Manning, Raghavan, Schutze



information retrieval

All Books Images News Videos More Settings Tools

About 3,020,000 results (0.52 seconds)

Information retrieval - Wikipedia
https://en.wikipedia.org/wiki/Information_retrieval ▾
Information retrieval (IR) is the activity of obtaining **information** resources relevant to an **information** need from a collection of **information** resources. Searches can be based on full-text or other content-based indexing.
Overview · History · Model types · Performance and ...

Introduction to Information Retrieval - Stanford NLP Group
https://nlp.stanford.edu/IR-book/ ▾
The book aims to provide a modern approach to **information retrieval** from a computer science perspective. It is based on a course we have been teaching in ...
Information Retrieval Resources · Probabilistic information retrieval · Errata

[PDF] Introduction to Information Retrieval - Stanford NLP Group
https://nlp.stanford.edu/IR-book/pdf/01bool.pdf ▾
Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an **information** need from within large collections (usually stored on computers).

Introduction to Information Retrieval - Stanford NLP Group
https://nlp.stanford.edu/IR-book/html/htmledition/irbook.html ▾
Introduction to **Information Retrieval**. By Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze. Website: http://informationretrieval.org/. Cambridge ...

Information Retrieval Journal - Springer
https://link.springer.com/journal/10791 ▾
The journal provides an international forum for the publication of theory, algorithms, and experiments across the broad area of **information retrieval**. Topics of ...

Information Retrieval – incl. option to publish open access - Springer
www.springer.com › ... › Database Management & Information Retrieval ▾
The journal provides an international forum for the publication of theory, algorithms, and experiments across the broad area of **information retrieval**. Topics of ...

Information retrieval

Information retrieval is the activity of obtaining information resources relevant to an information need from a collection of information resources. Searches can be based on full-text or other content-based indexing. Wikipedia

Feedback

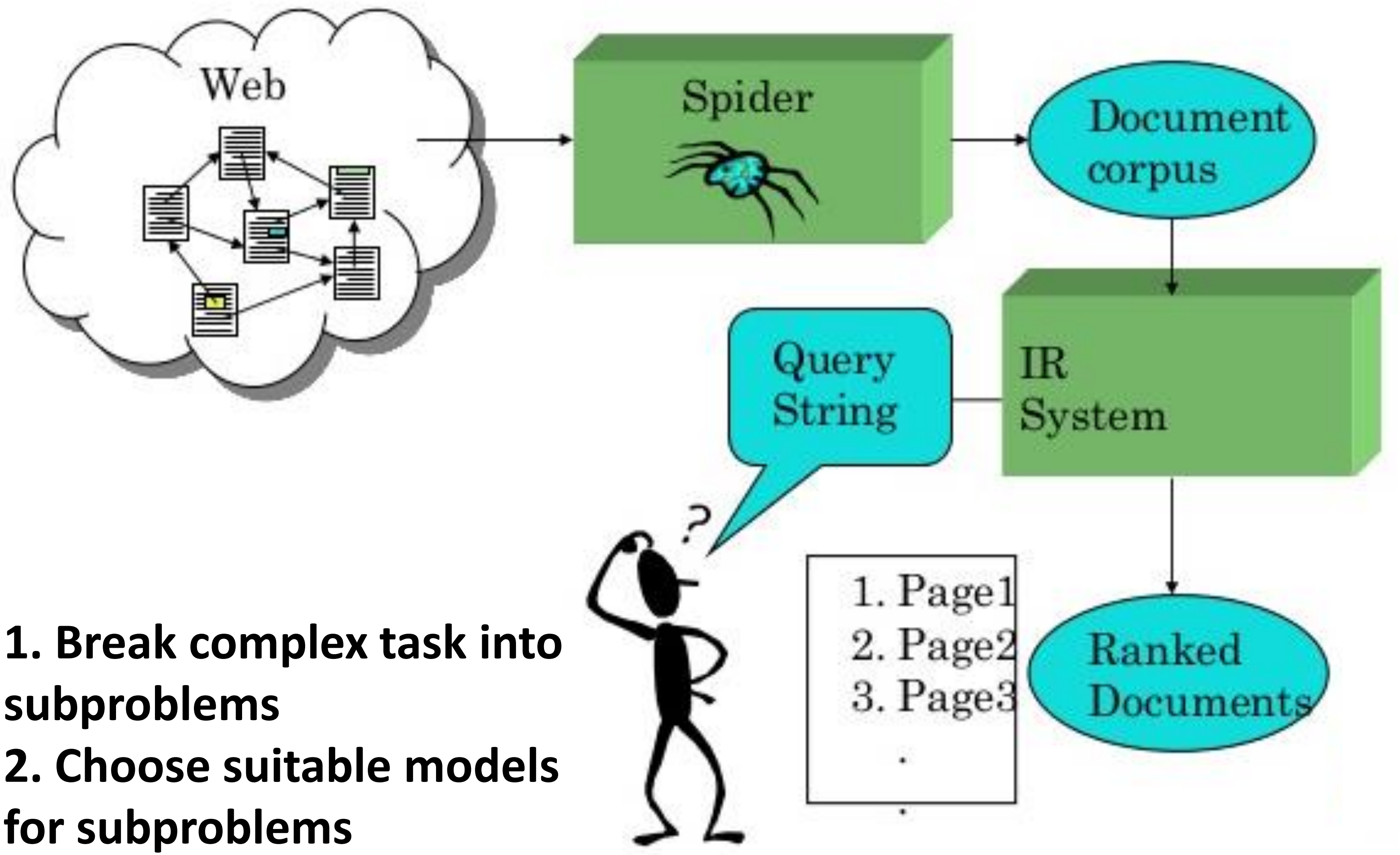
Query: the information need

Found/ranked documents from trillions of pages

Webpage/document
,mostly text

IR example:
web search

Model thinking



index

Search for relevant document

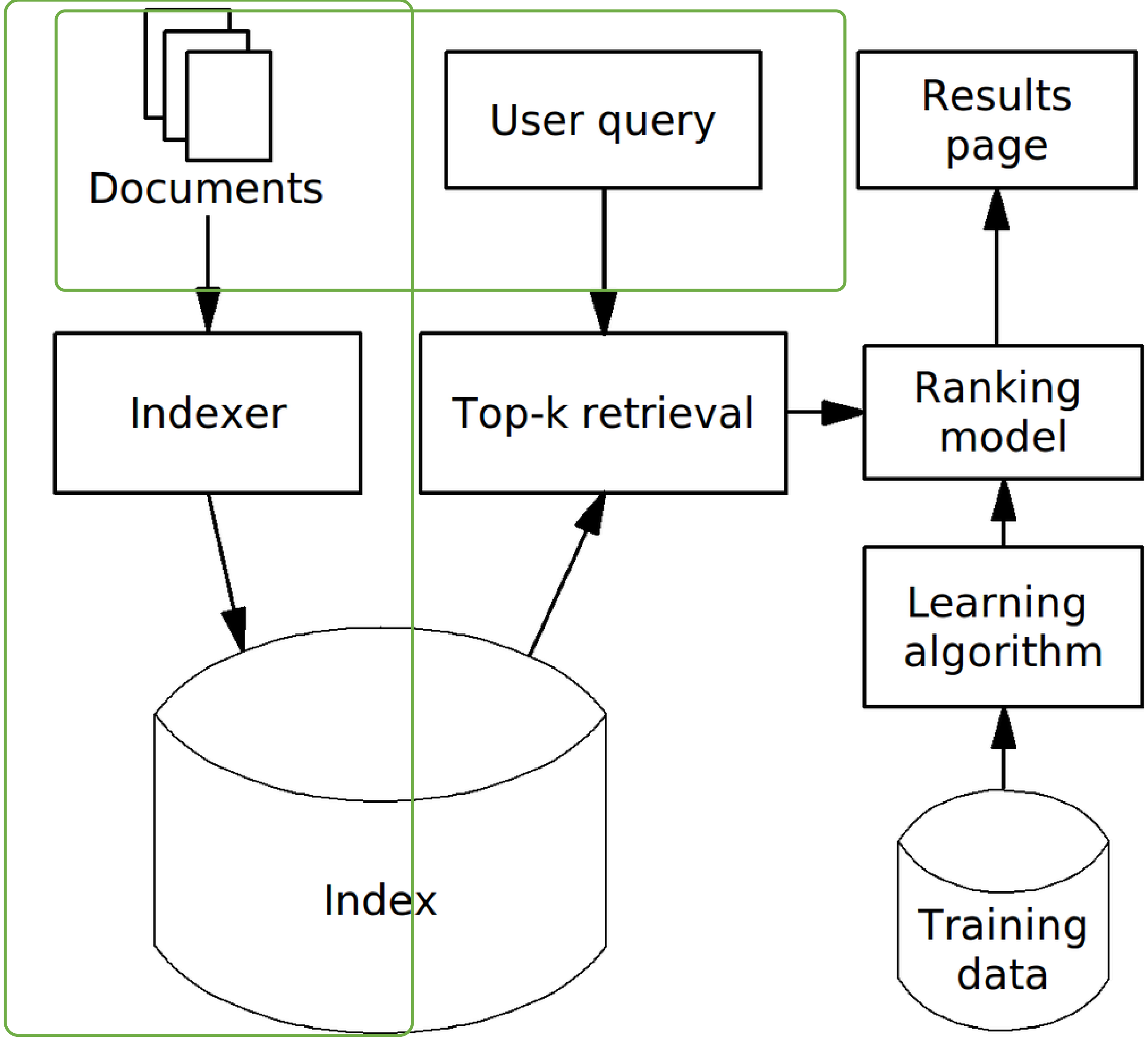
Select relevant document

1. Break complex task into subproblems
2. Choose suitable models for subproblems

Source of figure

Search

Boolean retrieval and inverted index

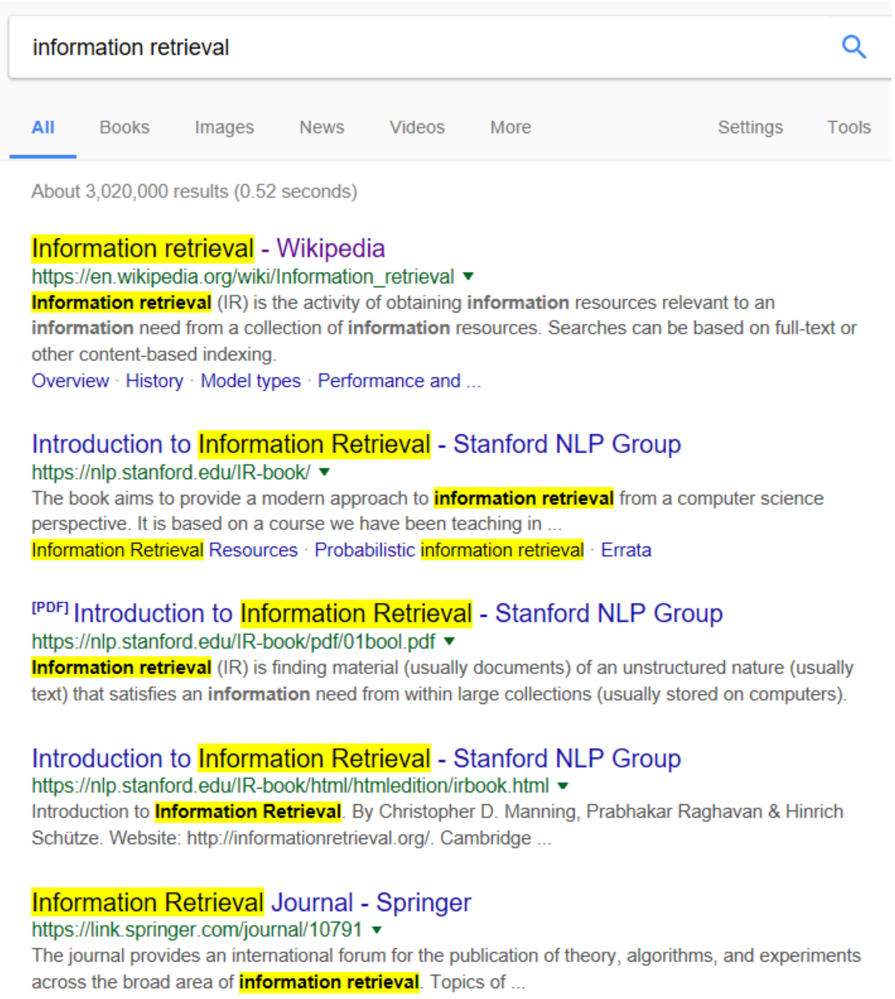


Query and retrieved documents

- **The ad hoc retrieval problem**: Given a user information need and a collection of documents, the IR system **determines how well the documents satisfy the query** and returns a subset of **relevant documents** to the user.
- **Relevance** between Query (Q) and document (D) is a tricky notion:
  - Will the user like it /click it?
  - Will it help the users achieve a task, find information they need?
  - Is the result novel (not redundant)?
  - Does Q and D share similar meaning?
  - Does Q and D cover the same topic?

Quantify the relevance

- Key idea: relevant items are similar
  - Items using similar **vocabulary** are likely to share similar meaning, topic and be relevant
  - Similar documents are likely to be relevant to same query



information retrieval

About 3,020,000 results (0.52 seconds)

Information retrieval - Wikipedia
https://en.wikipedia.org/wiki/Information_retrieval ▾
Information retrieval (IR) is the activity of obtaining **information** resources relevant to an **information** need from a collection of **information** resources. Searches can be based on full-text or other content-based indexing.
Overview · History · Model types · Performance and ...

Introduction to Information Retrieval - Stanford NLP Group
https://nlp.stanford.edu/IR-book/ ▾
The book aims to provide a modern approach to **information retrieval** from a computer science perspective. It is based on a course we have been teaching in ...
Information Retrieval Resources · Probabilistic information retrieval · Errata

[PDF] Introduction to Information Retrieval - Stanford NLP Group
https://nlp.stanford.edu/IR-book/pdf/01bool.pdf ▾
Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an **information** need from within large collections (usually stored on computers).

Introduction to Information Retrieval - Stanford NLP Group
https://nlp.stanford.edu/IR-book/html/htmledition/irbook.html ▾
Introduction to **Information Retrieval**. By Christopher D. Manning, Prabhakar Raghavan & Hinrich Schütze. Website: http://informationretrieval.org/. Cambridge ...

Information Retrieval Journal - Springer
https://link.springer.com/journal/10791 ▾
The journal provides an international forum for the publication of theory, algorithms, and experiments across the broad area of **information retrieval**. Topics of ...

Query document via keywords

- Given Shakespeare’s Collected Works, we want to determine which plays of Shakespeare contain the words
  - **Brutus** AND **Caesar** AND NOT **Calpurnia**

1. For every play (document), does it contain “Brutus AND Caesar AND NOT Calpurnia”?
2. For every word from “Brutus AND Caesar AND NOT Calpurnia”, which document contains it?

Term document matrix

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	…
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
…							

A term document incident matrix. Matrix element (t,d) is 1 iff the play (document) in column **d** contains term **t**

Boolean retrieval

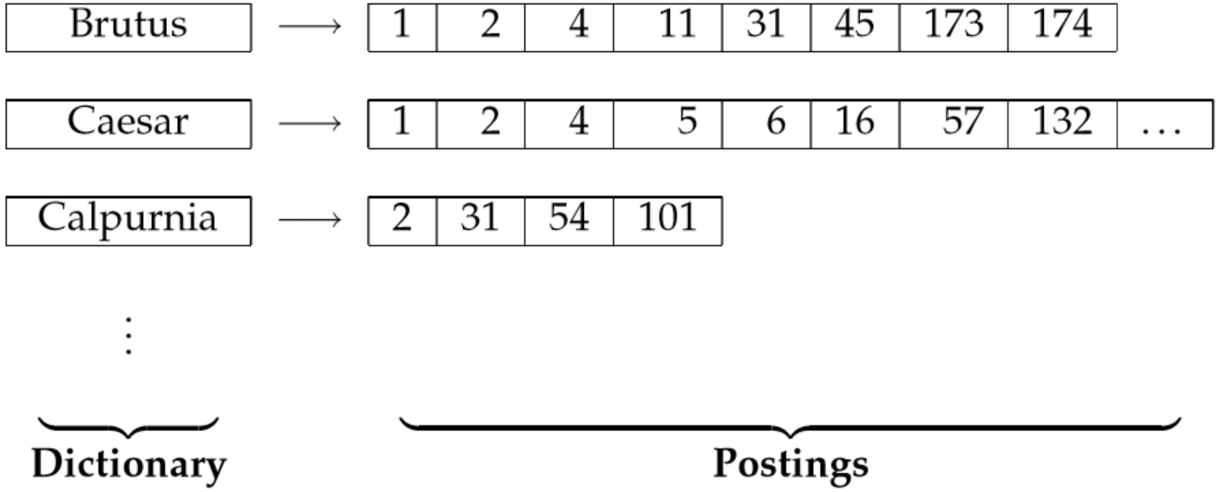
	Antony and Cleopatra	Julius Caesar	The Tempest	**Hamlet**	Othello	Macbeth	…
Antony	1	1	0	0	0	1	
Brutus	**1**	**1**	**0**	**1**	**0**	**0**	
Caesar	**1**	**1**	**0**	**1**	**1**	**1**	
Calpurnia	**0**	**1**	**0**	**0**	**0**	**0**	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
…							

Query: Caesar AND Brutus NOT Calpurnia

110100 AND 110111 AND 101111 = 100100, found ‘Antony and Cleopatra’ and ‘Hamlet’

Record &query using inverted index

	Antony and Cleopatra	Julius Caesar	The Tempest	**Hamlet**	Othello	Macbeth	…
Brutus	**1**	**1**	**0**	**1**	**0**	**0**	
Caesar	**1**	**1**	**0**	**1**	**1**	**1**	
Calpurnia	**0**	**1**	**0**	**0**	**0**	**0**	



Brutus	→	1	2	4	11	31	45	173	174
Caesar	→	1	2	4	5	6	16	57	132	. . .
Calpurnia	→	2	31	54	101				

