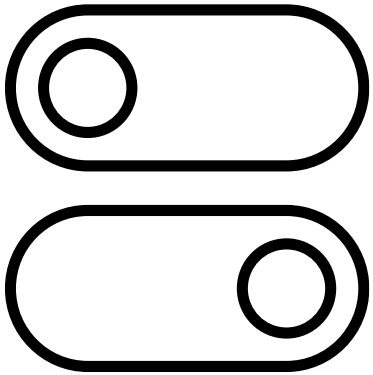
Cluster of Excellence
The Politics of Inequality

Steinbeis Transfer Center
Linguistic Data Analysis

UNIVERSITÄT PASSAU



Information Retrieval & Natural Language Processing

Week 2: IR intro, indexing, Boolean IR

Annette Hautli-Janisz, Prof. Dr.

24 October 2024

Information Retrieval (IR): Today

(IIR): Manning, Raghavan and Schütze, Introduction to IR, 2008

Chapter 1: Boolean retrieval

Chapter 2: The term vocabulary and postings lists

Information Retrieval (IR): Definition

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).

Manning, Raghavan and Schütze, Introduction to IR, 2008

Information Retrieval (IR): Definition

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).*

Manning, Raghavan and Schütze, Introduction to IR, 2008

A few important concepts here…

Unstructured data:

- not easily searchible
- takes no particular structure regarding shape or content
- formats can include text, audio, video, social media postings…
- for text:
  - any character encoding
  - any document length
  - anything between a meaningless string of characters and the complete works of Shakespeare

A few important concepts here…

Structured data:

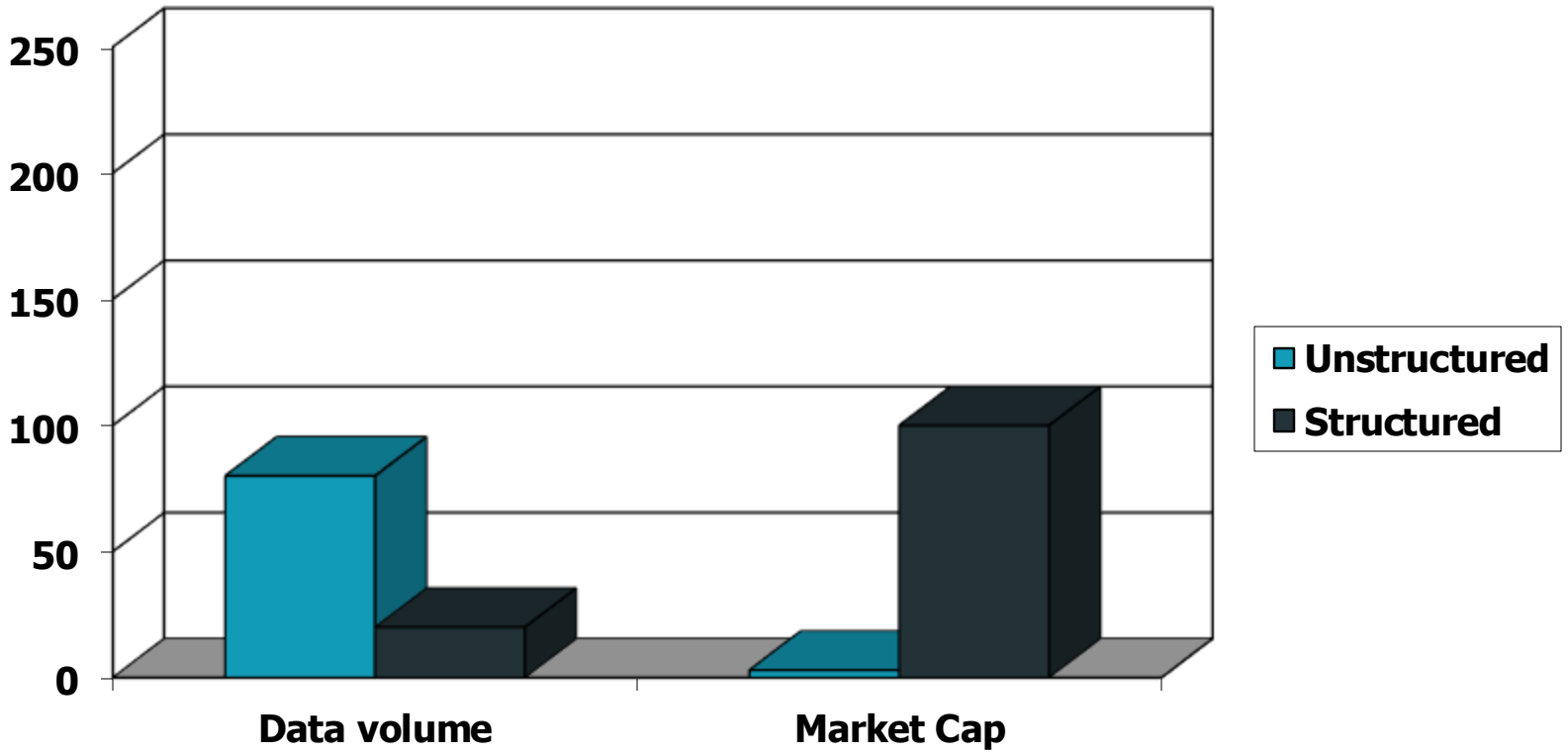
- data conforms to a particular interface or schema
- relational databases (SQL databases)
- easily searchable with queries
- allows for assertions and predictions of the output type and shape

A few important concepts here…

Semi-structured data:

