



THE UNIVERSITY
of EDINBURGH

Text Technologies for Data Science

INFR11145

Indexing

Instructor:
Walid Magdy

07-Oct-2026

1

Lecture Objectives

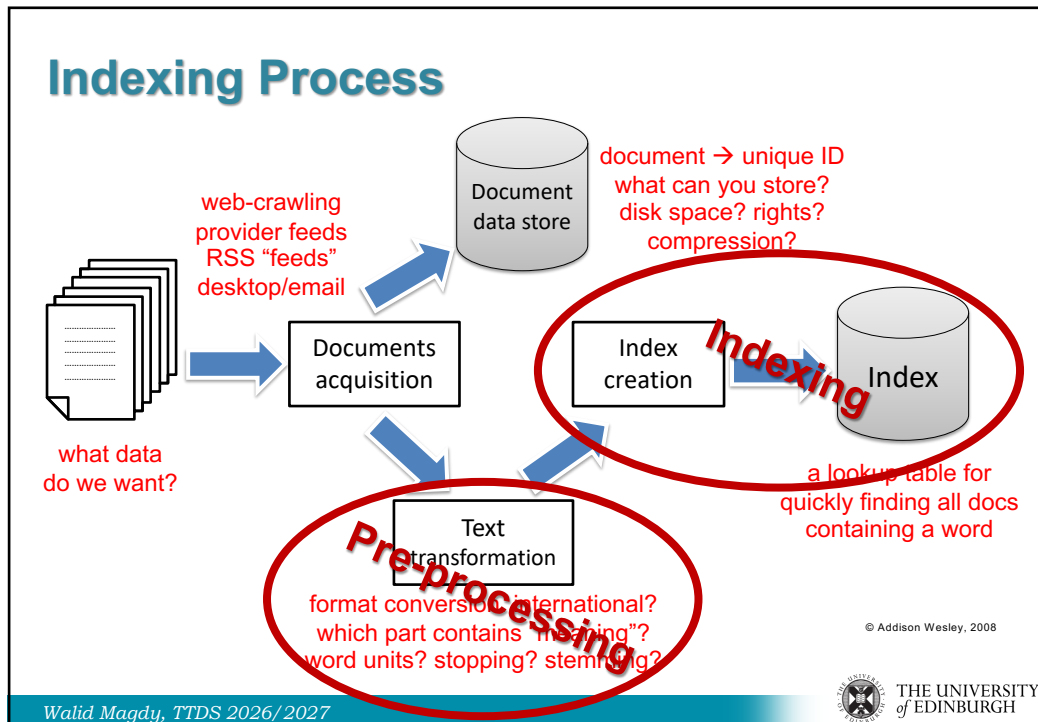
- Learn about and implement
- Boolean search
- Inverted index
- Positional index