⋮

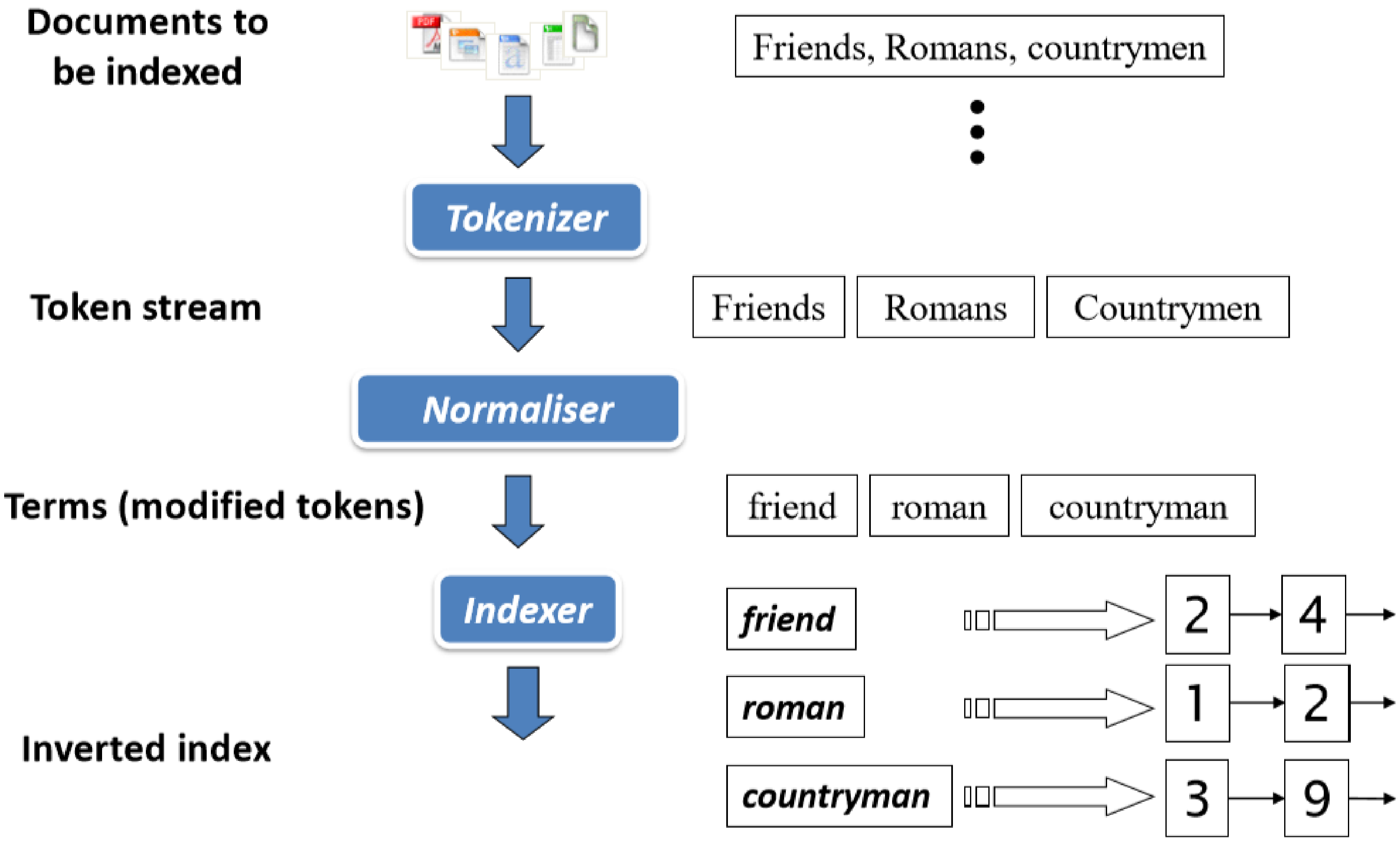
Dictionary

Postings

Query: Caesar AND Brutus NOT Calpurnia

Return: [1, 2, 4] – [2] = [1, 4]

Building invert index



Documents to be indexed

Friends, Romans, countrymen

Tokenizer

Token stream

Friends | Romans | Countrymen

Normaliser

Terms (modified tokens)

friend | roman | countryman

Indexer

Inverted index

friend → 2 → 4

roman → 1 → 2

countryman → 3 → 9

Page source

Step 1: Term Sequence

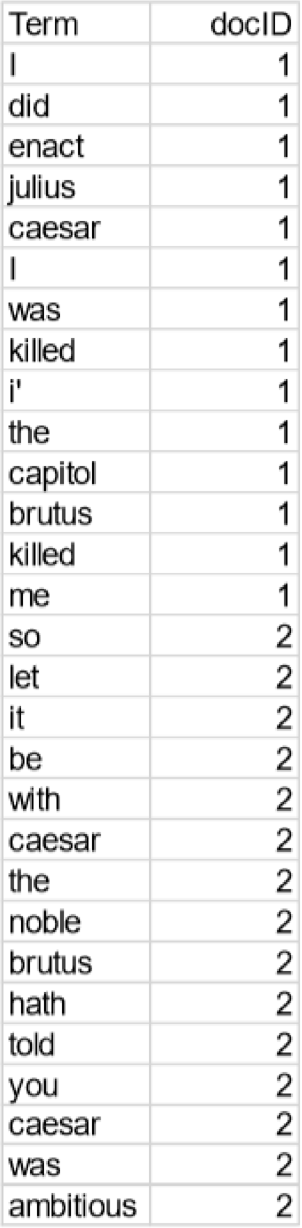
Doc 1

I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Sequence of (term, Doc ID) pairs



Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Page source

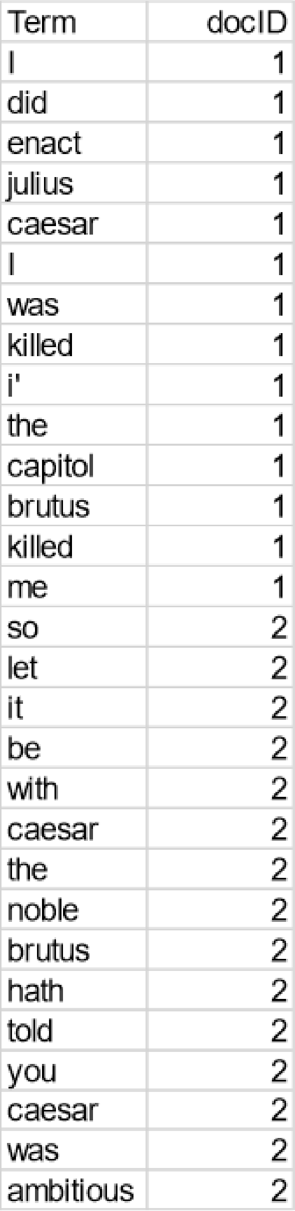
Step 2: Sorting

- **Sort by:**

  1) Term

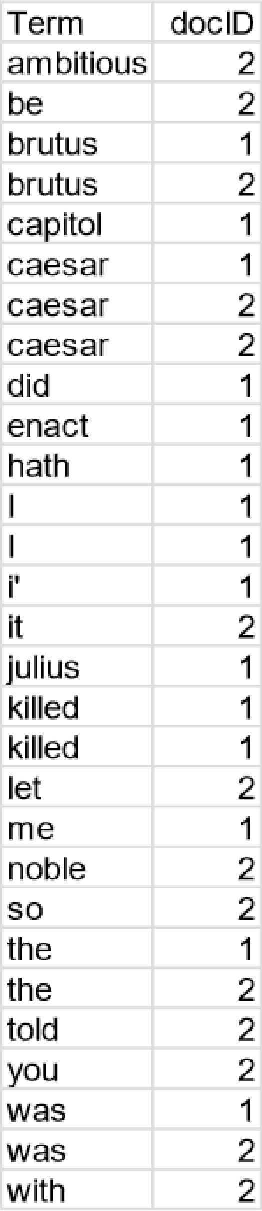
  then

  2) Doc ID



Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Sorting →

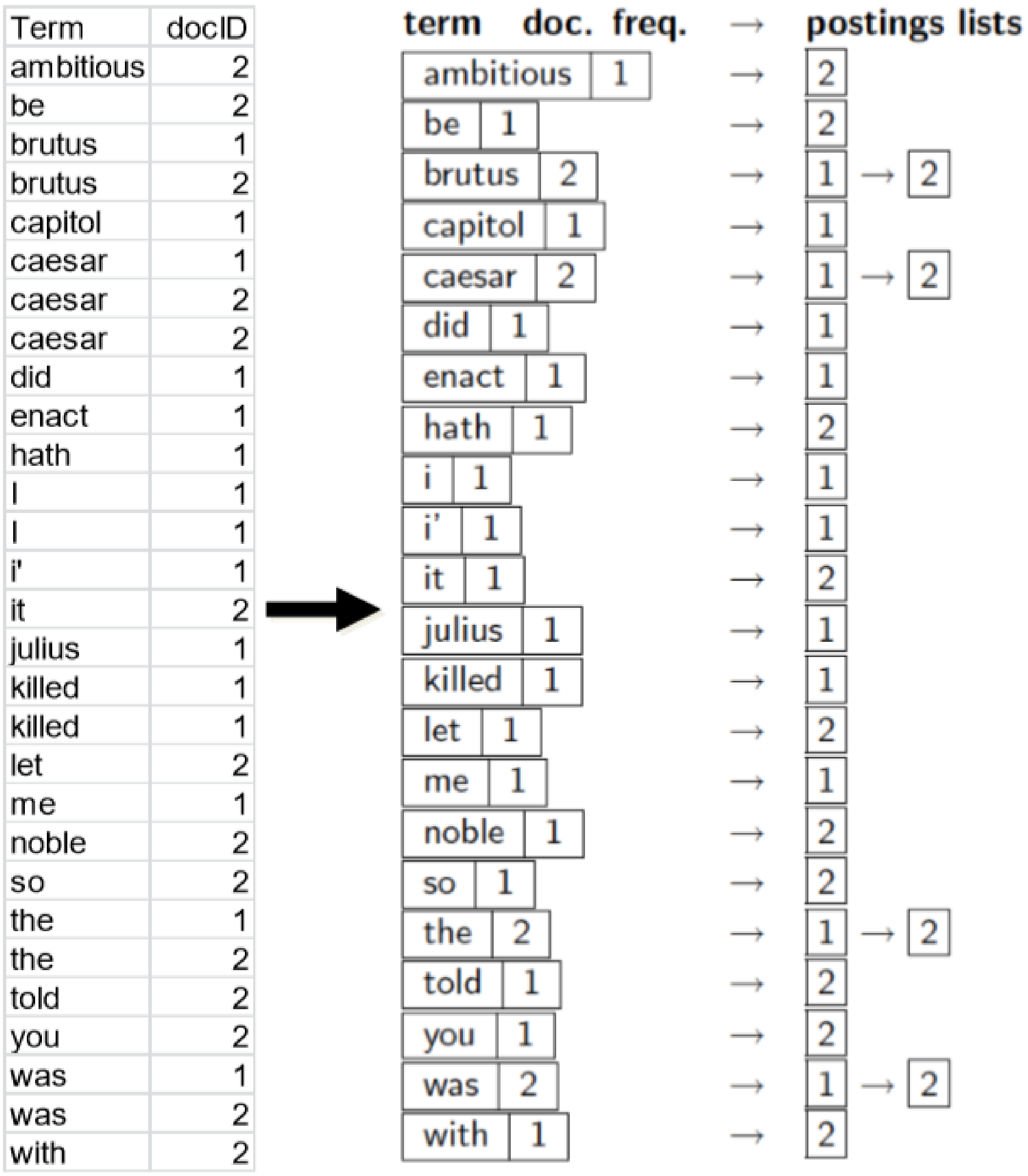


Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Page source

Step 3: Posting

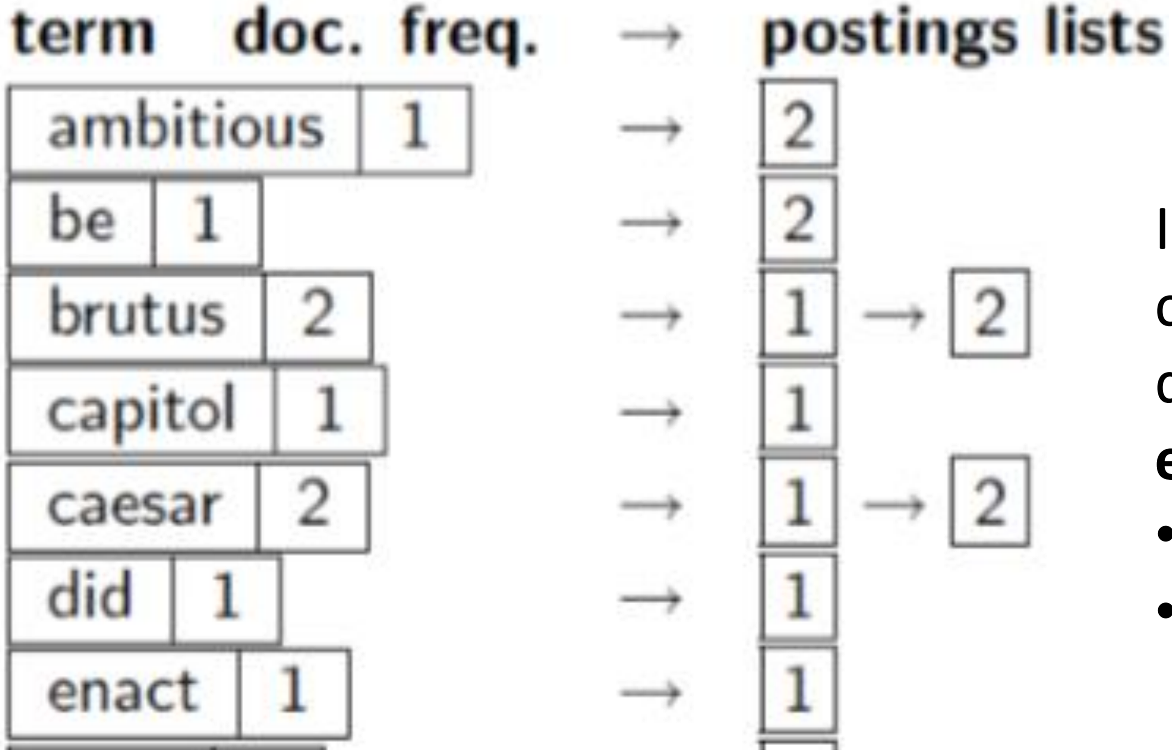
1. Multiple term entries in a single document are merged
2. Split into Dictionary and Postings
3. Doc. Frequency (*df*) information is added



Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

Page source



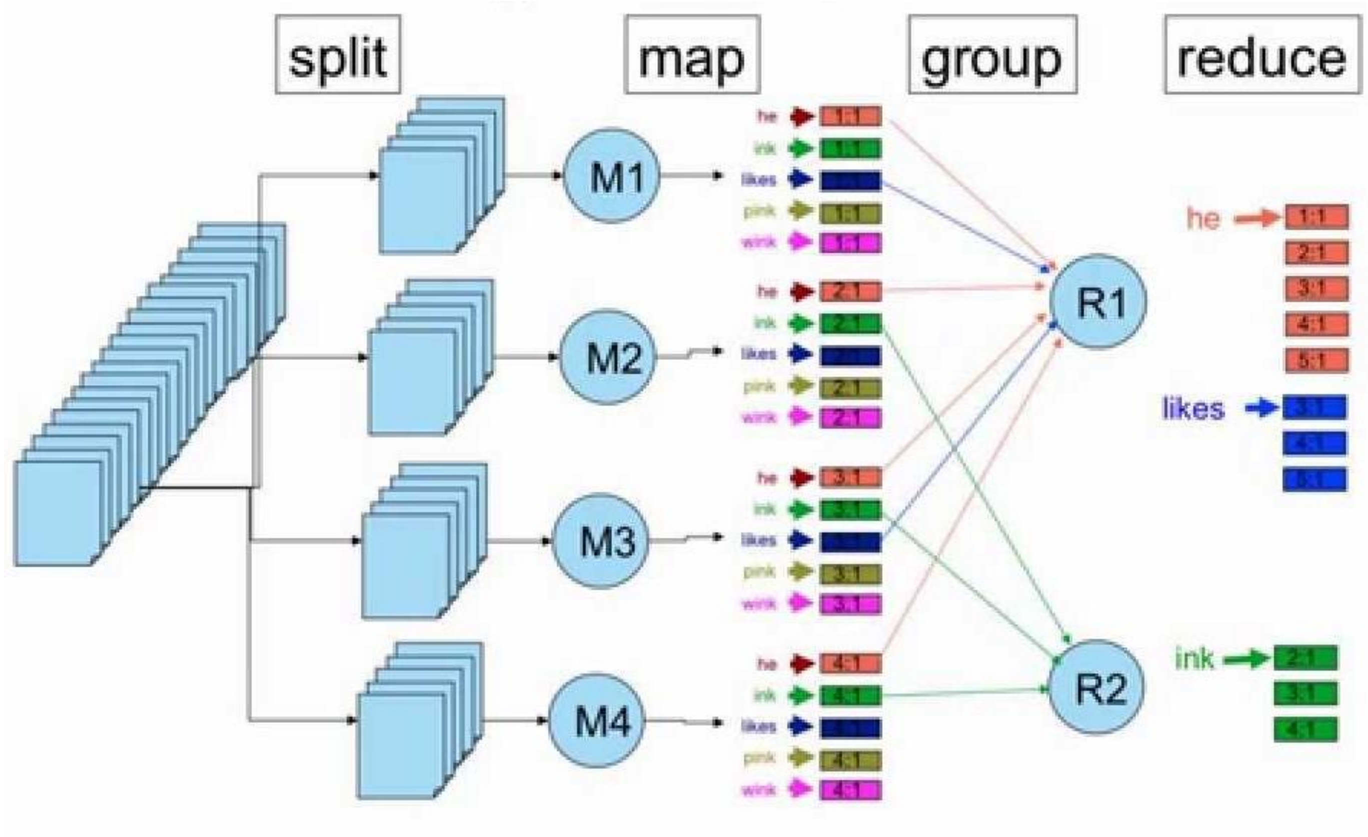
term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1

In practice, posting lists of each term could contain more information than doc id

e.g.