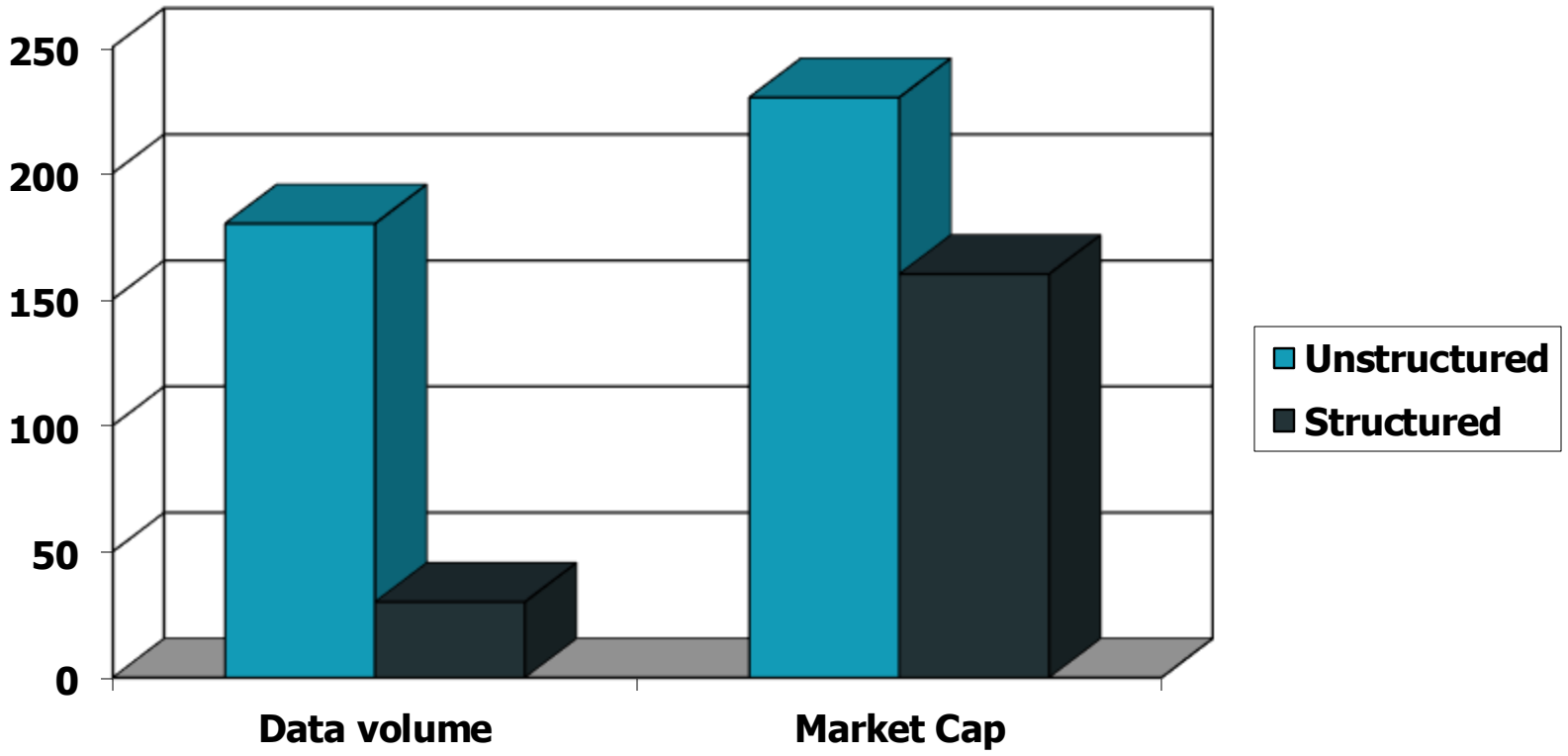
- “everything else”
- object notation like json and xml
- use tools to parse the structure
- a lot of flexibility and chance for errors to be introduced to the structure

Unstructured (text) versus structured (database) data: 1990s



https://web.stanford.edu/class/cs276/

Unstructured (text) versus structured (database) data: today



https://web.stanford.edu/class/cs276/

Information Retrieval (IR): Definition

*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).*

Manning, Raghavan and Schütze, Introduction to IR, 2008

A few important concepts here…

Information need

- An individual or group's desire to locate and obtain information to satisfy a conscious or unconscious need.
- Information needs are expressed as queries.
- Main input parameter and main evaluation criteria for an IR system.

→ An accurate information need assessment is crucial for the success of an IR system.

Information Retrieval (IR): Definition

*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).*

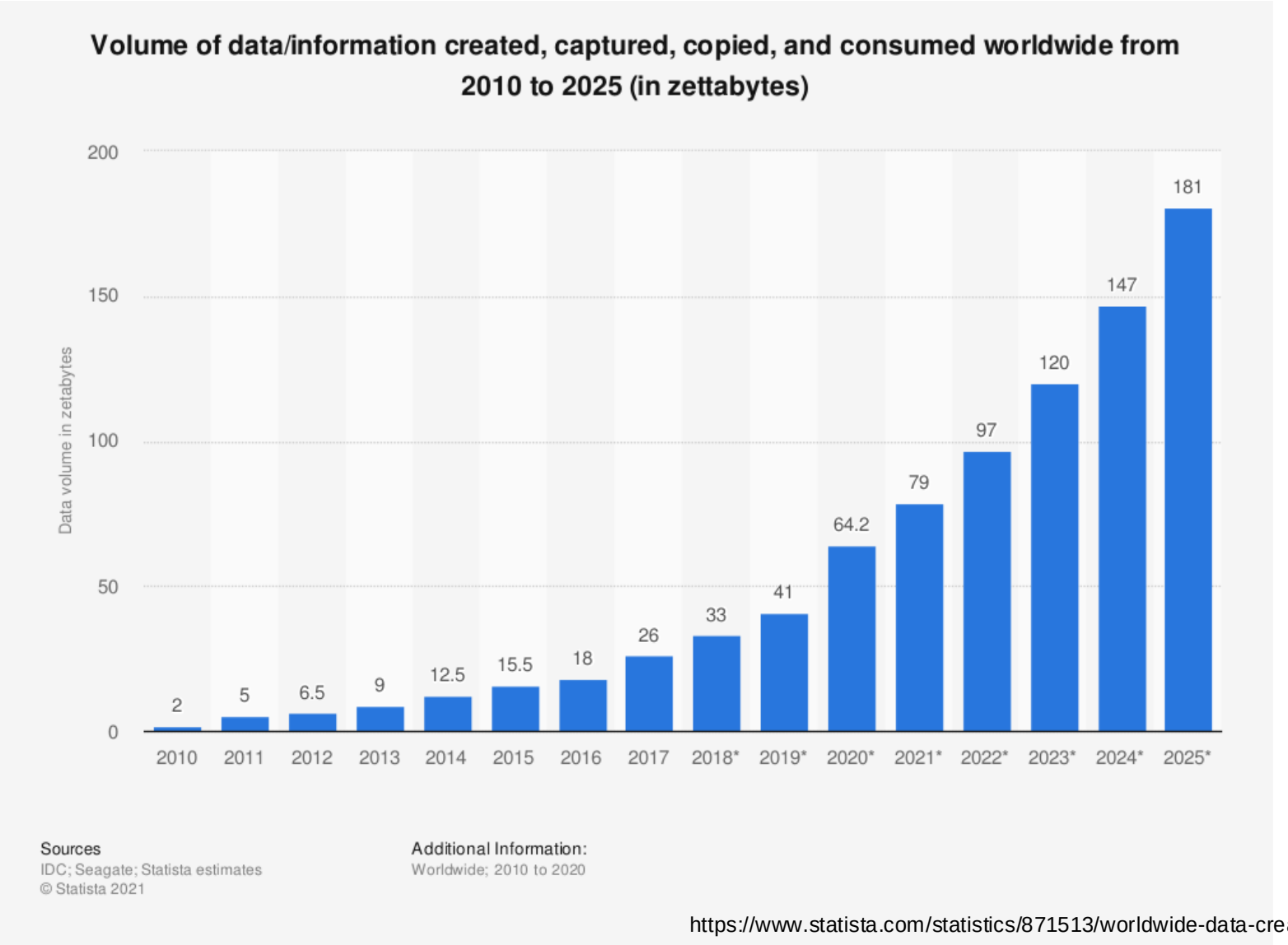
Manning, Raghavan and Schütze, Introduction to IR, 2008

A few important concepts here…

Large collections

In the case of web search: mind-boggling!

A few important concepts here…



https://www.statista.com/statistics/871513/worldwide-data-created/

A few important concepts here…

The International System of Units

Prefix	Decimal
kilo	1,000 (3 zeros)
mega	1,000,000 (6 zeros)
giga	1,000,000,000 (9 zeros)
tera	1,000,000,000,000 (12 zeros)
peta	1,000,000,000,000,000 (15 zeros)
exa	1,000,000,000,000,000,000 (18 zeros)
zetta	1,000,000,000,000,000,000,000 (21 zeros)
yotta	1,000,000,000,000,000,000,000,000 (24 zeros)

A few important concepts here…

Large collections

https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/

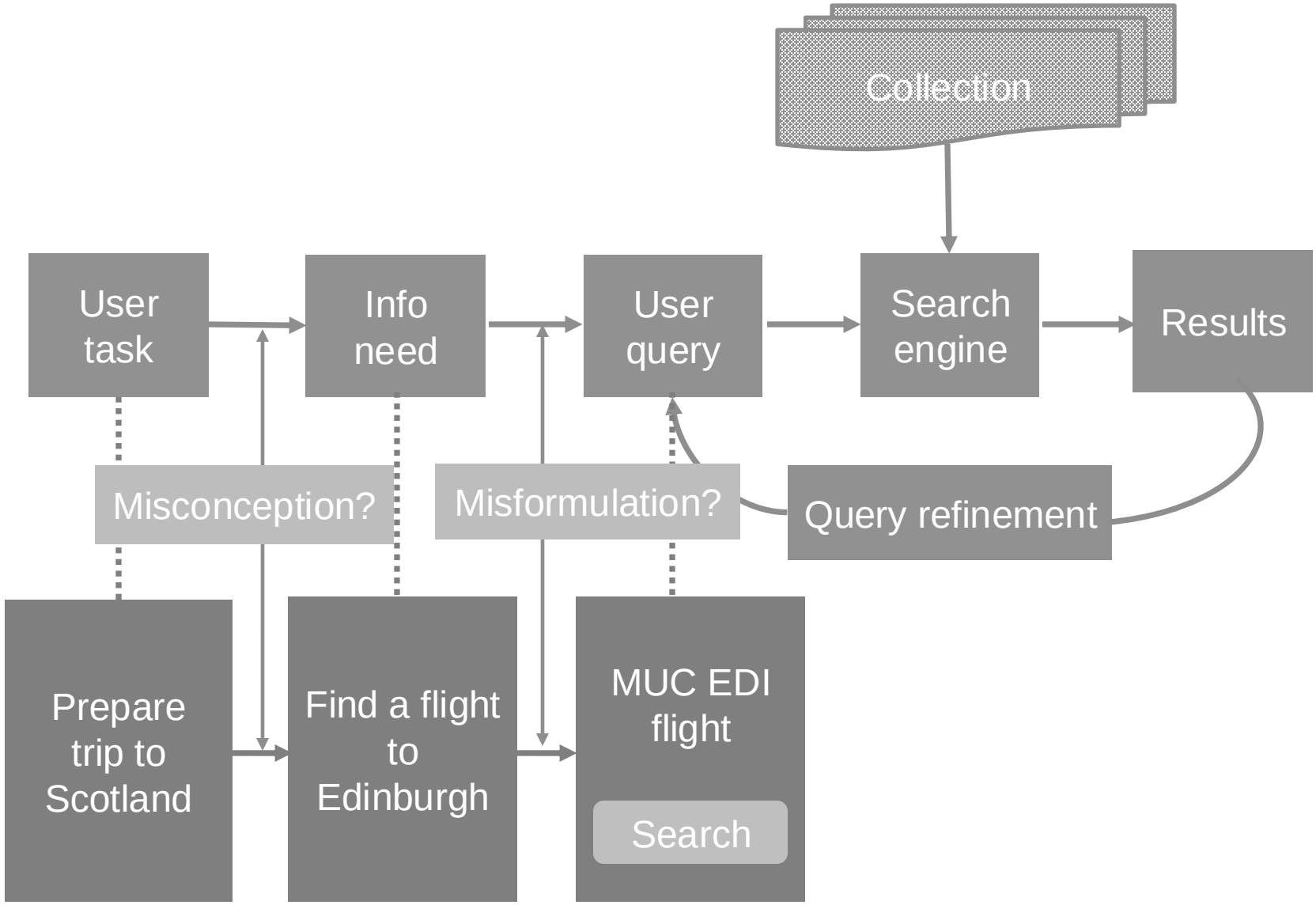
Forbes

May 21, 2018, 12:42am EDT

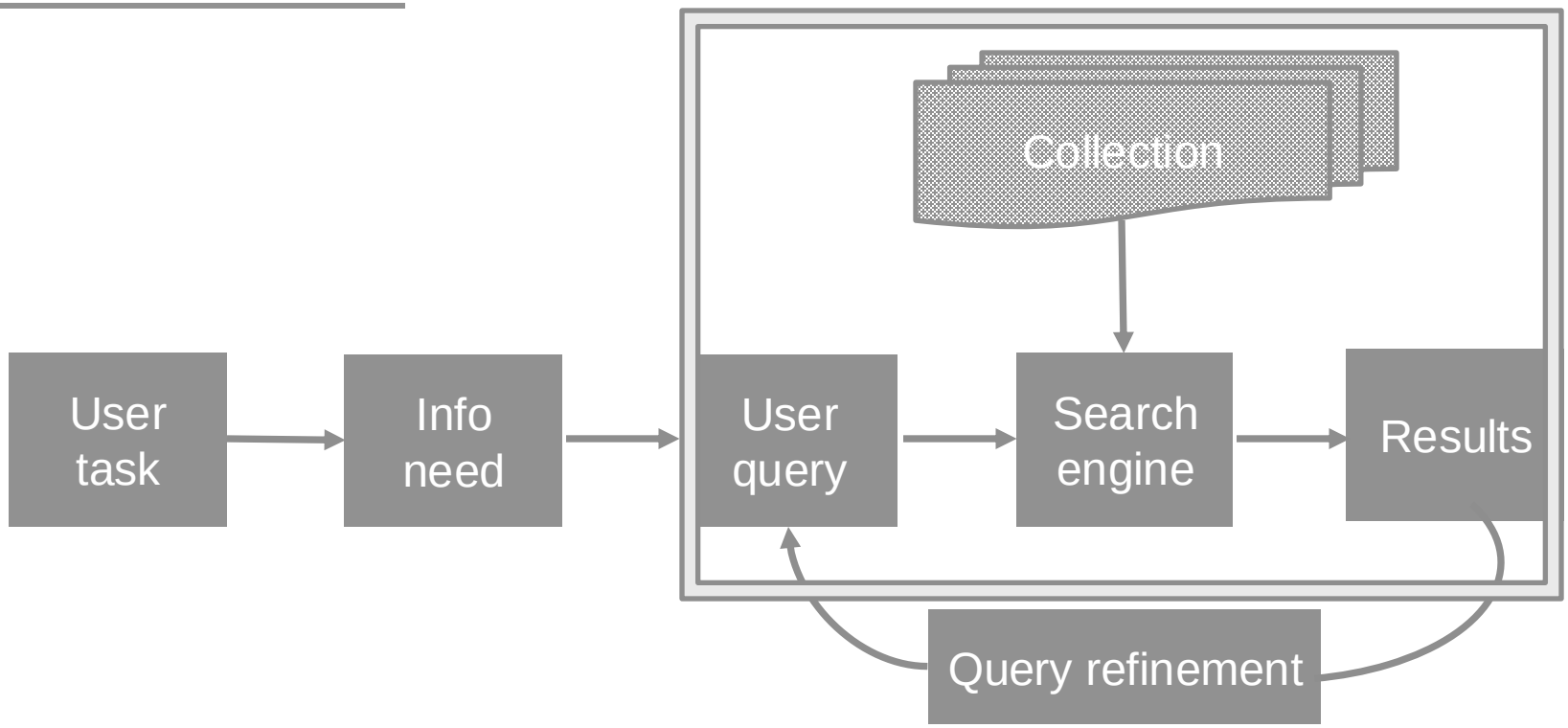
How Much Data Do We Create Every Day? The Mind-Blowing Stats Everyone Should Read