- Brutus --> [1, 2]
- Brutus --> [1:3, 2:5]   **# how many times term appear in doc?**
- Brutus --> [1:[10, 111, 234] , 2:[45, 200, 211, 340, 789]]   **# positions term appear in doc?**

Exercise: Large scale indexing

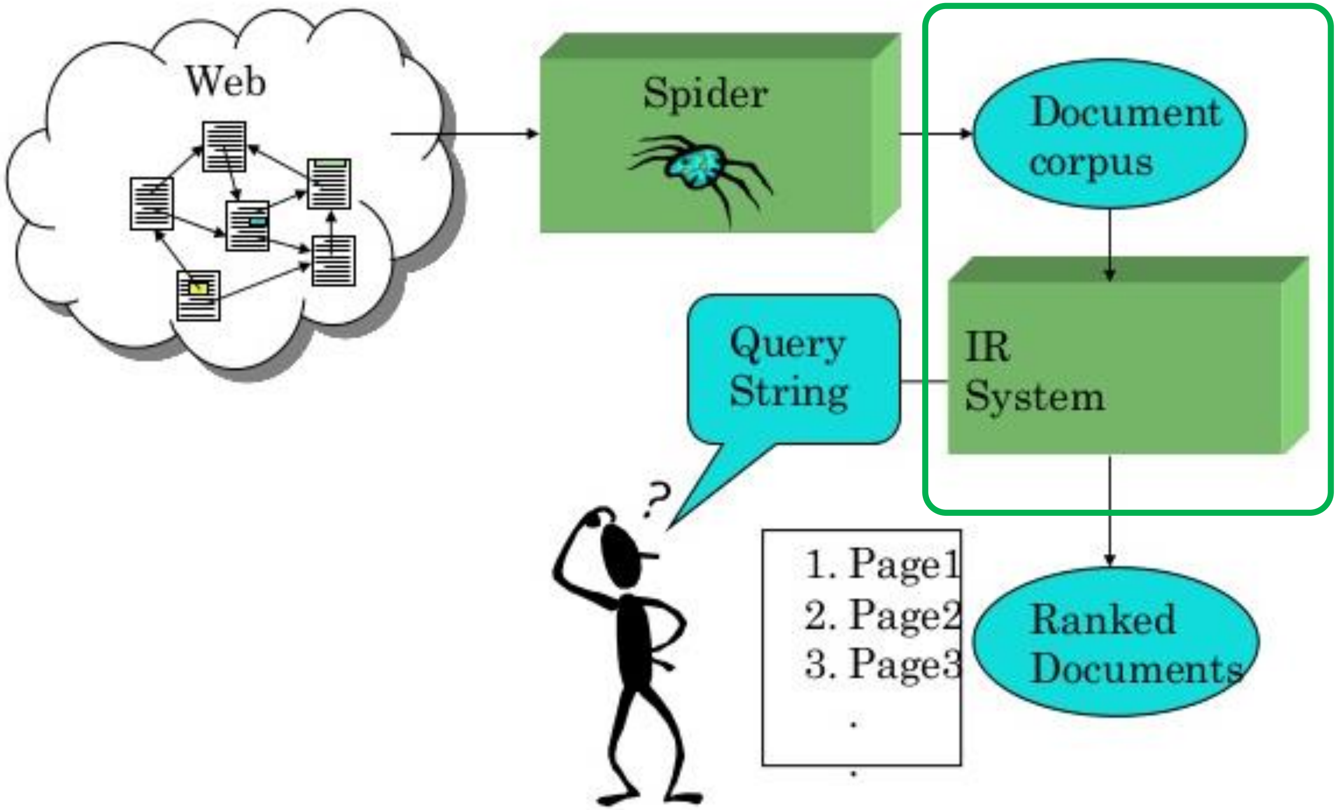


Copyright © 2013 Victor Lavrenko

Figure by Victor Lavrenko

Further reading: Building Software Systems at Google and Lessons Learned

Quick summary



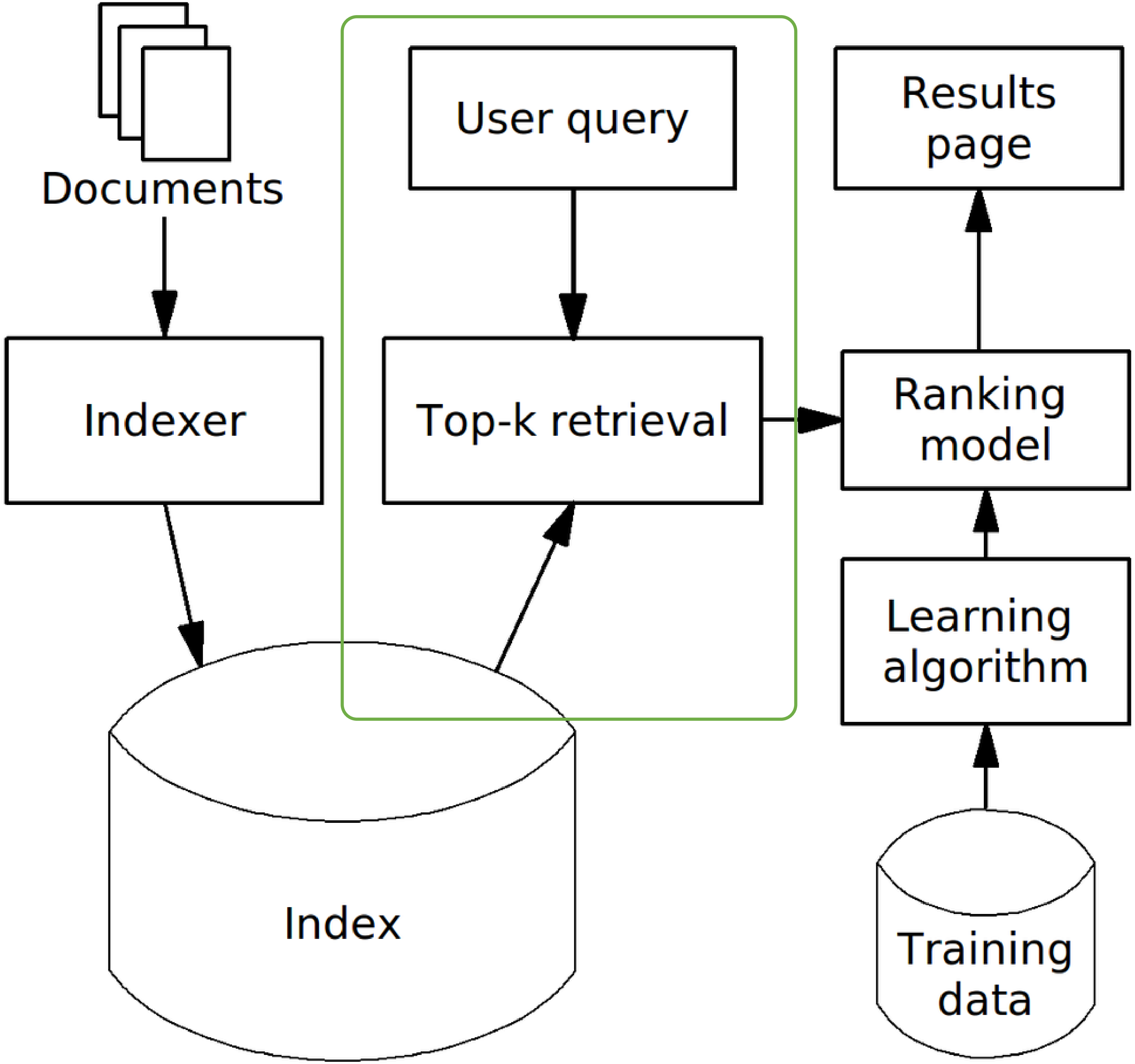
Built data structure suitable for documents search

Models we used:

- Bag of words (BOW)
- Term document representation of dataset
- Adjacent list for efficient data storage

Rank

Choosing the most relevant documents



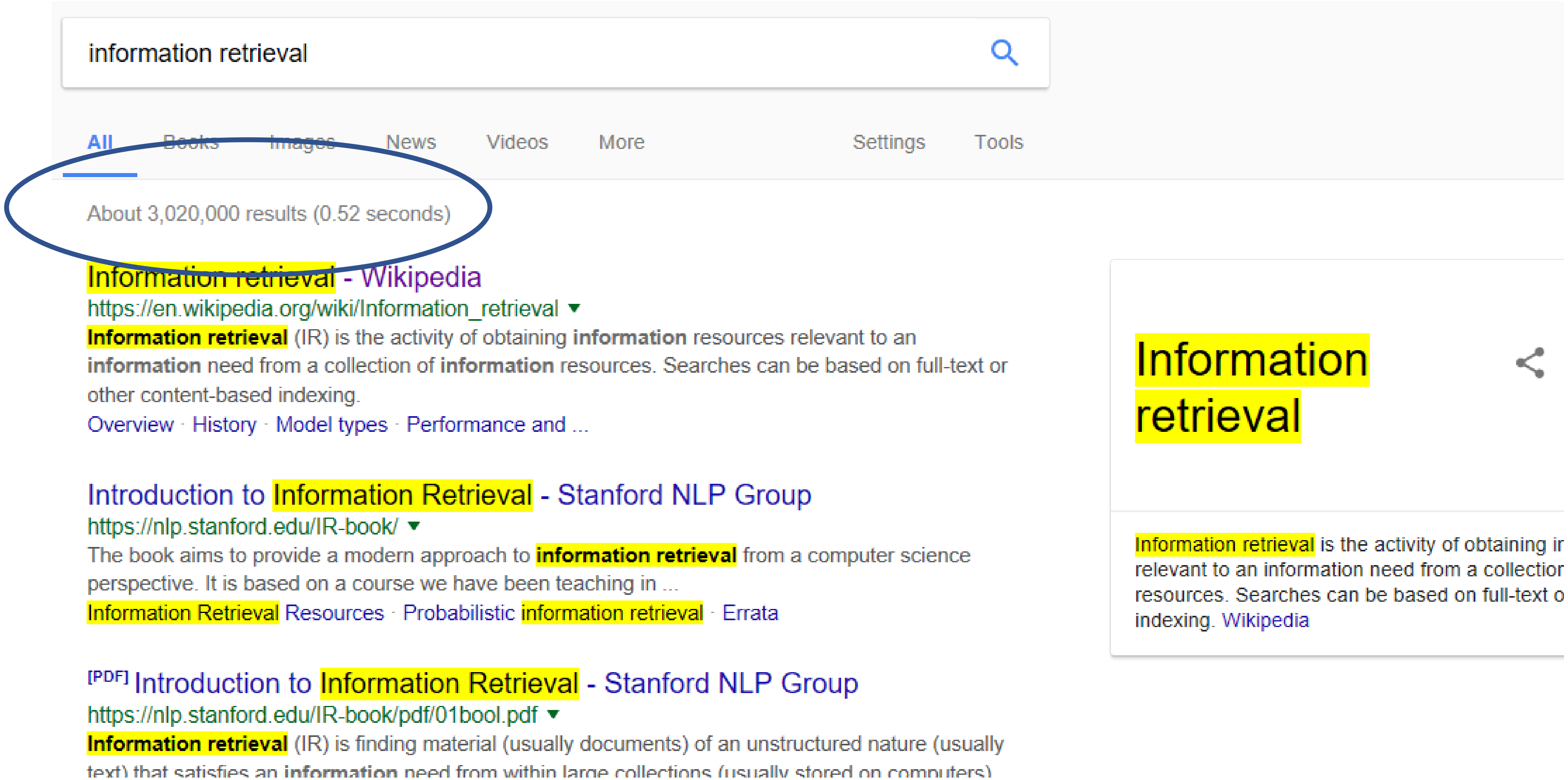
Recap: BR rates relevant documents equally

	Antony and Cleopatra	Julius Caesar	The Tempest	**Hamlet**	Othello	Macbeth	…
Antony	1	1	0	0	0	1	
Brutus	**1**	**1**	**0**	**1**	**0**	**0**	
Caesar	**1**	**1**	**0**	**1**	**1**	**1**	
Calpurnia	**0**	**1**	**0**	**0**	**0**	**0**	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
…							

Query: Caesar AND Brutus NOT Calpurnia

‘Antony and Cleopatra’ and ‘Hamlet’ both have similarity score = 3

Recap: BR may found too many relevant docs



information retrieval

About 3,020,000 results (0.52 seconds)

Information retrieval - Wikipedia
https://en.wikipedia.org/wiki/Information_retrieval ▾
Information retrieval (IR) is the activity of obtaining **information** resources relevant to an **information** need from a collection of **information** resources. Searches can be based on full-text or other content-based indexing.
Overview · History · Model types · Performance and ...

Introduction to Information Retrieval - Stanford NLP Group
https://nlp.stanford.edu/IR-book/ ▾
The book aims to provide a modern approach to **information retrieval** from a computer science perspective. It is based on a course we have been teaching in ...
Information Retrieval Resources · Probabilistic information retrieval · Errata

[PDF] Introduction to Information Retrieval - Stanford NLP Group
https://nlp.stanford.edu/IR-book/pdf/01bool.pdf ▾
Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an **information** need from within large collections (usually stored on computers)

Information retrieval

Information retrieval is the activity of obtaining ir
relevant to an information need from a collectior
resources. Searches can be based on full-text o
indexing. Wikipedia

Ranking method is core of modern IR system

- In what order do we present documents to the user?
- We want the “best” document to be first, second best second, etc…
- Idea: rank by some quantitative sore that measure relevance between query and documents

Vector Space Model (VSM)

- Based on BOW representation of items, we use these notations:
  - Q, D*:* vector representation of *Query, Document*
  - V*:* total number of words in vocabulary
  - Q_w , D_w*:* coordinate of Q, D along dimension w
  - e.g.
    - Each D vector is one play (doc)
    - Elements $\{D_w\}$ are either 0 or 1
  - $S(Q, D)$: score of relevance between *Q, D*
  - ***Problem: design a score function $\boldsymbol{S(Q, D)}$ s.t. the more document D is relevant to query Q, the higher number $\boldsymbol{S(Q, D)}$ returns***

Term (word) weighting

- Observation 1: presence / absence most important
  - D_w=1 if word present, 0 otherwise
  - Document = binary vector = set

	Antony and Cleopatra	Hamlet	Brutus AND Caesar
Antony	1	0	0
Brutus	**1**	**1**	**1**
Caesar	**1**	**1**	**1**
Calpurnia	0	0	0
Cleopatra	1	0	0

$$S(Q, D) = \sum_{w} \delta(w \in Q)\delta(w \in D)$$

Term (word) weighting

- Observation 2: key words tend to be repeated in a doc
  - $tf_{w,d}$, term frequency = number of times w occurred in D

$$S(Q,D) = \sum_{w} tf_{w,Q} tf_{w,D}$$

- E.g. Q = “machine learning”
  - Doc1: {machine: 10, learning:12}
  - Doc2: {machine: 2, learning:1}
  - Doc1 (score = 22) might be more relevant to the query than Doc2 (score = 3)

Term (word) weighting

- Observation 3: BOW biased toward long document
  - Long docs -> higher tf, spurious word occurrences
  - Normalized by document length |D|

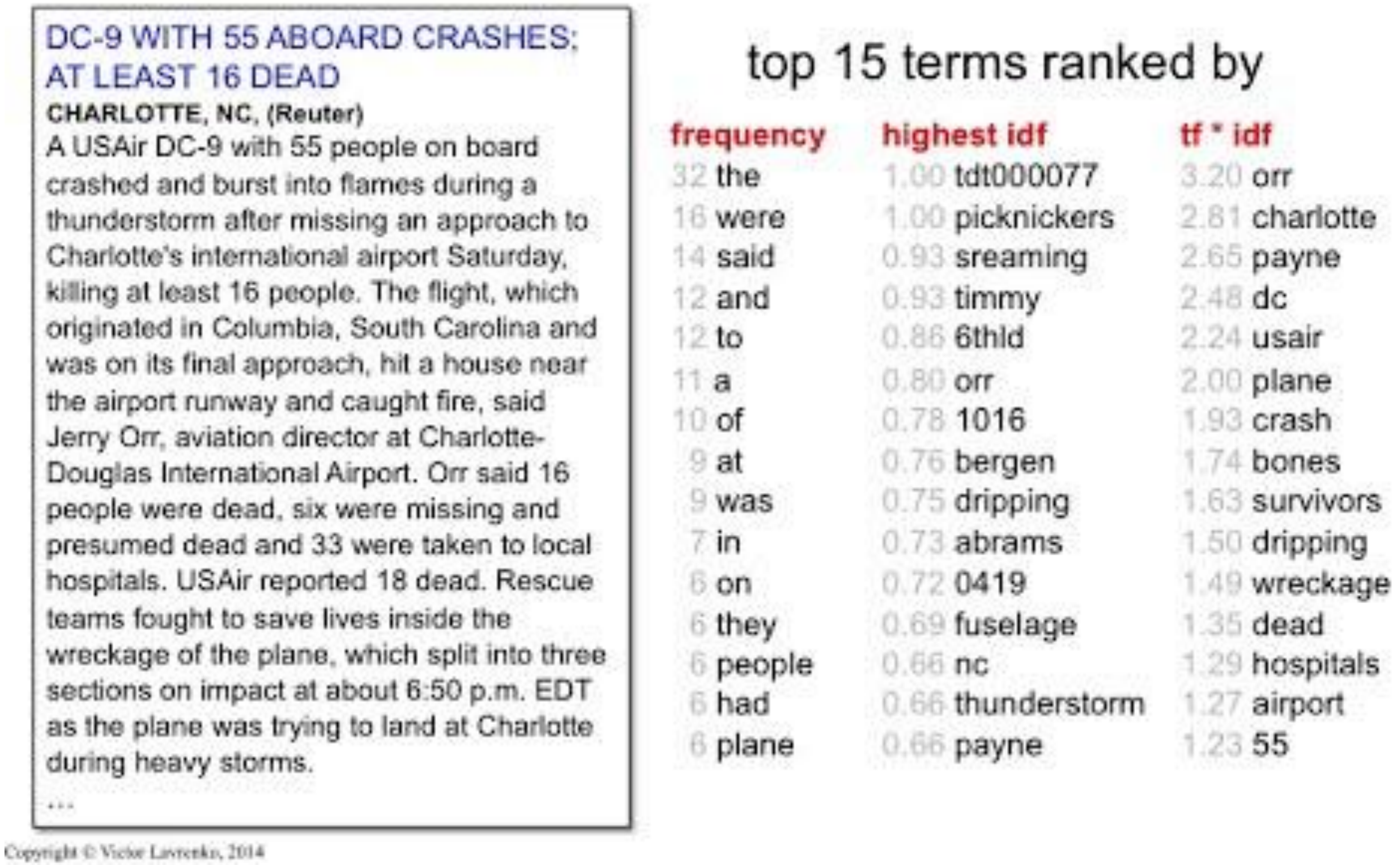
$$S(Q, D) = \sum_{w} tf_{w,Q} \frac{tf_{w,D}}{|D|}$$

- E.g. Q = “Introduction Logistic Regression”
  - Doc1 “Logistic Regression and Max Entropy Principle”
    - 1000 words, {Logistic: 20, Regression: 21}
  - Doc2 “Introduction to Information Retrieval”
    - 50000 words, {Logistic: 100, Regression: 150}

Term (word) weighting

- Observation 4: rare words carry more meaning
  - Cryogenic, aardvark, jacquard … proper nouns, topical content, important
  - Said, went, of, the big … linguistic glue, less useful for IR
- Give more weight to rare words: $\log \frac{|C|}{df_w}$
  - $|C|$, number of documents in collection (corpus)
  - df_w, number of documents containing word w
- Inverse document frequency (idf)
  - Very effective heuristic for picking out important words
  - Sometimes *idf* used on the query weights
  - Logarithm: to put *idf* on the same scale as the tf component

tf-idf selects informative terms



DC-9 WITH 55 ABOARD CRASHES; AT LEAST 16 DEAD

CHARLOTTE, NC, (Reuter)

A USAir DC-9 with 55 people on board crashed and burst into flames during a thunderstorm after missing an approach to Charlotte's international airport Saturday, killing at least 16 people. The flight, which originated in Columbia, South Carolina and was on its final approach, hit a house near the airport runway and caught fire, said Jerry Orr, aviation director at Charlotte-Douglas International Airport. Orr said 16 people were dead, six were missing and presumed dead and 33 were taken to local hospitals. USAir reported 18 dead. Rescue teams fought to save lives inside the wreckage of the plane, which split into three sections on impact at about 6:50 p.m. EDT as the plane was trying to land at Charlotte during heavy storms.