Bernard Marr Contributor
Enterprise Tech

The classic search model



The next sessions



Search engine: Different types of retrieval techniques (Boolean, vector space, probabilistic)

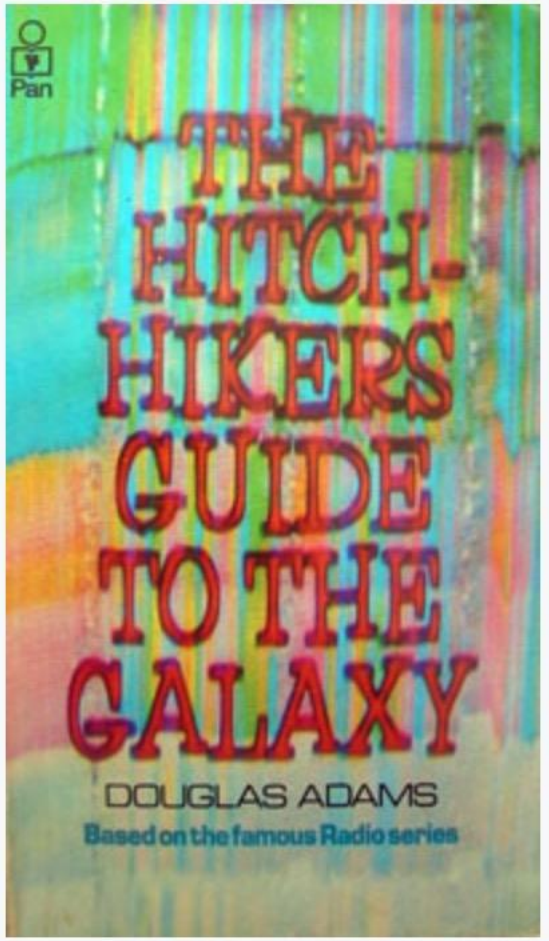
Results: Ranking, evaluation

Term-document incidence matrices

Retrieve information from Douglas Adams' *The Hitchhiker's Guide to the Galaxy.*

For instance, find the books which contain the words 'Arthur' and 'Ford' but not 'Random'.

What would you do?



Term-document incidence matrices

Why is grepping, i.e., a linear scan through the documents, problematic?

- Slow (for large collections)
- More sophisticated search is not feasible, e.g., retrieving terms with close proximity to each other is impossible
- Ranked retrieval is not possible (which criteria do you propose?)

The way out: Indexing!

Term-document incidence matrices

	The Hitch-hiker's Guide to the Galaxy	The Restau-rant at the End of the Universe	Life, the Universe and Every-thing	So Long, and Thanks for all the Fish	Mostly Harmless	And Another Thing...
Arthur	1	1	0	0	0	1
Ford	1	1	0	1	0	0
Zaphod	1	1	0	1	1	1
Trillian	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
Marvin	1	0	1	1	1	1
Random	1	0	1	1	1	0

0 = Douglas Adam's work DOES NOT contain the word

1 = Douglas Adam's work DOES contain the word

Here: Binary term-document incidence matrix – the basis for Boolean IR.

Term-document incidence matrices

	The Hitch-hiker's Guide to the Galaxy	The Restau-rant at the End of the Universe	Life, the Universe and Every-thing	So Long, and Thanks for all the Fish	Mostly Harmless	And Another Thing...
Arthur	1	1	0	0	0	1
Ford	1	1	0	1	0	0
Zaphod	1	1	0	1	1	1
Trillian	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
Marvin	1	0	1	1	1	1
Random	1	0	1	1	1	0

What do we do to answer the query 'Ford AND Zaphod AND NOT Trillian'?

Take the vectors for Ford, Zaphod and Trillian (complemented!)

Term-document incidence matrices

	The Hitch-hiker's Guide to the Galaxy	The Restau-rant at the End of the Universe	Life, the Universe and Every-thing	So Long, and Thanks for all the Fish	Mostly Harmless	And Another Thing...
Arthur	1	1	0	0	0	1
Ford	1	1	0	1	0	0
Zaphod	1	1	0	1	1	1
Trillian	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
Marvin	1	0	1	1	1	1
Random	1	0	1	1	1	0

110100 AND 110111 AND 101111 = 100100

Bitwise AND → Answer to query: 'The Hitchhikers...' and 'So Long...'

Reality check: Bigger collections

Consider N = 1 million documents, each with about 1000 words (2-3 pages)

Avg 6 bytes/word incl. spaces/punctuation → 6GB of data in the collection.

Around M = 500k distinct terms among these.

→ Can't build the matrix!

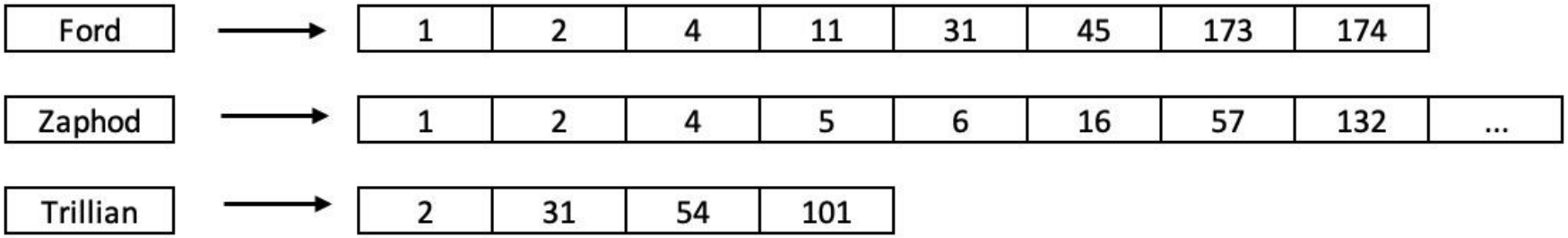
- 500k x 1M matrix has half-a-trillion 0's and 1's
- But it is extremely sparse!

Better representation? We only record the 1's.

Inverted index

First major concept in Information Retrieval.

For each term t, we store a list of all documents that contain t.

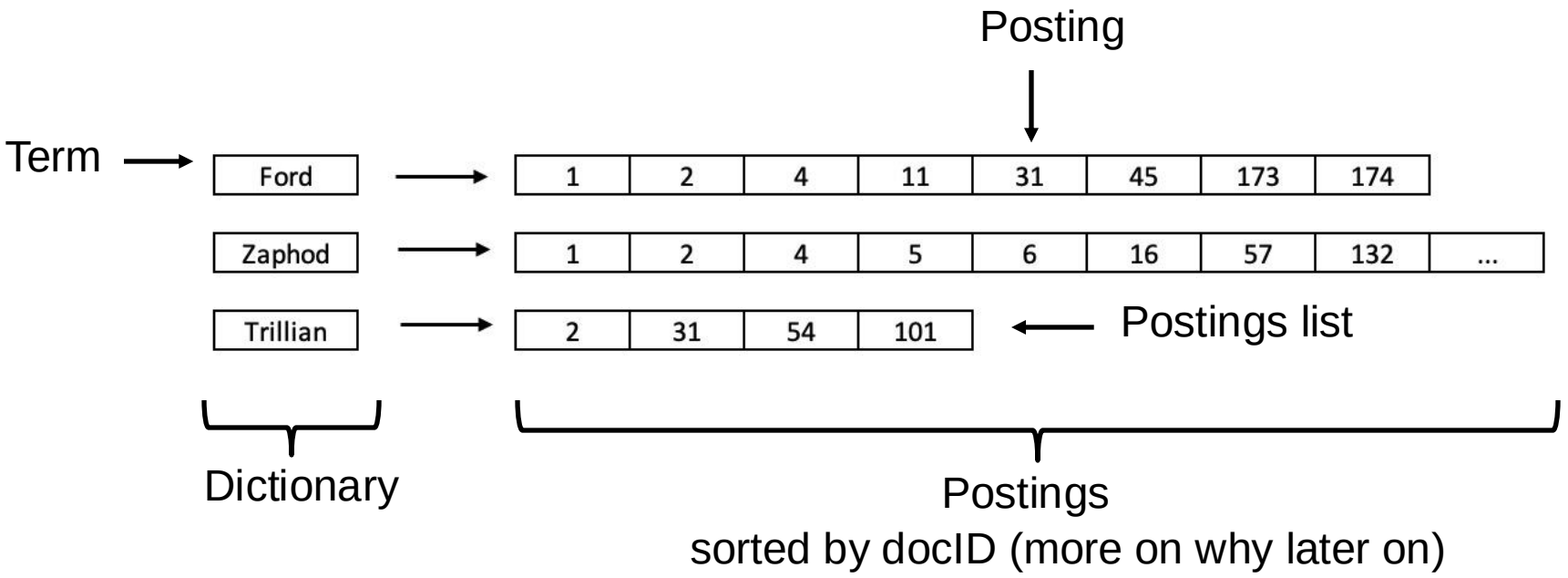


Each document is identified by a docID (document serial number).

Can we use fixed-sized arrays for this? What happens if the word ’Trillian’ is added to document 173?

Inverted index

We need variable-size postings lists



Building an inverted index

1. Collect the documents to be indexed.

2. Tokenize the text (each document is converted into a set of terms)

3. Do linguistic processing of tokens.

4. Index the documents by creating an inverted index.

Building an inverted index

1. **Collect the documents to be indexed.**
2. Tokenize the text (each document is converted into a set of terms)
3. Do linguistic processing of tokens.
4. Index the documents by creating an inverted index.

Building an inverted index

1. Collect the documents to be indexed.

1.1. Obtaining a character sequence

Digital documents are typically bytes in a file or on a web sever.

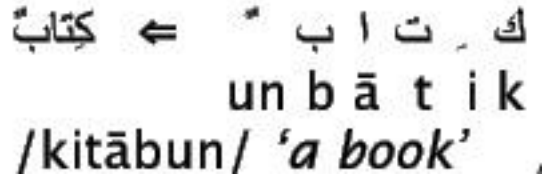
Convert a byte sequence into a linear sequence of characters (trivial for English, but tricky for writing systems like Arabic).

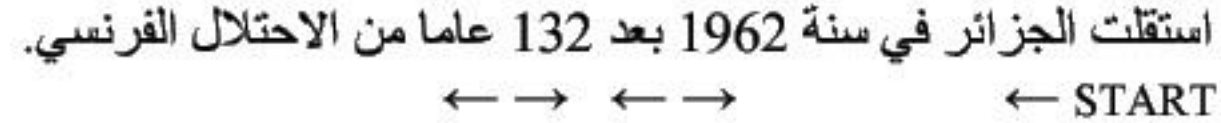
In the case of binary representations (e.g., MS Office documents) → decoder needed.

Building an inverted index

1. Collect the documents to be indexed.

1.1. Obtaining a character sequence





'Algeria achieved its independence in 1962 after 132 years of French occupation.'

Tricky for languages like Arabic.

Building an inverted index

1. Collect the documents to be indexed.

1.2. Choosing a document unit

Individual emails as one document, but what about an individual book?
What granularity should we apply?

Eventually: up to the developer and their knowledge of

- document collection
- the users
- their likely information need and
- their usage patterns

Building an inverted index

1. Collect the documents to be indexed.

2. **Tokenize the text (each document is converted into a set of terms).**

3. Do linguistic processing of tokens.

4. Index the documents by creating an inverted index.

Building an inverted index

2. Tokenize the text (each document is converted into a set of terms).

(aka, chop the character sequence into pieces, called tokens)

Token = an instance of a character sequence

Type = class of all tokens containing the same character sequence

Term = (perhaps normalized) type that is included in the IR systems dictionary.

What are the types and tokens in ‘A rose is a rose is a rose’?

Building an inverted index

2. Tokenize the text (each document is converted into a set of terms).

Tricky question: What tokens to use?

Split at whitespace and throw out punctuation?

Tokenization is language-specific!

Tokenize `Mr. O'Neill things that the boys' stories about Chile's capital aren't amusing.'