...

Copyright © Victor Lavrenko, 2014

top 15 terms ranked by

frequency	highest idf	tf * idf
32 the	1.00 tdt000077	3.20 orr
16 were	1.00 picknickers	2.81 charlotte
14 said	0.93 sreaming	2.65 payne
12 and	0.93 timmy	2.48 dc
12 to	0.86 6thld	2.24 usair
11 a	0.80 orr	2.00 plane
10 of	0.78 1016	1.93 crash
9 at	0.76 bergen	1.74 bones
9 was	0.75 dripping	1.63 survivors
7 in	0.73 abrams	1.50 dripping
6 on	0.72 0419	1.49 wreckage
6 they	0.69 fuselage	1.35 dead
6 people	0.66 nc	1.29 hospitals
6 had	0.66 thunderstorm	1.27 airport
6 plane	0.66 payne	1.23 55

From tutorial by Victor Lavrenko

Term (word) weighting

- *tf.idf* weighted sum as similarity measure

$$S(Q,D)=\sum_{w}\underbrace{tf_{w,Q}}_{\textbf{Observation 2}}\frac{tf_{w,D}}{\underbrace{|D|}_{\textbf{Observation 3}}}\log\frac{|C|}{\underbrace{df_{w}}_{\textbf{Observation 4}}}$$

Frequency normalization

- Observation 5:
  - Q = “**angry aardvark**”
  - D1 = “ … **angry** … **aardvark** … “; D2 = “ … **aardvark** … **aardvark** … “
  - ‘**aardvark**’ has higher *idf.* Is D1 more relevant to Q than D2?
- Correction:
  - First occurrence more important than a repeat
  - “squash” the growth of term frequency: $\mathbf{tf}_{\boldsymbol{w,D}} \to \frac{\boldsymbol{tf_{w,D}}}{\boldsymbol{tf_{w,D}+k}}$

Frequency normalization

- Observation 6:
  - Repetitions important in long docs
  - Make it reflect document length: $\mathbf{tf}_{\boldsymbol{w},\boldsymbol{D}} \rightarrow \dfrac{\boldsymbol{tf}_{\boldsymbol{w},\boldsymbol{D}}}{\boldsymbol{tf}_{\boldsymbol{w},\boldsymbol{D}}+\boldsymbol{k}\frac{|\boldsymbol{D}|}{\boldsymbol{avg}(|\boldsymbol{D}|)}}$

tf.idf weighted sum as score of relevance

If word is repeated in the query, it's probably important

Repetitions of query words in the doc

$$\boldsymbol{S(Q,D) = \sum_{w} tf_{w,Q} \frac{tf_{w,D}}{tf_{w,D} + k\frac{|D|}{avg(|D|)}} \log\frac{|C|}{df_w}}$$

The more query words doc match, the better

Repetitions of same word less important than different words. Except in very long documents

Rare words more important

tf.idf weighted sum as score of relevance

$$S(Q,D) = \boxed{\sum_{w} tf_{w,Q}} \frac{tf_{w,D}}{tf_{w,D} + k\frac{|D|}{avg(|D|)}} \log\frac{|C|}{df_{w}}$$

Note that we only care about words that appear in query.

Efficient!

tf.idf weighted sum as score of relevance

- **Rank** documents in order of decreasing s(Q,D)
- State-of-the-art formula for **short queries (very few words in query, simply using $tf_{w,Q}$)**
- Variations actively used by many search engines

$$S(Q,D) = \sum_{w} tf_{w,Q} \frac{tf_{w,D}}{tf_{w,D} + k\frac{|D|}{avg(|D|)}} \log\frac{|C|}{df_w}$$

Application: Matching two documents

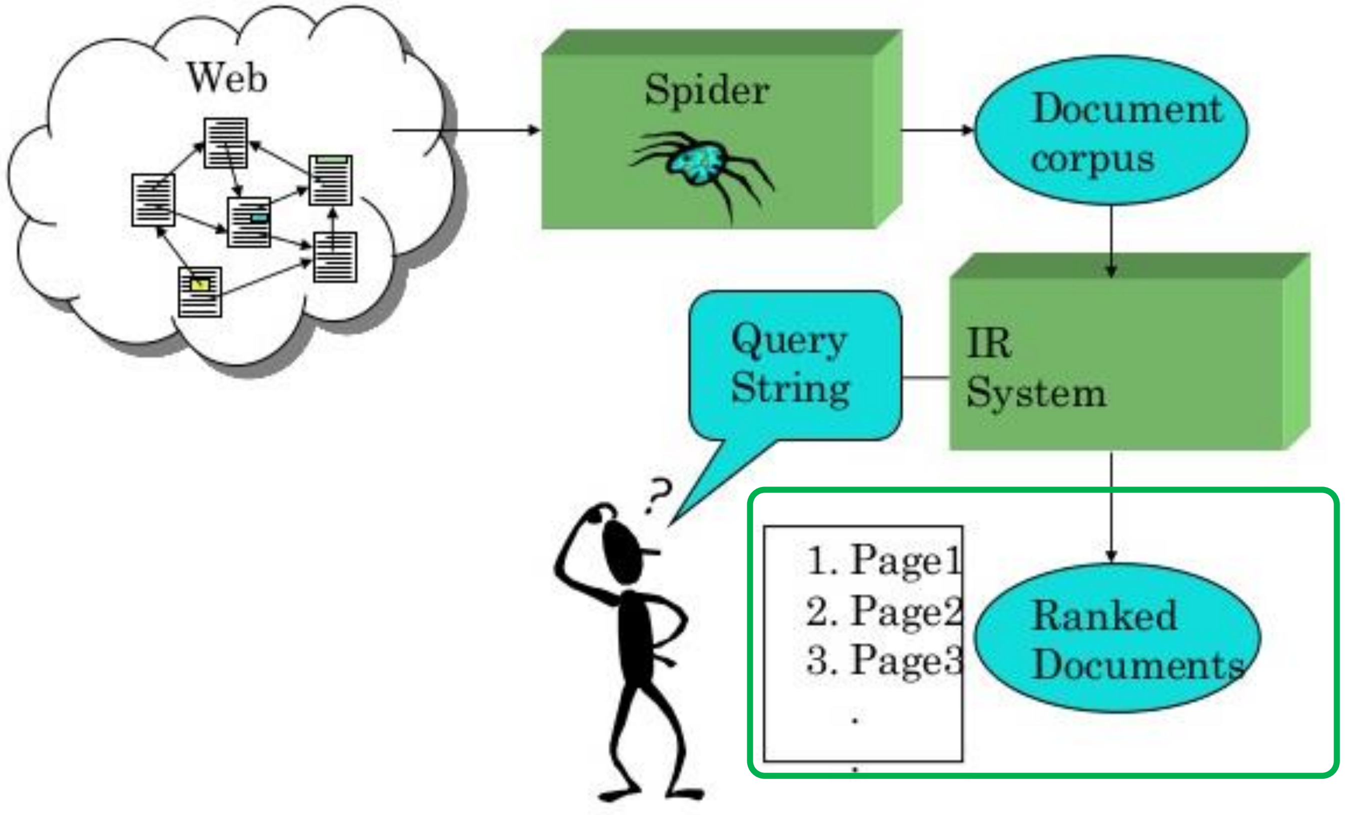
- $S(D_1, D_2)$: matching two documents
  - Patent search
  - Plagiarism check
  - Document recommendation
- $S(D_1, D_2) = cos(D_1, D_2)$
  - Where
  - $D_{1,w} = \frac{tf_{w,D1}}{tf_{w,D1}+k\frac{|D1|}{avg(|D1|)}} \log\frac{|C|}{df_w}$
  - $D_{2,w} = \frac{tf_{w,D2}}{tf_{w,D2}+k\frac{|D2|}{avg(|D2|)}} \log\frac{|C|}{df_w}$

Exercise: Why we use cosine similarity here instead of simply multiplying tf.idf vectors?

Application: Machine learning design interview

- Recommend new friends for a SNS (e.g. facebook, twitter) user
  - Model every user as a “document”
    - with a distribution of keywords and tags (many approaches in practice)
  - From O(N^2) friends of O(N) friends, search and then rank a few “document” that best match current user.
    - A lot of feature engineering in practice

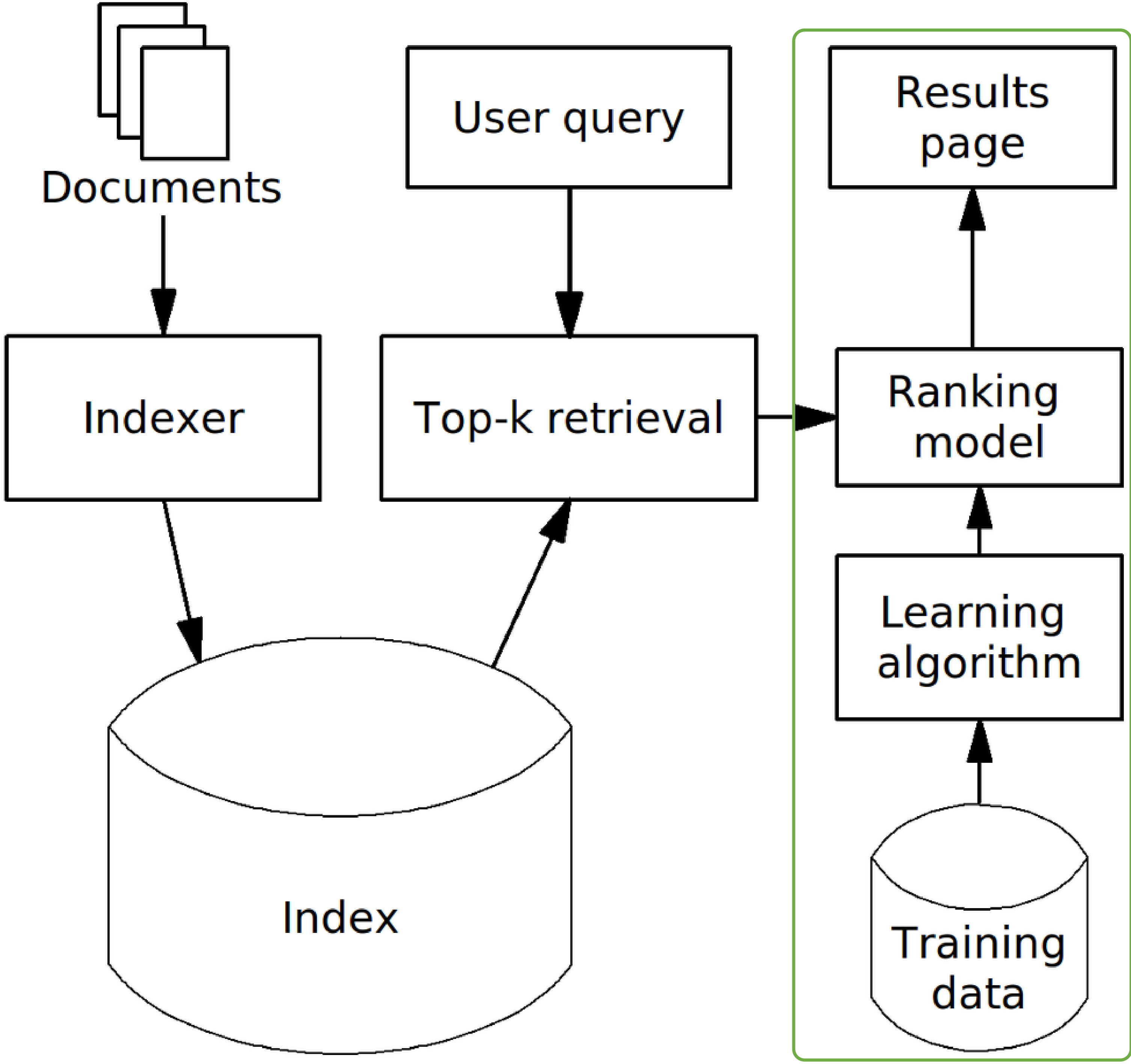
Quick summary



- word based formulation of relevance score
- **Models we used**
  - Bag of words (BOW)
  - Ranking
  - Feature engineering
- Tf.idf: famous example of manually designed features

Learning to Rank

1. **Probabilistic IR with BIM**
2. **Probabilistic IR with BM**
3. **Machine learning ranking**



Ranking method is core of modern IR system

- In what order do we present documents to the user?
- We want the "best" document to be first, second best second, etc…
- Idea: rank by probability of relevance of the document w.r.t. information need
  - P(R=1|document, query)

Probabilistic ranking

- How do we compute all probabilities $P(R = 1|D_i, Q)$?
  - Binary independent model (BIM), under strong irrelevance assumption
  - Incorporating extra information like term frequency, document length: BM25

Binary independence model (BIM)

- Assumptions
  - **Binary:**
    - **Documents** are represented as **binary** incidence vectors of terms (like in Boolean retrieval)
    - **Queries** are represented as **binary** incidence vectors of terms
  - **Independence**: terms occur in documents independently
- Task:
  - Given query Q, for each document D, compute $p(R=1|Q,D)$

Probability of being relevance

- For a specific query Q, we want to find the documents of highest scores S$(R|Q,D)$

$$S(R|Q,D) = \frac{P(R=1|Q,D)}{P(R=0|Q,D)}$$

- Binary Independence model (BIM)

- *Tip: in logistic regression*
- *Linear regression score (before feeding into sigmoid activation) can be interpreted as ratio of two conditional probabilities:*

$$w \cdot x = \frac{P(y=1|w,x)}{P(y=-1|w,x)}$$

A DELIBERATE PRACTICE

- For a specific query Q, we want to find the documents of highest scores $\mathrm{S}(R|Q,D)$

$$S(R|Q,D) = \frac{P(R=1|Q,D)}{P(R=0|Q,D)}$$

- Binary Independence model (BIM)

- **We are going to prove that:**
  - Starting with a simple formulation $S(R|Q,D) = \frac{P(R=1|Q,D)}{P(R=0|Q,D)}$
  - By using Bayesian rule, and using some assumptions
  - We can derive **tf.idf** weights

BIM 1: Bayesian rule

- To calculate $S(R|Q,D)$ for a document D for raking

$$S(R|Q,D) = \frac{P(R=1|Q,D)}{P(R=0|Q,D)} = \frac{P(R=1|Q)P(D|Q,R=1)}{P(R=0|Q)P(D|Q,R=0)}$$

Discriminative prob.

Generative prob.

- We only need to calculate $\frac{P(D|Q,R=1)}{P(D|Q,R=0)}$, since $\frac{P(R=1|Q)}{P(R=0|Q)}$ is independent of document, and does not affect ranking of documents.