And what about `co-education' versus 'Hewlett-Packard' versus 'the hold-him-back-and-drag-him-away maneuver'? And 'Computerlinguistik', 'Lebensversicherungsgesellschaftsangestellter'?

Building an inverted index

1. Collect the documents to be indexed.

2. Tokenize the text (each document is converted into a set of terms).

3. **Do linguistic processing of tokens.**

4. Index the documents by creating an inverted index.

Building an inverted index

3. Do linguistic processing of tokens.

Extremely common words are of little value in helping select documents → remove them (*stopword removal*)

They are not part of the index.

IR systems started with long stopword lists, then moved to very small stopword lists.

Web search: no stopword lists at all, instead: term weighting (Week 3).

Building an inverted index

3. Do linguistic processing of tokens.

Equivalence classing: `U.S.A.' and `USA' match, as do 'anti-discriminatory' and 'antidiscriminatory'

Stemming and lemmatization → same root for `authorize' and `authorization'

Case-folding: reduce all letters to lower case. Problem: proper nouns ('General Motors', 'The Associated Press', 'Bush', 'Fed')

Building an inverted index

3. Do linguistic processing of tokens.

Exercise (IIR, p. 33): The following pairs of words are stemmed to the same form by the Porter stemmer. Which pairs, would you argue, should not be conflated? Give your reasoning.

a) abandon/abandonment
b) absorbency/absorbent
c) marketing/markets
d) university/universe
e) volume/volumes

Building an inverted index

1. Collect the documents to be indexed.

2. Tokenize the text (each document is converted into a set of terms).

3. Do linguistic processing of tokens.

4. **Index the documents by creating an inverted index.**

Building an inverted index

4.1: Get the token sequence

Sequence of (modified token, docID) pairs.

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:

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

Building an inverted index

4.2: Sort the tokens

Here, alphabetically.

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:

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

⟹

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

Building an inverted index

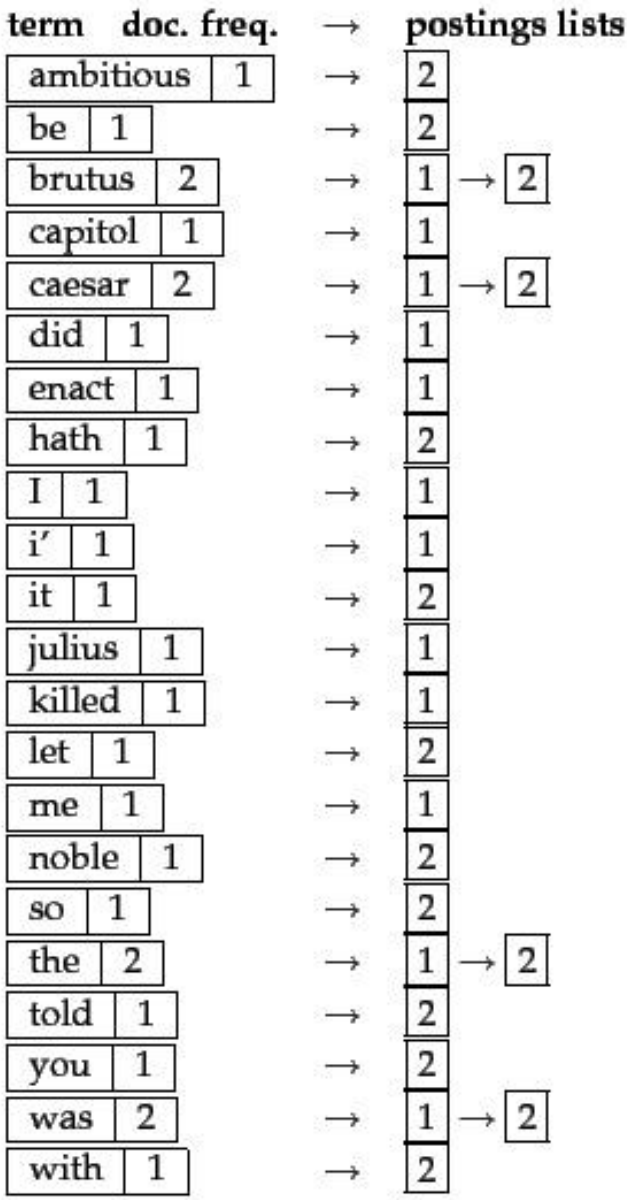
4.3: Get dictionaries and postings

Multiple term entries in a single document are merged.

Split into dictionaries and postings.

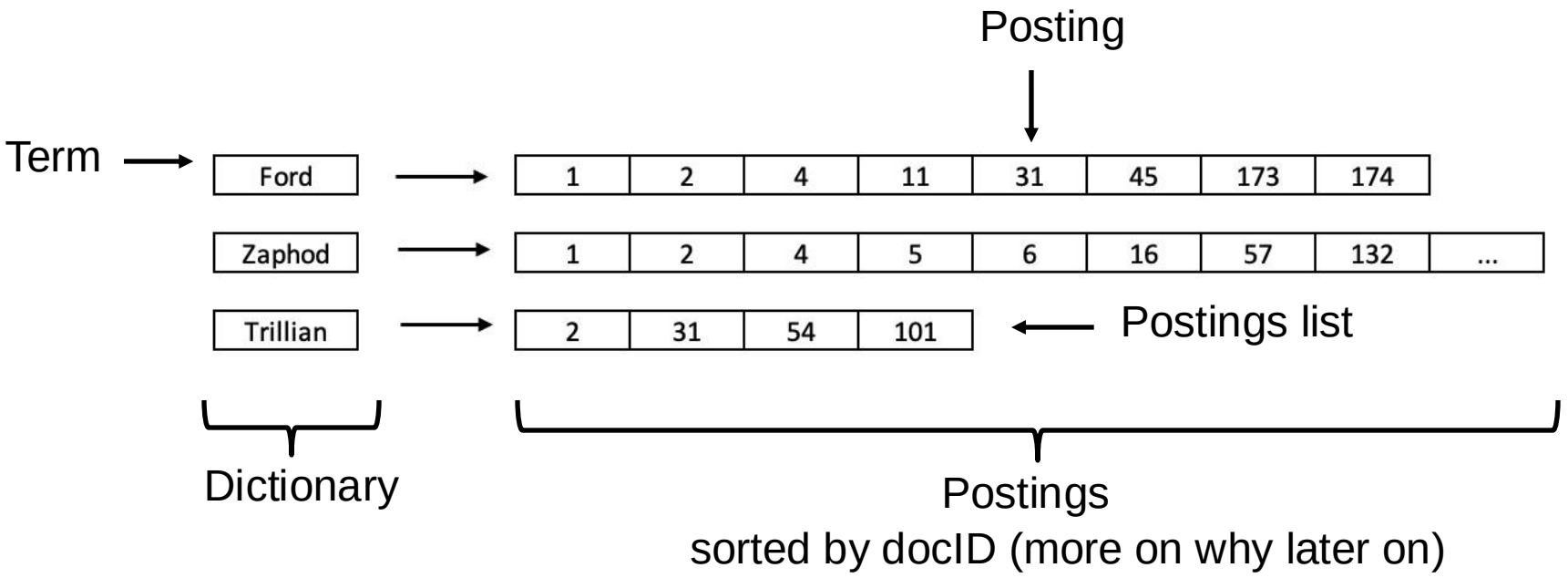
Add document frequency (the number of documents which contain each term).

Why frequency?
Will see later/next week.



Inverted index

We need variable-size postings lists

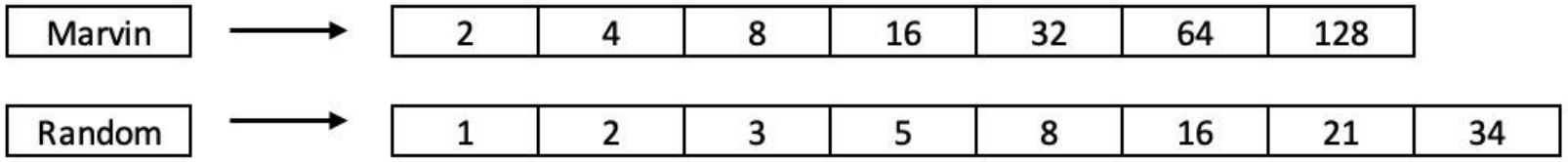


Query processing with an inverted index

Focus now: How can we process a query using an inverted index?

Example: ‘Marvin AND Random’

- Locate ‘Marvin’ in the dictionary and retrieve its postings.
- Locate ‘Random’ in the dictionary and retrieve its postings.
- Merge the two postings (intersect the document sets):

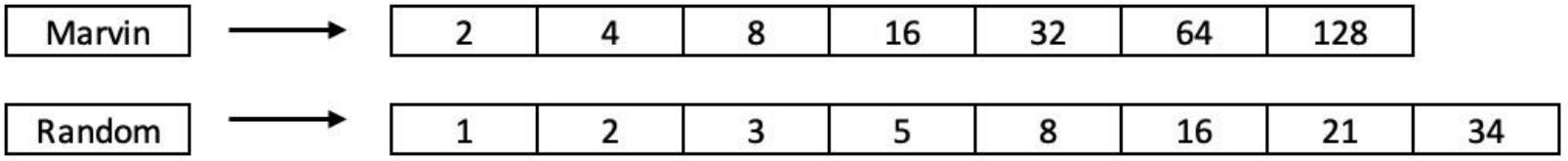


Merging postings

Maintain pointers into both lists and we walk through the two postings simultaneously, comparing the docID pointed to by both pointers.

If the list lengths are x and y, the merge takes O(x+y) operations (linear growth).

Crucial: postings are sorted by docID.

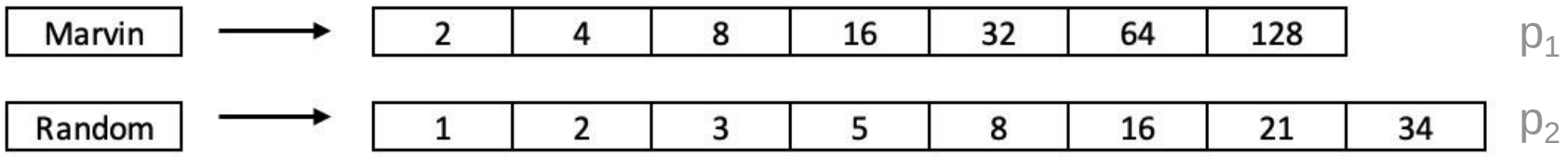


Merging postings

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

Figure 1.6: Algorithm for the intersection of two postings lists p_1 and p_2.

Query: 'Marvin AND Random'

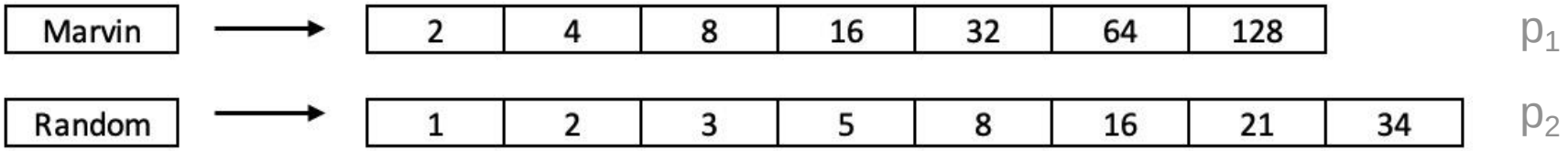


Merging postings

On your own:

Adapt the merge algorithm for the query

’Marvin AND NOT Random’



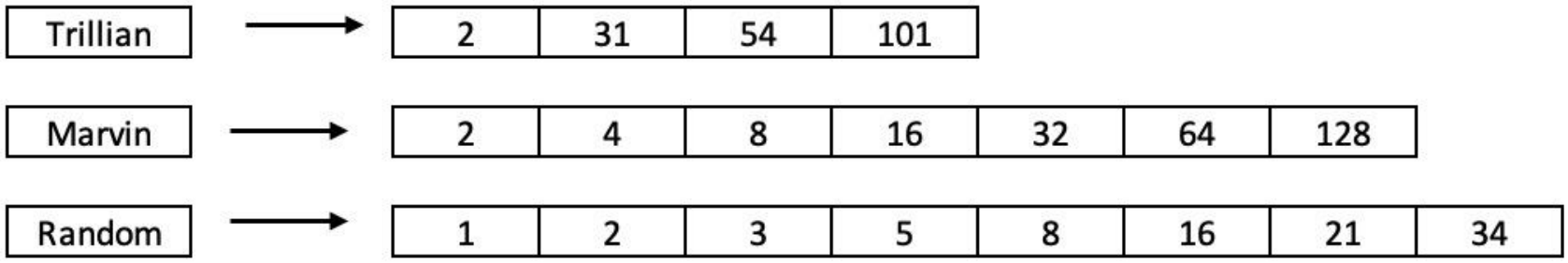
Query optimization

What is the best order for query processing?

Consider a query that is an AND of *n* terms.

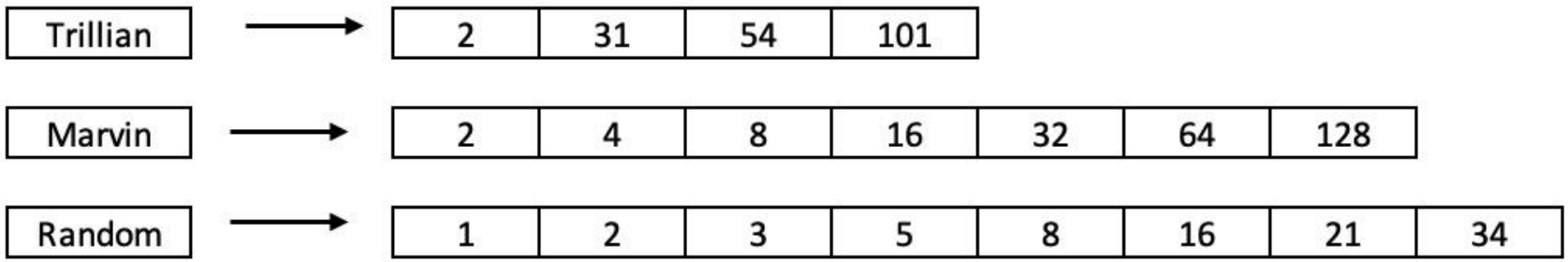
For each of the *n* terms, get its postings, then AND them together.