BIM 2: Independence assumption I

- Like in naïve Bayesian model

$$S(R|Q,D) = \frac{P(D|Q,R=1)}{P(D|Q,R=0)} = \prod_{i=1}^{V} \frac{P(x_i|Q,R=1)}{P(x_i|Q,R=0)}$$

- $x_i \in \{0,1\}$, i.e. whether word-i exists in document D,
- Define: $p_i = p(x_i = 1\,|R=1,Q), r_i = p(x_i = 1|R=0,Q)$ Generative prob.

$$S(R|Q,D) = \prod_{x_i=1} \frac{p_i}{r_i} \prod_{x_i=0} \frac{1-p_i}{1-r_i}$$

BIM 3: Independence assumption II

- **Assume** $p(x_i = 1\,|R = 1, Q) = p(x_i = 1|R = 0, Q)$ **if $\boldsymbol{x_i}$ is not in Q.**
  - **sometimes it makes sense,** e.g.
    - Q = {**Caesar**, **Brutus**}, D = document “**Hamlet**”
    - P(‘**Hamlet**’ = 1| R=1, {**Caesar**, **Brutus**}) = P(‘**Hamlet**’ = 1 | R=0, {**Caesar**, **Brutus**})
    - “Whether the play is relevant to story between Caesar and Brutus or not” has no effect on whether word “**Hamlet**” appear in this play.

$$S(R|Q,D) = \prod_{x_i=1,q_i=1} \frac{p_i}{r_i} \prod_{x_i=0,q_i=1} \frac{1-p_i}{1-r_i}$$

As a result, we only care about words that appear in query

BIM 3: Independence assumption II

- **Assume** $p(x_i = 1\,|R = 1, Q) = p(x_i = 1|R = 0, Q)$ **if** $\boldsymbol{x_i}$ **is not in Q**.
  - **But not always**, e.g.
    - Q = {**Caesar**, **Brutus**}, D = document "**Antony** and **Cleopatra**"
    - P('**Augustus**' = 1| R=1, {**Caesar**, **Brutus**}) = P('**Augustus**' = 1 | R=0, {**Caesar**, **Brutus**})
    - "Whether the play is relevant to story between Caesar and Brutus or not" is very likely to effect on whether word "**Augustus**" appear in this play.

$$S(R|Q,D) = \prod_{x_i=1,q_i=1} \frac{p_i}{r_i} \prod_{x_i=0,q_i=1} \frac{1-p_i}{1-r_i}$$

As a result, we only care about words that appear in query

BIM 4: Retrieval Status Value

$$S(R|Q,D) = \prod_{x_i=1,q_i=1} \frac{p_i}{r_i} \prod_{x_i=0,q_i=1} \frac{1-p_i}{1-r_i}$$

$$S(R|Q,D) = \prod_{x_i=1,q_i=1} \frac{p_i(1-r_i)}{r_i(1-p_i)} \prod_{q_i=1} \frac{1-p_i}{1-r_i}$$

Exercise: prove it

BIM 4: Retrieval Status Value

- Finally simplified score:

$$S(R|Q,D) = \prod_{x_i=1,q_i=1} \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

- Retrieval Status Value (RSV):

$$RSV = \log(S) = \sum_{x_i=q_i=1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)}$$

- To estimate $S(R|Q,D)$, we only need to **estimate $\boldsymbol{p_i}$, $\boldsymbol{r_i}$** for words in Q

BIM 5: approximate estimation

Documents	Relevance (R=1)	Non-relevance (R=0)	Total
X_i = 1	s	n-s	*n*
X_i = 0	S-s	N-n-S+s	N-n
Total	S	N-S	N

$$p_i = \frac{s}{S}, r_i = \frac{n-s}{N-S}$$

BIM 5: approximate estimation

Documents	Relevance (R=1)	Non-relevance (R=0)	Total
X_i = 1	s	n-s	*n*
X_i = 0	S-s	N-n-S+s	N-n
Total	S	N-S	N

Usually $\boldsymbol{S} \ll \boldsymbol{N}$ (e.g. millions out of trillions of pages on google search)

$$\log \frac{1 - r_i}{r_i} = \log \frac{N - n + S - s}{n - s} \approx \log \frac{N - n}{n} \approx \log \frac{N}{n} = idf$$

BIM 5: approximate estimation

Documents	Relevance (R=1)	Non-relevance (R=0)	Total
X_i = 1	s	n-s	*n*
X_i = 0	S-s	N-n-S+s	N-n
Total	S	N-S	N

$$RSV = \sum_{x_i=q_i=1} \log \frac{p_i(1-r_i)}{(1-p_i)r_i} = \sum_{x_i=q_i=1} \log \frac{p_i}{1-p_i} + idf$$

BIM 5: approximate estimation

Documents	Relevance (R=1)	Non-relevance (R=0)	Total
X_i = 1	s	n-s	*n*
X_i = 0	S-s	N-n-S+s	N-n
Total	S	N-S	N

If assume $p_i = p(x_i = 1 \,|R = 1, Q) = 0.5 \text{ for word in } Q$

$$RSV = \sum_{x_i=q_i=1} 1 \cdot idf$$

We derived simplest tf-idf weighted score using statistics ☺

BIM uses too strong assumptions

1. Term independence
2. Boolean representation of document/queries/relevance

$$S(R|Q,D) = \frac{P(D|Q,R=1)}{P(D|Q,R=0)} = \prod_{i=1}^{V} \frac{P(x_i|Q,R=1)}{P(x_i|Q,R=0)}$$

Where $\boldsymbol{x_i} \in \{\mathbf{0}, \mathbf{1}\}$

BIM uses too strong assumptions

3. Terms not in query don’t affect the outcome

$$p(x_i = 1\,|R = 1, Q) = p(x_i = 1|R = 0, Q)$$

If x_i is not in Q:

4. If x_i is in Q:

$$p_i = p(x_i = 1\,|R = 1, Q) = 0.5$$

BM: relaxing assumptions in BIM

- **BIM**: Boolean representation of document/queries/relevance

$$RSV = \sum_{x_i=q_i=1} \log \frac{p_i(1-r_i)}{(1-p_i)r_i}$$

where $p_i = p(x_i = 1\ |R = 1, Q), r_i = p(x_i = 1|R = 0, Q)$

- **BM**: Word frequency representation of document/queries/relevance

$$RSV = \sum_{x_i=q_i=1} \log \frac{p_i(1-r_i)}{(1-p_i)r_i}$$

where $p_i = p(TF_i = \ \boldsymbol{tf_i}|R = 1, Q), r_i = p(TF_i = \boldsymbol{tf_i}|R = 0, Q)$

BM: relaxing assumptions in BIM

- **BIM**: If x_i is in Q:

$$p_i = p(x_i = 1\,|R = 1, Q) = 0.5$$

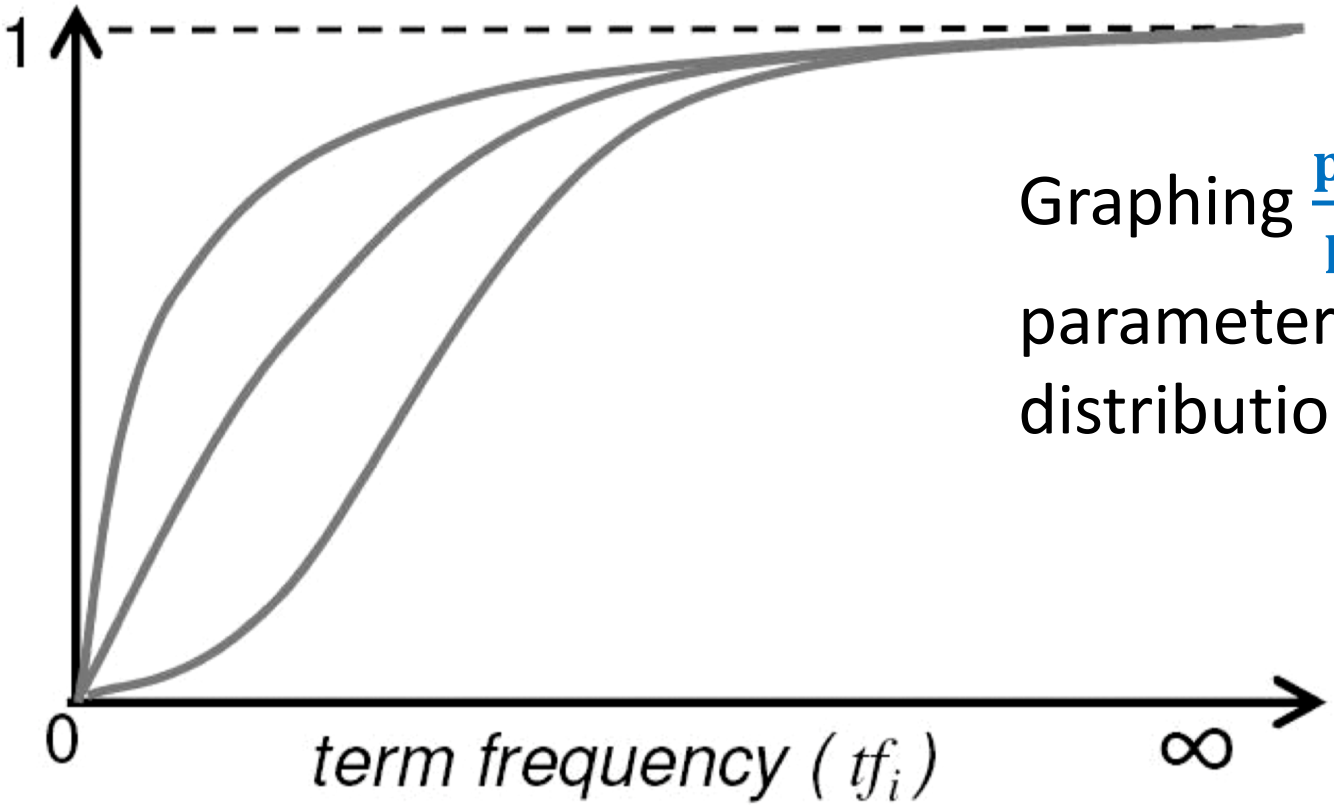
- **BM**: in a relevant document, term frequency of word x_i is modeld as mixture of Poisson distribution:

$$P(TF_i = k|Q, R = 1) = \pi\frac{\lambda^k}{k!}e^{-\lambda} + (1 - \pi)\frac{\mu^k}{k!}e^{-\mu}$$

where π, μ, λ are parameters not known (like we do not know p_i in BIM)

Better p_i estimation

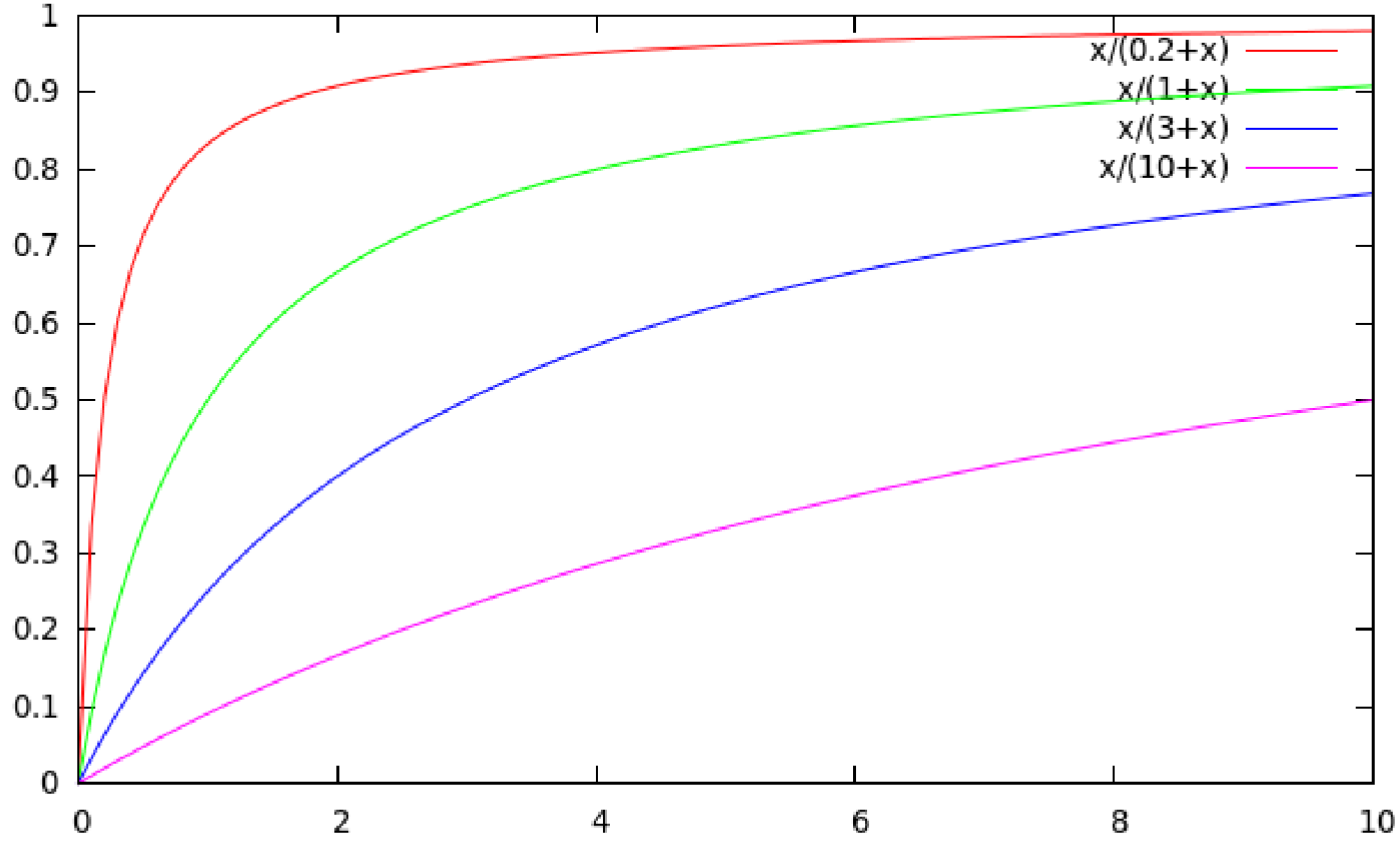
BM: relaxing assumptions in BIM



Graphing $\frac{\mathbf{p}_i(TF=tf)}{\mathbf{p}_\mathrm{i}(TF=0)}$ for different parameter values of the mix-Poisson distribution

Better p_i estimation

BM: relaxing assumptions in BIM



The monotonically saturating pattern can be approximated with

$$\frac{tf}{k + tf}$$

Observation 5.

Better p_i estimation

BM: relaxing assumptions in BIM

- BIM: Assuming $p_i \; = \; 0.5$

$$RSV = \sum_{x_i=q_i=1} 1 \cdot idf$$

- BM: Approximating $\frac{p_i}{1-p_i} = \frac{tf}{k+tf}$

$$RSV = \sum_{x_i=q_i=1} \frac{tf}{k+tf} \cdot idf$$

or

$$RSV = \sum_{x_i=q_i=1} \frac{(k+1)tf}{k+tf} \cdot idf$$

Better p_i estimation

BM: relaxing assumptions in BIM

- Longer documents are likely to have larger tf values
- Normalize *tf* using document length

$$tf' = \frac{tf}{(1-b) + b\frac{|D|}{avg(|D|)}}, 0 \le b \le 1$$

- b=1: full document length normalization
- b=0: no document length normalization

Better p_i estimation

BM25

$$RSV = \sum_{x_i = q_i > 0} \frac{(k+1)tf_i}{k\left((1-b) + b\frac{|D|}{avg(|D|)}\right) + tf_i} idf_i$$

*This weighting formulation is called **BM25***

- *k controls term frequency scaling*
  - *K=0 is binary model BIM; k large is raw term frequency*
- *b controls document length normalization*
  - *b=0 is no length normalization; b= 1 is relative frequency (fully scaled by doc length)*
- *Typically k is set 1.2~2 and b ~0.75*

Better p_i estimation

Quick summary

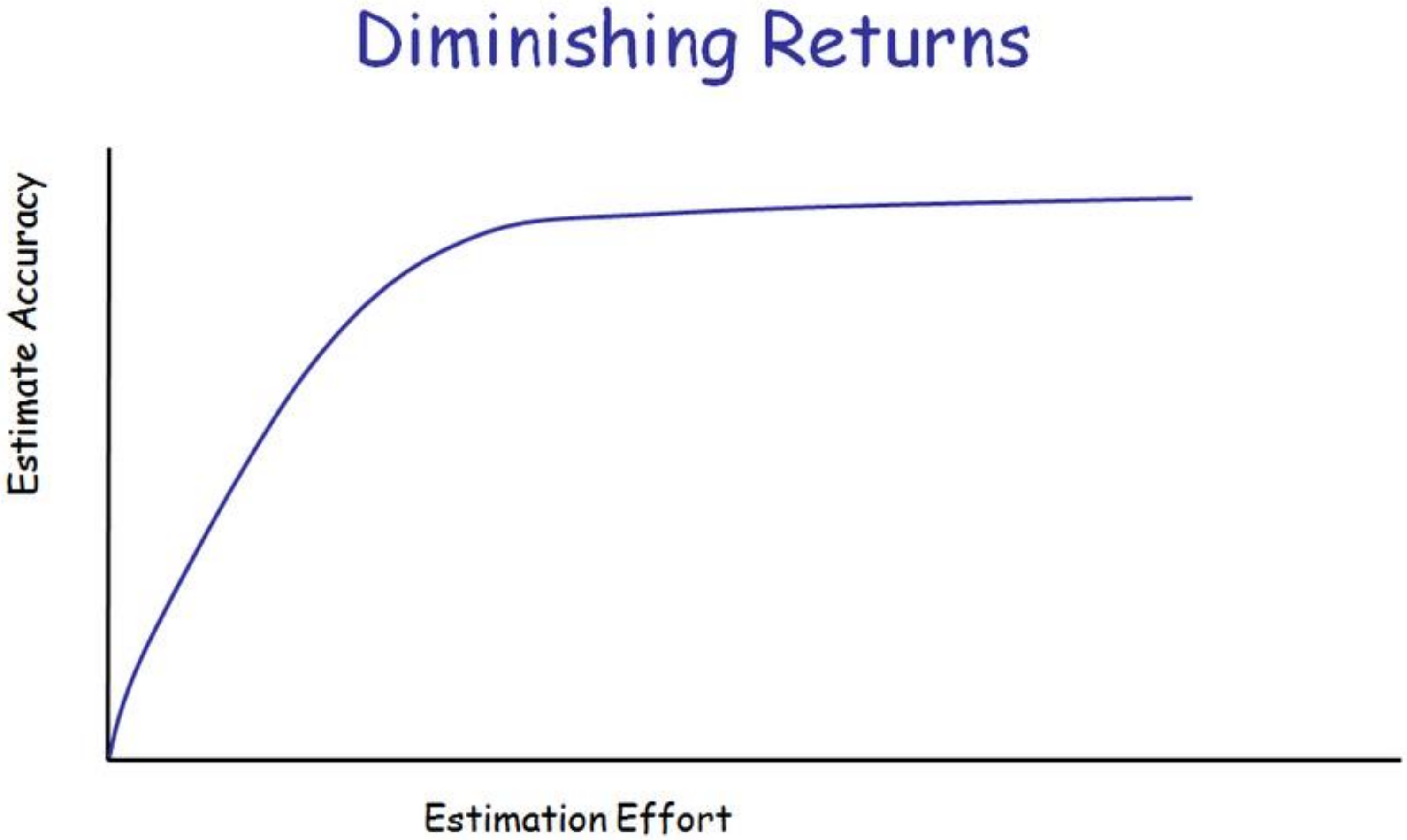
- Derived tf.idf term weighting using mathematical way
- **Models we used**
  - Bayesian rule and Naïve Bayesian
  - Using assumptions and approximation to simplify problems
  - Curve fitting