Query: ‘Trillian AND Marvin AND Random’



Query optimization

Process in order of increasing frequency (start with the smallest set then keep cutting further)



Execute the query as ‘(Trillian AND Marvin) AND Random’

This is why we kept document frequency in the dictionary!

Query optimization

(IIR, Ex 1.7) Recommend a processing order for the query

‘(tangerine OR trees) AND
(marmalade OR skies) AND
(kaleidoscope OR eyes)’

given the term frequencies on the right.

Term	Freq
eyes	213.312
kaleidoscope	87.009
marmalade	107.913
skies	271.658
tangerine	46.653
trees	316.812

Which two terms should we process first?
Union the postings list and then compare.

Boolean queries

Boolean queries are using AND, OR and NOT to join query terms.

- Each document is a set of words
- A document matches the condition or not (precise!)

Boolean retrieval models were the primary commercial retrieval tool for three decades.

Many search systems still rely on Boolean retrieval: email, library catalogues, macOS spotlight.

Boolean queries

www.westlaw.com, the largest commercial legal search service in the US.

Tens of terabytes of data, ~700.000 users

Majority of users still use Boolean queries.

Exemplary information need: What is the statute of limitations in cases involving the federal tort claims act?

Query: “LIMIT! /3 STATUTE ACTION /S FEDERAL /2 TORT /3 CLAIM”
(/3 = within 3 words, /S = in same sentence, SPACE is disjunction)

Phrase queries

We want to be able to answer queries such as *University of Passau* as a phrase.

→ The sentence “I went to university at Passau” is not a match.

The concept of *phrase queries* has proven to be easily understood by users (one of the few advanced search ideas that actually works!)

→ It no longer suffices to store only <term : doc> entries.

Bigram phrases

Index every consecutive pair of terms (a ‘bigram’) as a phrase.

What bigrams does the following sentence generate?

”I went to university at Passau”

These are the new dictionary terms!

Longer phrase queries

Longer phrase queries can be processed by breaking them down.

E.g., ‘University of Passau Bavaria’ can be broken into the Boolean bigram query

‘University of’ AND ‘of Passau’ AND ‘Passau Bavaria’

What’s the problem here?

- False positives! The documents matching the query do not necessarily contain the original four-word phrase.

→ create a positional index

Positional indexes

In the postings, store for each term the position(s) in which tokens of it appear.

<term, doc. freq.;
doc1: position1, position2, …;
doc2: position1, position2, …;
etc.>

<be, 993427;
1: 7, 18, 33, 72, 86, 231;
2: 3, 149;
4: 17, 191, 291, 430, 434;
5: 363, 367>

Which of the docs could contain `to be or not to be'?

Processing with positional indexes

E.g., Shakespeare's 'to be or not to be'

Extract inverted index entries for each term 'to', 'be', 'or', 'not',

Merge the doc:position lists to enumerate all positions with 'to be or not to be'

to: 2:1,17,74,222,551; 4:8,16,190,429,433; 7:13,23,191; ...

be: *1*:17,19; *4*:17,191,291,430,434 *5*:14,19,101; ...

Positional index size

Need an entry for each occurrence of a term, not just one per document → substantial increase in storage required.

And: Index size then depends on average document size.

Consider a term with a frequency of 0.1%:

Document size (terms)	Postings	Positional postings
1000	1	1
100.000	1	100

Combination schemes

These two approaches can be profitably combined.

- Positional indices for non-compounding structures.
- Phrases like ‘Michael Jackson’ and ‘Brittney Spears’ are encoded as bigram postings.

Williams et al. 2004:

- A typical web query architecture was executed in ¼ of the time when using just a positional index.
- But: 26% more space than having a positional index alone.

Wrap-up of today's lecture

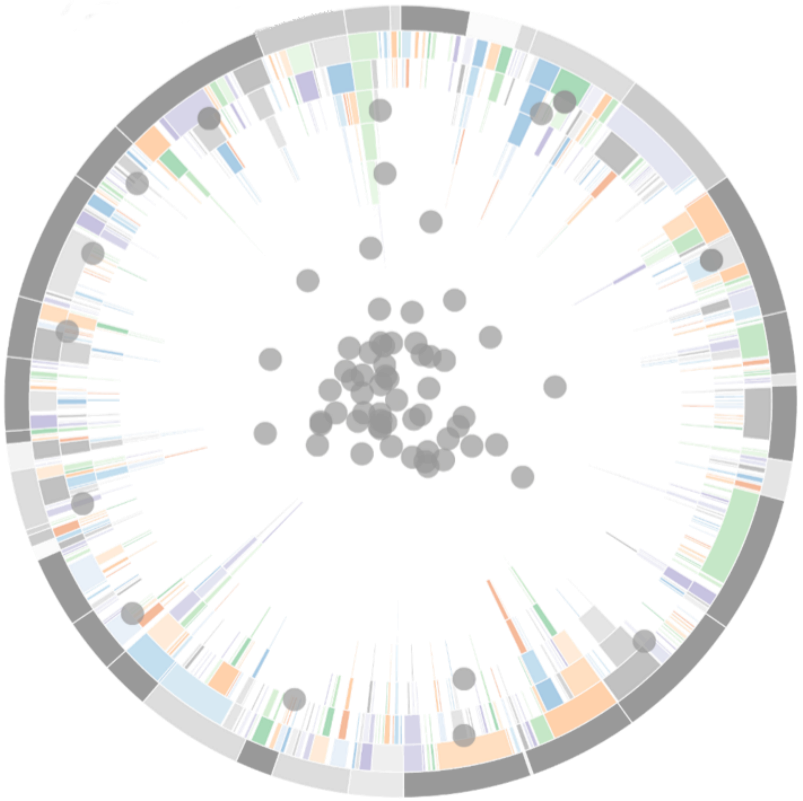
- A few important concepts of IR:
  - Data
  - Information need
  - Large collections
- The classic search model
- Term-document incidence matrices
- The inverted index
- Query processing

Cluster of Excellence
The Politics of Inequality

Steinbeis Transfer Center
Linguistic Data Analysis

UNIVERSITÄT PASSAU

Thank you.
Questions?
Comments?



Annette Hautli-Janisz, Prof. Dr.
cornlp-teaching@uni-passau.de