Machine “Learning to Rank”

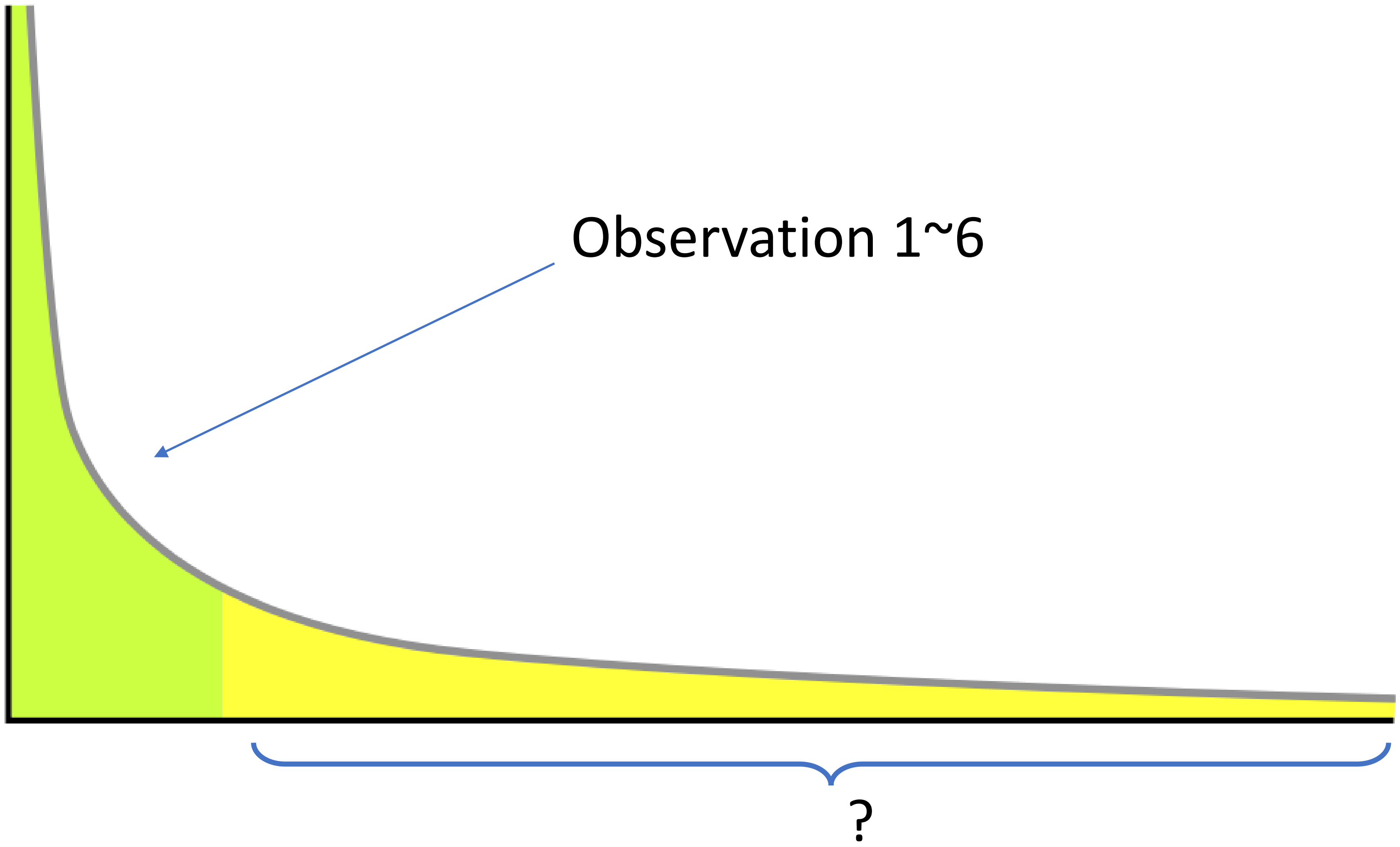
- Tf.idf and BM25
  - Effective and efficient in industrial information retrieval task
- Yet,
  - How to further improve it?
  - Can we relax the term independence in statistical IR model?
- Can we directly estimate P(R=1|Q,D), using machine learning, e.g. logistic regression, directly from data?

Why machine learning?

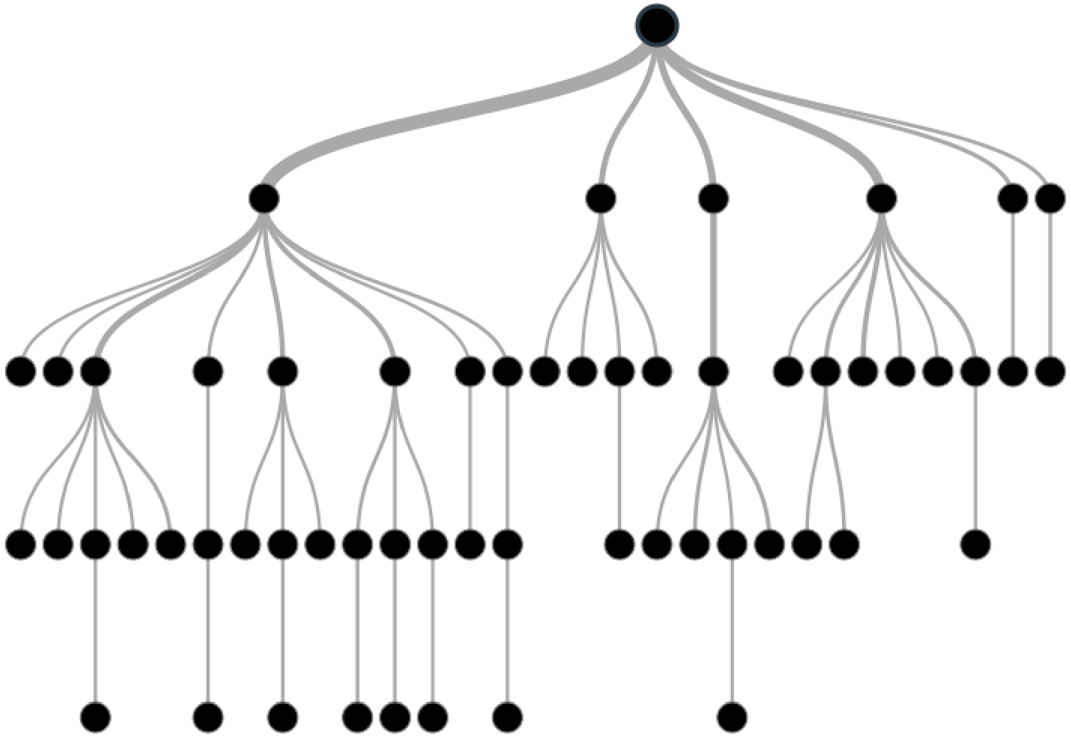
Why machine learning?

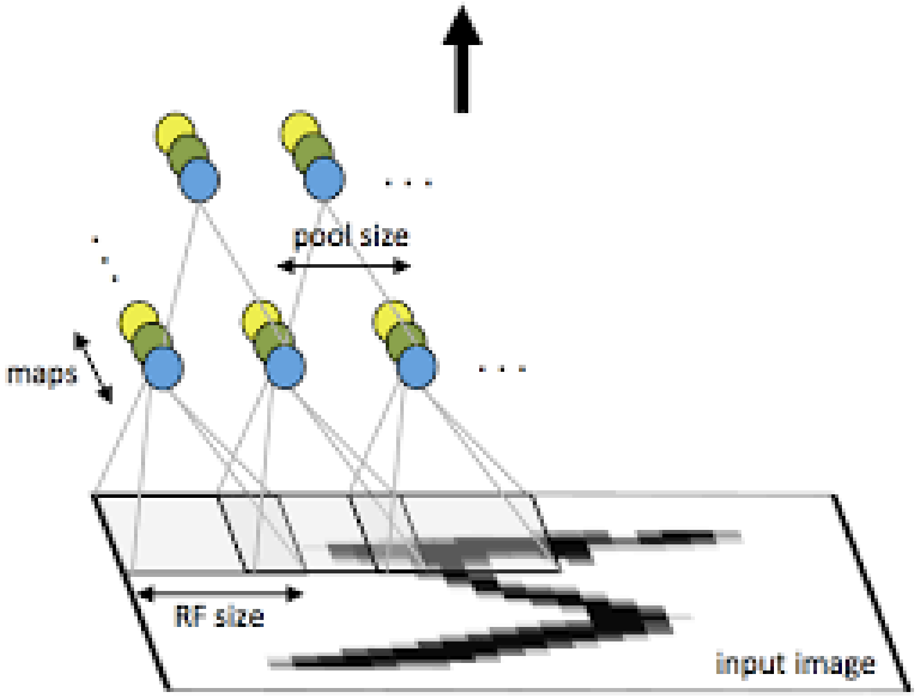


Why machine learning?



Why machine learning?





Machine learning can discover complicated, high order correlation between raw features, which can explain many long tail observations.

Machine “learning to rank”

- **Classification problems**:
    - Map to an unordered set of classes, $P(R = 1|Q, D)$
    - Classification probably isn’t the right way to think about ranking
- **Regression problems**:
    - Map to a real value
    - Absolute score of relevance? We do not have data
- **Ordinal regression problems**:
    - Map to an ordered set of classes, suitable for ranking

Machine “Learning to Rank”

- The ordinal regression problem gives extra power:
  - **Relations** between relevance levels are modeled
  - Documents are good **versus** other documents for query given collection

Pair-wise learning for ordinal regression

- Assume training data is available consisting of **document-query pairs** represented as feature vectors F_i and relevance ranking c_i
- **pair-wise learning:** the input is a pair of results for a query, and the **class label** is the relevance ordering relationship between them
- Aim is to classify instance pairs as **correctly** ranked or **incorrectly** ranked
  - **This turns an ordinal regression problem back to a binary classification problem**
  - We want a ranking function $f(\cdot)$ such that $c_i > c_k \text{ iff } f(X_i) > f(X_k)$

Machine learning in practice

- Modern systems – especially on the Web – produced large amount of query log data, and with a great number of features:
- E.g. manually defined features, e.g.
    - Log frequency of query word in anchor text?
    - Query word in color on page?
    - # of images on page?
    - # of (out) links on page?
    - PageRank of page?
    - URL length?
    - URL contains “~”?
    - Page edit recency?
    - Page loading speed

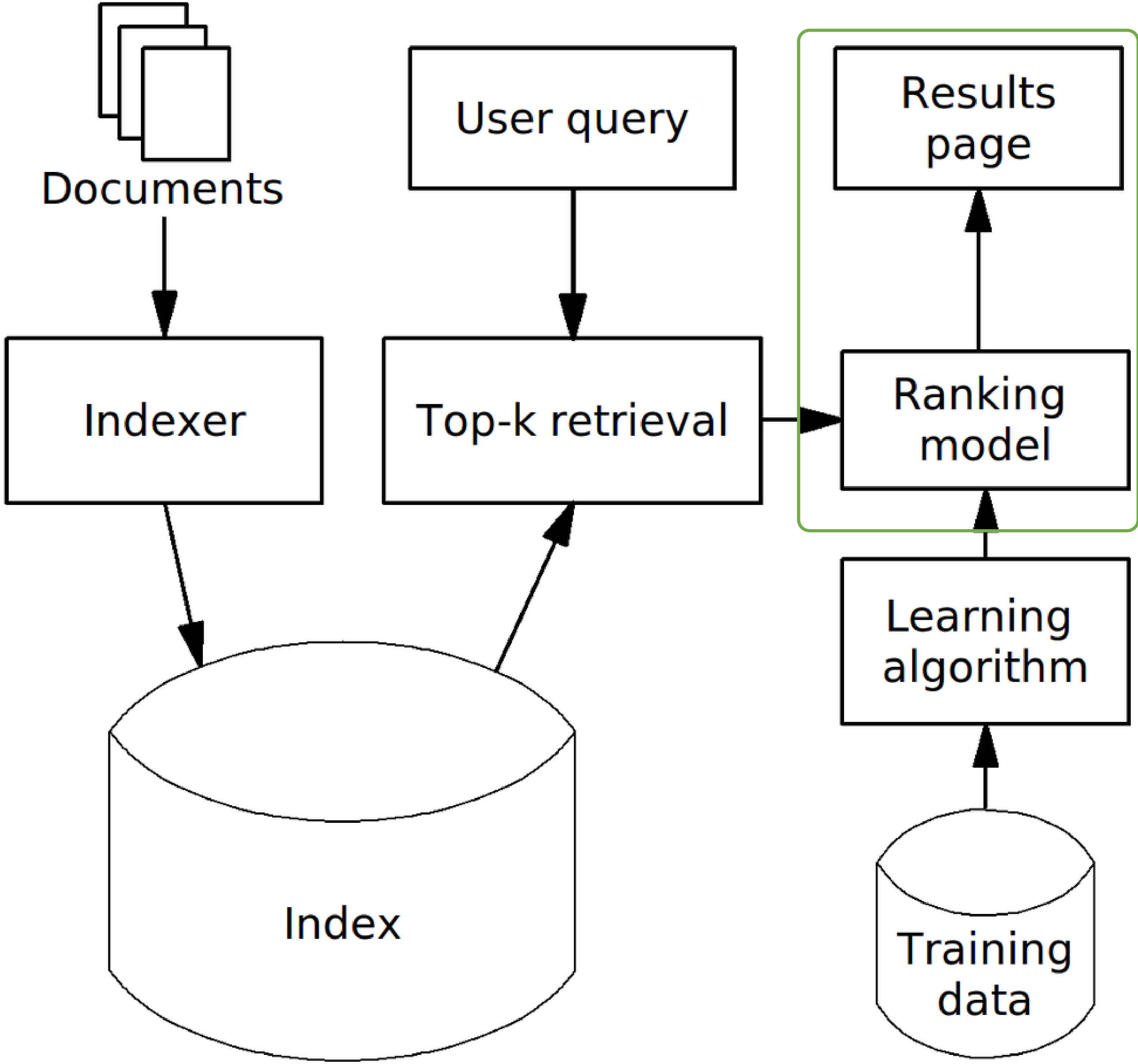
Machine learning in practice

- Modern systems – especially on the Web – produced large amount of query log data, and with a great number of features:
- Many high dimensional features, e.g.
  - Many embedding vectors
  - Kernel based feature expansion
  - Output of ensembles of models (e.g. boosting trees) as features

Quick summary

- Explained motivation and implementation of “learning to rank”
- **Models used:**
    - Regression
    - Ranking loss (choosing reasonable objective functions)

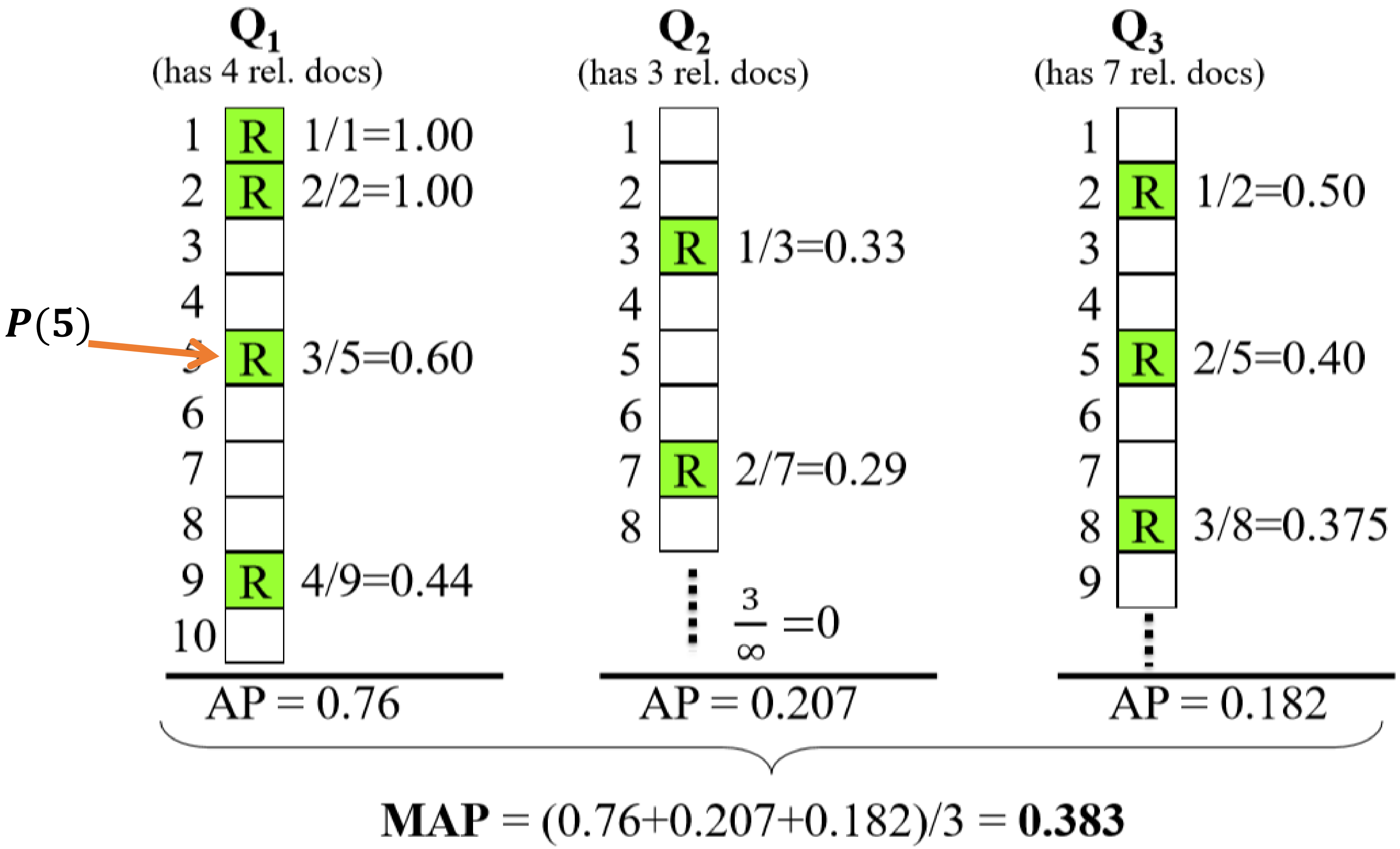
Evaluation


Documents
User query
Results page
Indexer
Top-k retrieval
Ranking model
Learning algorithm
Index
Training data

Imbalanced labels

- Imbalanced labels:
  - Labels are {relevant, irrelevant}
  - Most of documents in corpus are irrelevant to query
- Recall, precision, AUC are widely used evaluation metrics for imbalanced labels
- However, we
  - not only care if document is correctly labeled relevant or irrelevant
  - also care the position (**ranking**) of relevant among all retrieved documents

Mean Average Precision (MAP)



$P(5)$

Q_1 (has 4 rel. docs)

1 R 1/1=1.00
2 R 2/2=1.00
3
4
5 R 3/5=0.60
6
7
8
9 R 4/9=0.44
10

AP = 0.76

Q_2 (has 3 rel. docs)

1
2
3 R 1/3=0.33
4
5
6
7 R 2/7=0.29
8

$\frac{3}{\infty}$ =0

AP = 0.207

Q_3 (has 7 rel. docs)

1
2 R 1/2=0.50
3
4
5 R 2/5=0.40
6
7
8 R 3/8=0.375
9

AP = 0.182

MAP = (0.76+0.207+0.182)/3 = **0.383**

$$AP = \frac{1}{r}\sum_{k=1}^{n} P(k) \times rel(k)$$

- r: number of relevant docs given the query
- n: number of documents retrieved
- $P(k)$: $precision@k$
- $Rel(k)$: 0/1 indicator whether the k-th retrieved doc is relevant or not

Discounted Cumulative Gain (DCG)

- Gain is accumulated starting at the top of ranking and may be reduced (discounted) at lower ranks
- User care more about high-ranked documents, so we discount results by $1/\log_2(rank)$
  - The discount at 4 is 2, at 8 is 3
- DCG_k is the total gain accumulated at a particular rank k (sum of DG up to rank k):

$$DCG_k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{\log_2 i}$$

Where rel_i is grades of position *i*

Normalized DCG (nDCG)

- DCG values are often normalized by comparing the DCG at each rank with the DCG value for the perfect ranking

$$nDCG = \frac{DCG@k}{iDCG@k} \leq 1$$

Demo of nDCG

÷

k	G	DG	DCG@k	iG	iDG	iDCG@k	nDCG@k
1	3	3	3	3	3.00	3	1.00
2	2	2	5	3	3.00	6	0.83
3	3	1.89	6.89	3	1.89	7.89	0.87
4	0	0	6.89	2	1.00	8.89	0.78
5	0	0	6.89	2	0.86	9.75	0.71
6	1	0.39	7.28	2	0.77	10.52	0.69
7	2	0.71	7.99	1	0.36	10.88	0.73
8	2	0.67	8.66	0	0.00	10.88	0.80
9	3	0.95	9.61	0	0.00	10.88	0.88
10	0	0	9.61	0	0.00	10.88	0.88

$$DCG_k = rel_1 + \sum_{i=2}^{k} \frac{rel_i}{\log_2 i}$$

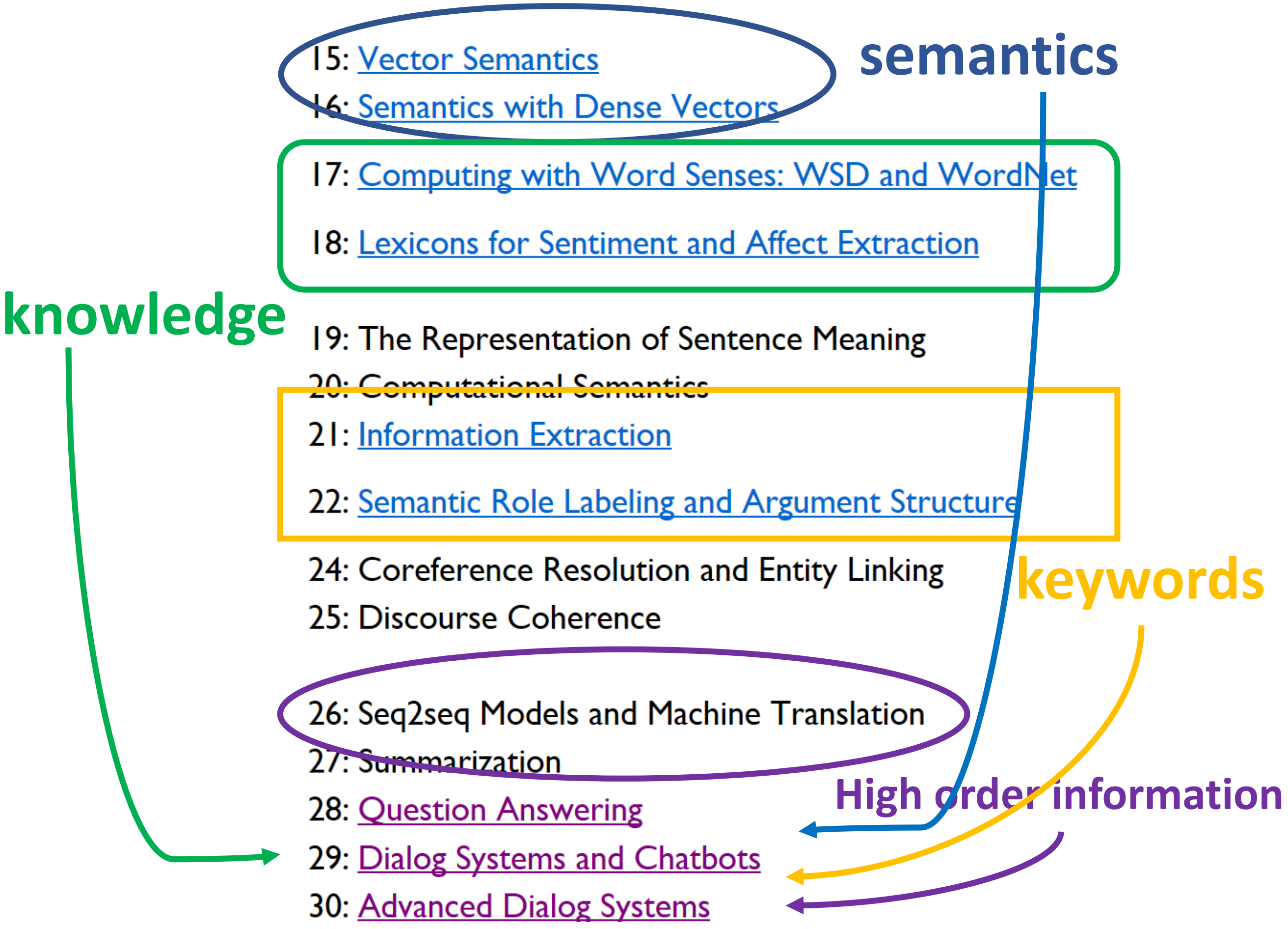
$$nDCG = \frac{DCG@k}{iDCG@k}$$

Question Answering

IR application

NLU & NLG

A road map toward stronger (still weak) AI



15: Vector Semantics

16: Semantics with Dense Vectors

17: Computing with Word Senses: WSD and WordNet

18: Lexicons for Sentiment and Affect Extraction

19: The Representation of Sentence Meaning

20: Computational Semantics

21: Information Extraction

22: Semantic Role Labeling and Argument Structure

24: Coreference Resolution and Entity Linking

25: Discourse Coherence

26: Seq2seq Models and Machine Translation

27: Summarization

28: Question Answering

29: Dialog Systems and Chatbots

30: Advanced Dialog Systems

Speech processing and NLP (3rd edition)

Will it rain in Cupertino?
Is the weather going to get worse today?
What is the weather like today?
What's the upcoming forecast ?
Do I need an umbrella today?

Figure source

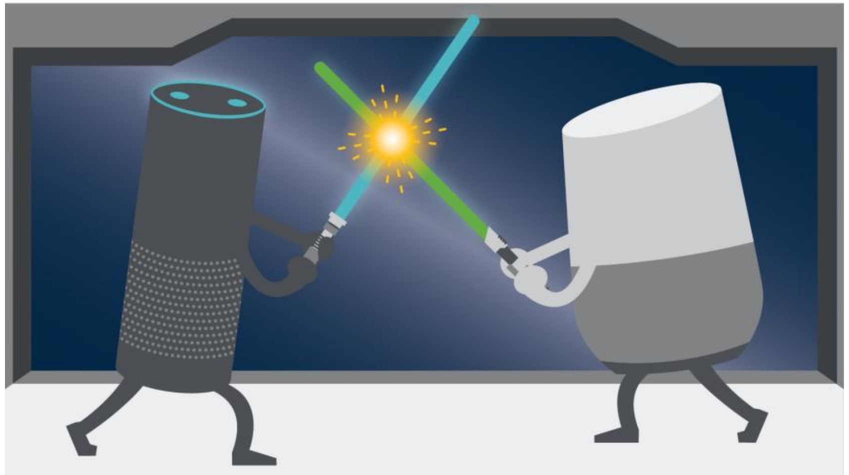


Figure source

Objective: factoid QA

Question	Answers
Where is the Louvre Museum located?	Paris, France
What’s the abbreviation for limited partnership?	L.P.
What currency is used in China?	The yuan
What kind of nuts are used in marzipan?	almonds
What instrument does Max Roach play?	drums

Most current QA systems focus on **factoid questions**. Factoid questions are questions that can be answered with simple facts expressed in short text answers.

In short, a factoid QA is about providing concise facts.

What functionalities are needed?

1. QA system needs to understand the **question**
2. QA system needs to **find** sources containing the answer
3. QA system needs to **locate** the concise fact precisely

IR based QA flowchart

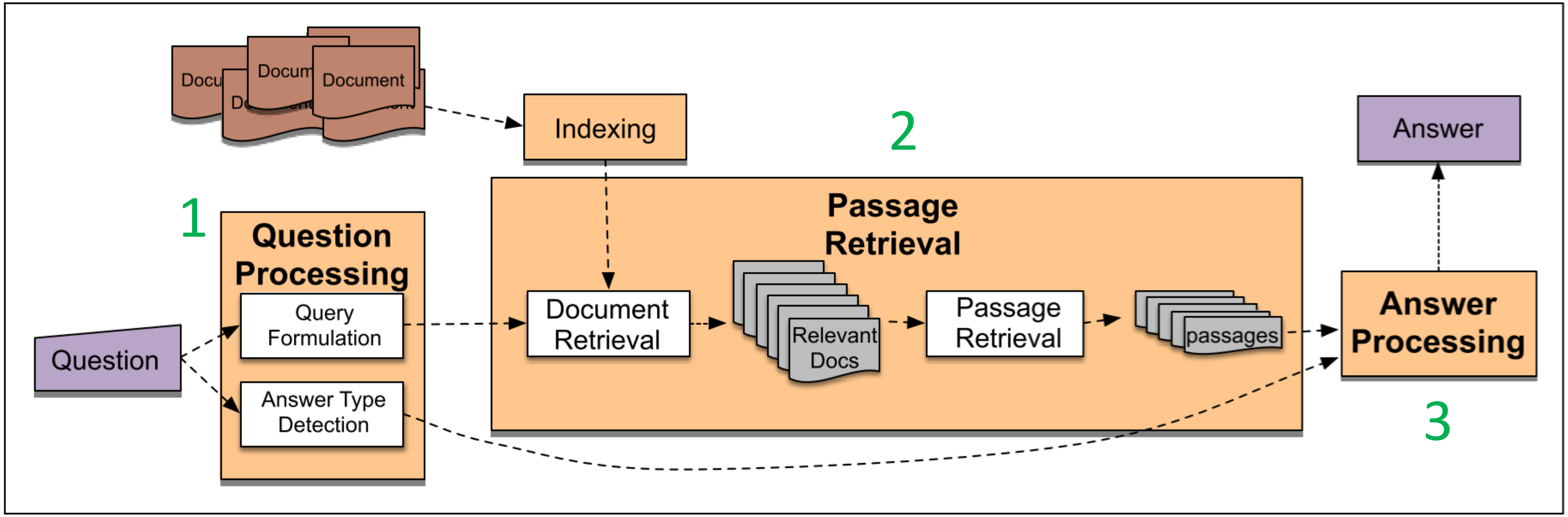


Figure 28.2 IR-based factoid question answering has three stages: question processing, passage retrieval, and answer processing.

Figure from chapter 28 of Speech processing and NLP (3rd edition)

Phase 1. NLU

In order to respond correctly to a free form factual question given a large collection of texts, one needs to understand the question to a level that allows determining some of the constraints the question imposes on a possible answer.

Li & Roth 2002

Parsing question via keywords

- From the question sentence, extract a number of **pieces** of information.
  - The **answer type** specifies the kind of entity the answer consists of (person, location, time, etc.).
  - The **query** specifies the keywords that should be used for the **IR system** to use in **searching** for documents.
  - The **focus**, which is the string of words in the question that are likely to be replaced by the answer in any answer string found
- *Which US state capital has the largest population?*
  - **Answer Type**: city
  - **Query**: US state capital, largest, population
  - **Focus**: state capital

Then we know what to look for

- If we know the **answer type** for a question, we can avoid looking at every sentence or noun phrase in the entire suite of documents for the answer, instead focusing on, for example, just people or cities.
  - **E.g.** Which US **state capital (city)** has the largest population?
  - From documents relevant to “population”, “us state capital” and “largest”, we can focus on the sentences containing NER of city name type.

Widely used in realistic NLU modules

- Siri, what is **weather** of Los Angeles tomorrow?
  - Sunny.
- Messenger bot, book a **flight** to JFK next Wednesday.
  - Sure, what is the departure airport?
- OK google, **call** Alexa.
  - Dialing …

- Intents discovery and dialogue state tracker (DST)
- Call modules corresponding to these intents

Models

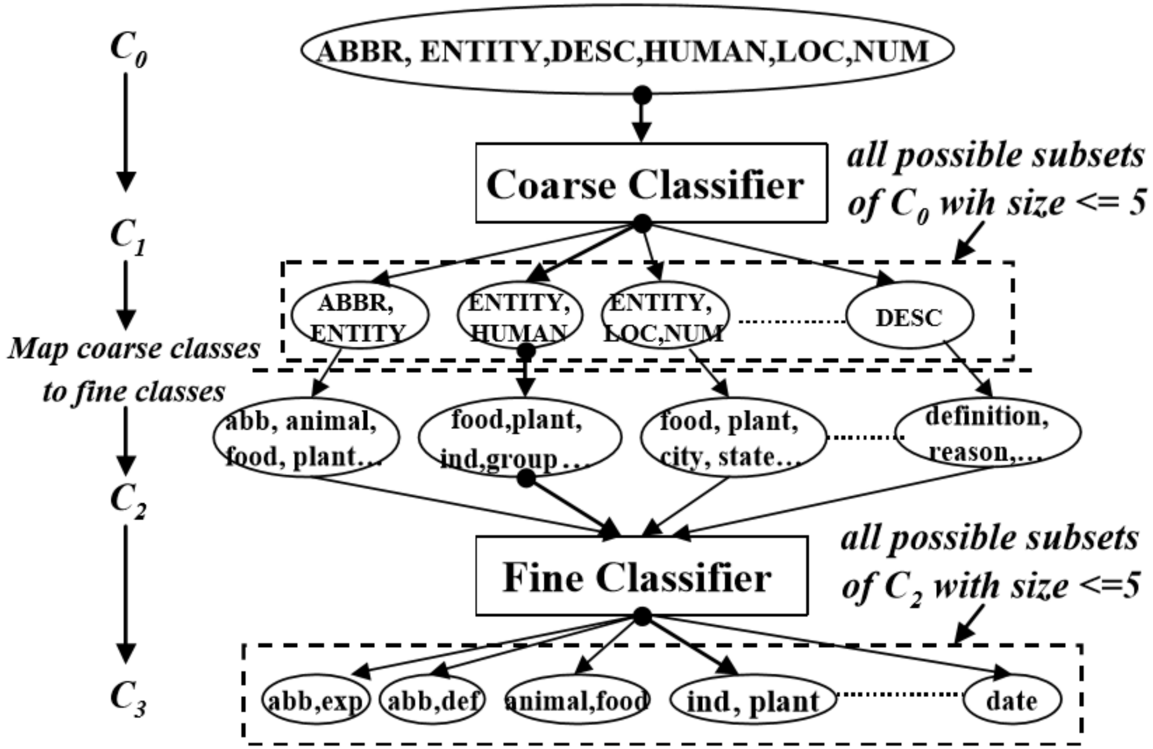
- Rules:
  - If a query starts with ***Who*** or ***Whom***: type **Person**.
  - If a query starts with ***Where***: type **Location**
  - If a query contains ***Which*** or ***What***, the head noun phrase determines the class.
- Information extraction: NER
  - Which US **state** **capital (city)** has the largest population?
  - … B-location I-location …
- machine learning

Formulating a machine learning problem

Features:

1. ***words***
2. Syntactic features:
    1. ***pos tags***
    2. ***Chunks*** (nonoverlapping phrases)
    3. ***head chunks*** (the first noun chunk in a sentence)
3. Semantic features:
    1. ***named entities***
    2. ***semantically related words*** (words that often occur with a specific question class)

Classifier architecture:



C_0
ABBR, ENTITY,DESC,HUMAN,LOC,NUM
Coarse Classifier
all possible subsets of C_0 wih size <= 5
C_1
ABBR, ENTITY
ENTITY, HUMAN
ENTITY, LOC,NUM
DESC
Map coarse classes to fine classes
abb, animal, food, plant…
food,plant, ind,group…
food, plant, city, state…
definition, reason,…
C_2
Fine Classifier
all possible subsets of C_2 with size <=5
C_3
abb,exp
abb,def
animal,food
ind, plant
date

Formulating a machine learning problem

Meaningful labels:

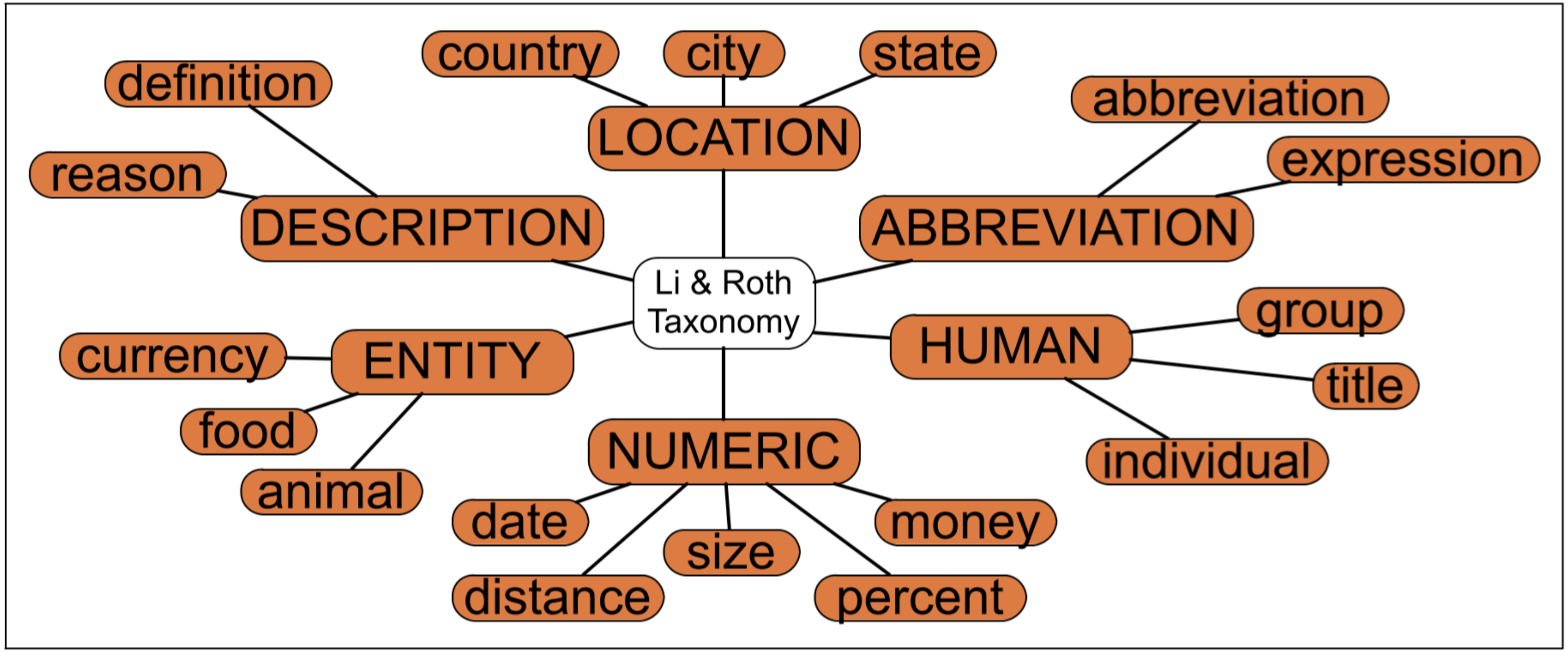
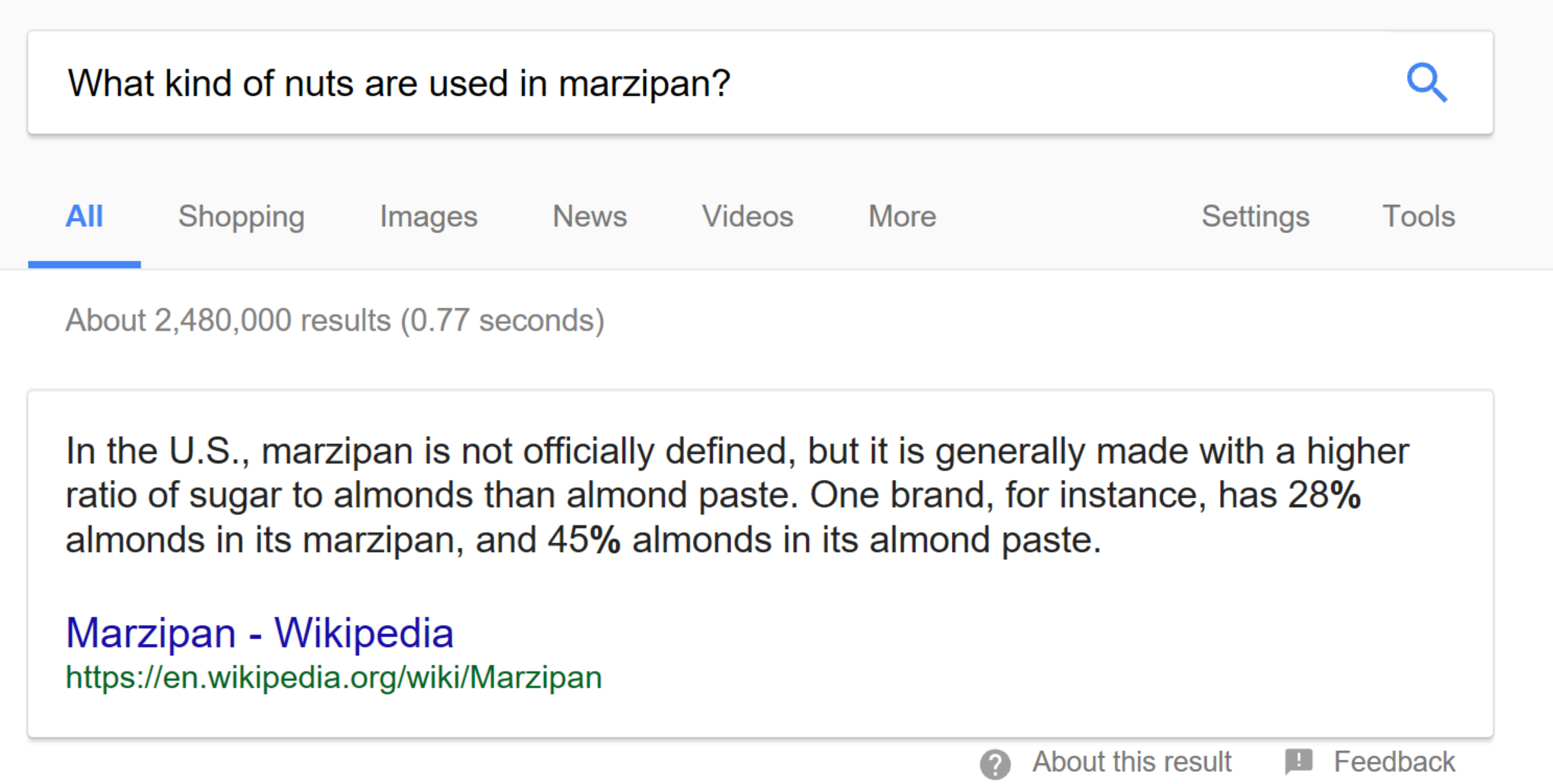


Figure 28.3 A subset of the Li and Roth (2005) answer types.

Phase 2. Information and Passage retrieval



What kind of nuts are used in marzipan?

About 2,480,000 results (0.77 seconds)

In the U.S., marzipan is not officially defined, but it is generally made with a higher ratio of sugar to almonds than almond paste. One brand, for instance, has 28**%** almonds in its marzipan, and 45**%** almonds in its almond paste.

Marzipan - Wikipedia
https://en.wikipedia.org/wiki/Marzipan

Find relevant documents and then filter out passages in the returned documents that don’t contain potential answers.

Phase 3. Answer processing

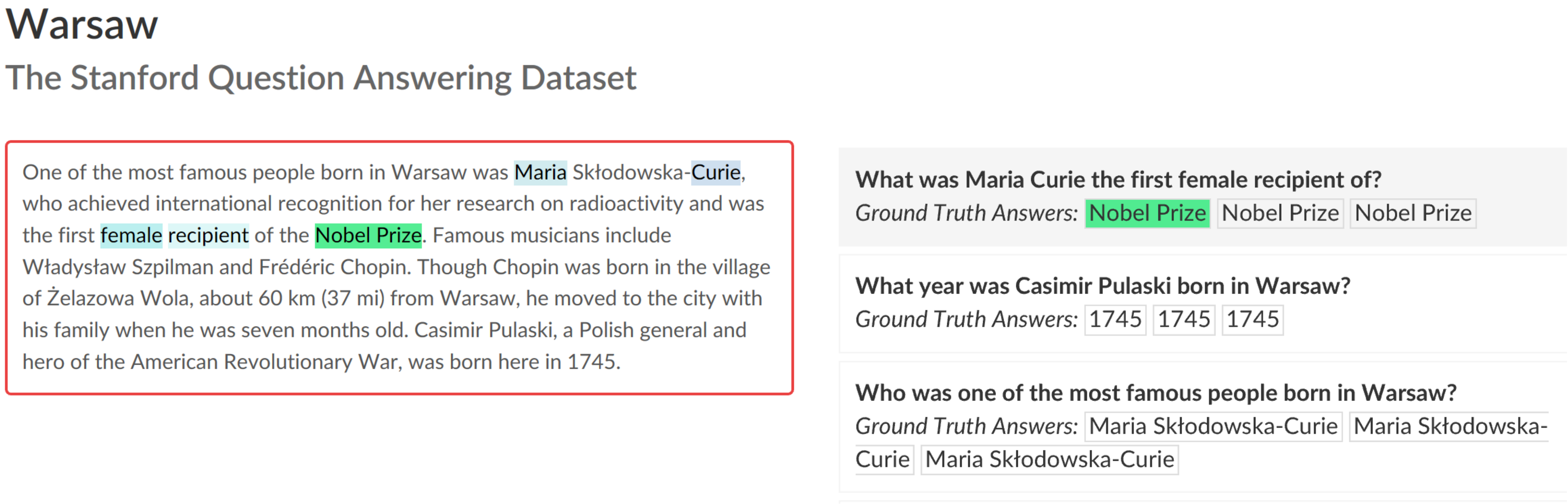
- Use answer type to locate corresponding NER in retrieved sentence, e.g.
  - "How **tall** is Mt. Everest?"
  - The official height of Mount Everest is **29029 feet**.
- Pattern match

Pattern	Question	Answer
<AP> such as <QP>	What is autism?	", developmental disorders such as autism"
<QP>, a <AP>	What is a caldera?	"the Long Valley caldera, a volcanic crater 19 miles long"

More QA examples

- Query from database, e.g.
  - **Input:** Hello, xxx. I'd like to book a table for six people in an expensive range. Moreover, I prefer Italian food, at downtown LA.
  - **Parsing:** {Italian; Los Angeles; Six; Expensive}
  - **API_Call:** My_SQL query "select * where food = Italian and Location = Los Angeles and price = Expensive and People = Six"

More QA examples



Warsaw

The Stanford Question Answering Dataset

One of the most famous people born in Warsaw was Maria Skłodowska-Curie, who achieved international recognition for her research on radioactivity and was the first female recipient of the Nobel Prize. Famous musicians include Władysław Szpilman and Frédéric Chopin. Though Chopin was born in the village of Żelazowa Wola, about 60 km (37 mi) from Warsaw, he moved to the city with his family when he was seven months old. Casimir Pulaski, a Polish general and hero of the American Revolutionary War, was born here in 1745.

What was Maria Curie the first female recipient of?
Ground Truth Answers: Nobel Prize | Nobel Prize | Nobel Prize

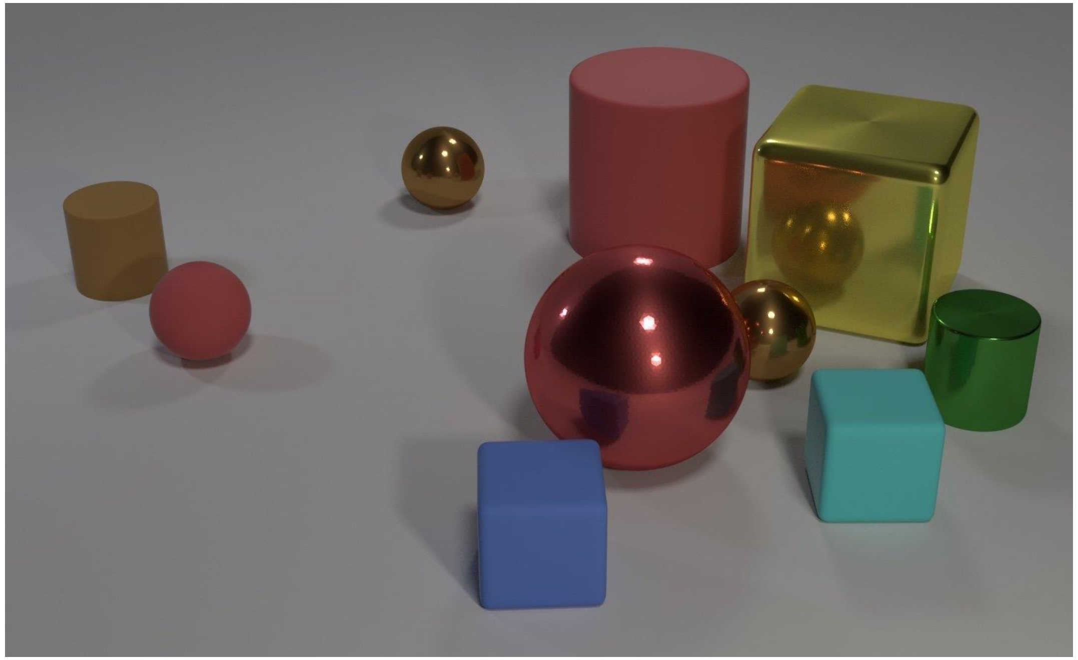
What year was Casimir Pulaski born in Warsaw?
Ground Truth Answers: 1745 | 1745 | 1745

Who was one of the most famous people born in Warsaw?
Ground Truth Answers: Maria Skłodowska-Curie | Maria Skłodowska-Curie | Maria Skłodowska-Curie

More QA examples

- I: Jane went to the hallway.
- I: Mary walked to the bathroom.
- I: Sandra went to the garden.
- I: Daniel went back to the garden.
- I: Sandra took the milk there.
- **Q:** Where is the milk?
- **A:** garden

More QA Examples

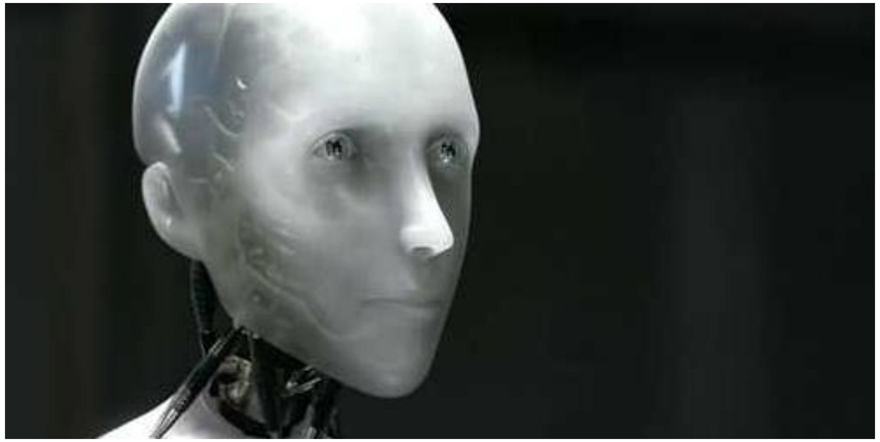


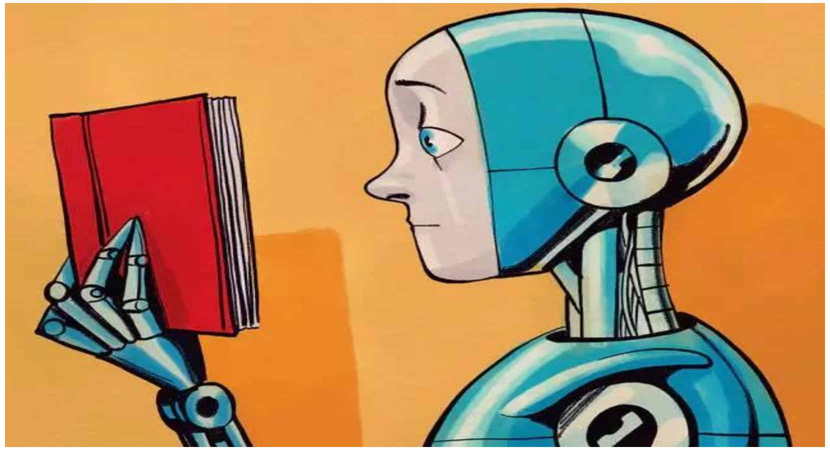
What size is the cylinder that is left of the brown metal thing that is left of the big sphere?

Quick summary

- Brief introduction of locating answers from retrieved documents
- Models used:
  - Machine learning classification models
  - Cascading of multiple subproblems
  - NER, POS tagging, chunking, etc
  - Ensemble method

The End





Still limited to “searching”, quite far away from real **NLU**