Walid Magdy, TTDS 2026/2027

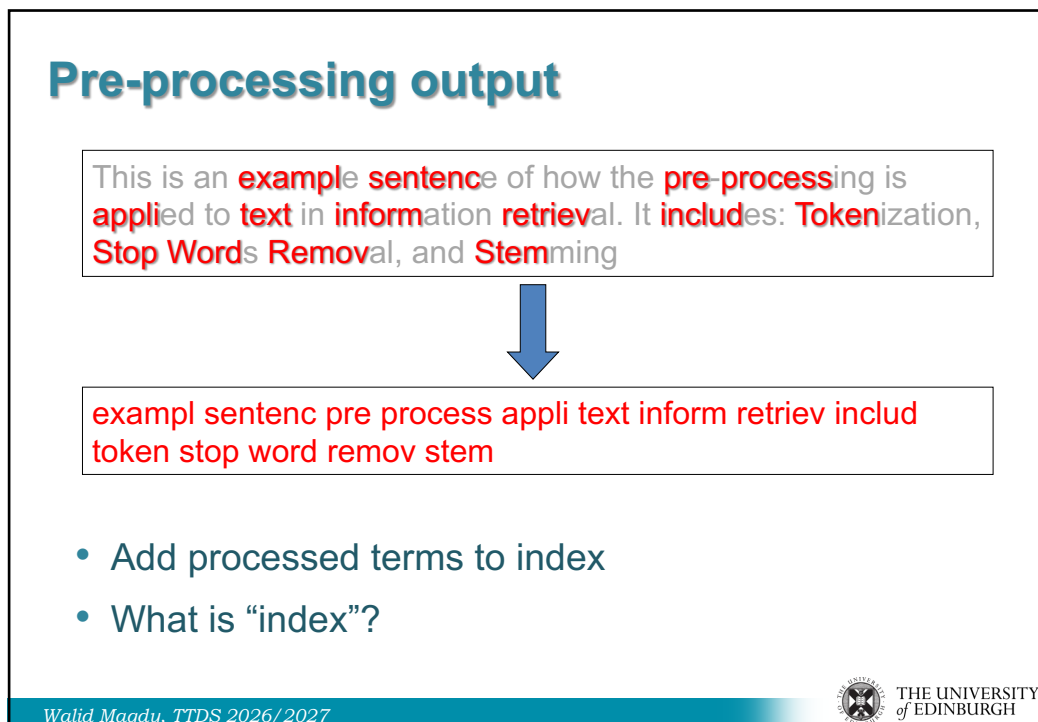


THE UNIVERSITY
of EDINBURGH

2



3



4

Index

- How to match your term in non-linear time?
- Find/Grep:
Sequential search for term
- Index:
Find term locations immediately

Walid Magdy, TTDS 2026/2027



THE UNIVERSITY
of EDINBURGH

5

Book Index

Index

- absolute error, 437
accuracy, 359
ad hoc search, 3, 280, 423
adaptive filtering, 425
adversarial information retrieval, 294
advertising, 218, 371
 classifying, 371
 contextual, 218–221
agglomerative clustering, 375
anchor text, 21, 56, 105, 280
API, 439, 461
architecture, 13–28
authority, 21, 111
automatic indexing, 400

background probability, *see* collection
 probability
bag of words, 345, 451
Bayes classifier, 245
Bayes Decision Rule, 245
Bayes' Rule, 246, 343
Bayes' rule, 342
Bayesian network, 268
bibliometrics, 120
bidding, 218
bigram, 100, 253
BigTable, 57

binary independence model, 246
blog, 111
BM25, 250–252
BM25F, 294
Boolean query, 235
Boolean query language, 24
Boolean retrieval, 235–237
boosting, 448
BPREF, 322
brute force, 331
burstiness, 254

caching, 26, 181
card catalog, 400
case folding, 87
case normalization, 87
categorization, *see* classification
CBIR, *see* content-based image retrieval
character encoding, 50, 119
checksum, 60
Chi-squared measure, 202
CJK (Chinese-Japanese-Korean), 50, 119
classification, 3, 339–373
 faceted, 224
 monothetic, 223, 374
 polythetic, 223, 374
classifier, 21

clickthrough, 6, 27, 207, 285, 306
CLIR, *see* cross-language information
 retrieval
cluster hypothesis, 389
cluster-based retrieval, 391
clustering, 22, 222–225, 339, 373
co-occurrence, 74, 191
code page, 50
collaborative filtering, 432
collaborative search, 420
collection, 3
collection language model, 256
collection probability, 256, 440
collocation, 74
color histogram, 473
combining evidence, 267–283
combining searches, 441
CombMNZ, 441
community-based question answering,
 415
complete-link clusters, 379
compression, 54
 lossless, 141
 lossy, 142
conditional random field, 122
conflation, *see* stemming
connected component, 192
content match, 371
content-based image retrieval, 473
context, 115, 201, 211–214
context vector, 206, 464
contingency table, 248
controlled vocabulary, 199, 401
conversion, 49
coordination level match, 257
corpus, 6
cosine correlation, 239
coverage, 8
CQA, 415

crawler, 17, 32
cross-language information retrieval, 226
cross-lingual search, *see* cross-language
 information retrieval
cross-validation, 331

Damerau-Levenshtein distance, 194
dangling link, 107
data mining, 113
database system, 459
DCG, *see* discounted cumulative gain
deep Web, 41, 448
delta encoding, 144
dendrogram, 375
desktop search, 3, 46
Dice's coefficient, 192
digital reference, 447
Dirichlet smoothing, 258
discounted cumulative gain, 319
discriminative model, 281, 360
distance measure, 374
distributed hash table, 445
distributed information retrieval, 438
distribution, 23
divisive clustering, 375
document, 2
document conversion, 18
document crawler, 17
document data store, 19
document distribution, 180
document slope curve, 64
document statistics, 22
document structure, 101, 269, 459–466
document summary, 215
downcasing, 87
dumping, 366
duplicate documents, 60
dwell time, 27
dynamic page, 42

Walid Magdy, TTDS 2026/2027



THE UNIVERSITY
of EDINBURGH

6

Indexing

- Search engines vs PDF find or grep?
 - Infeasible to scan large collection of text for every “search”
 - Find section that has: “UK and Scotland and Money”?!
- Book Index
 - For each word, list of “relevant” pages
 - Find topic in sub-linear time
- IR Index:
 - Data structure for fast finding terms
 - Additional optimisations could be applied

Walid Magdy, TTDS 2026/2027



7

Document Vectors

- Represent documents as vectors
 - Vector $\rightarrow$ document, cell $\rightarrow$ term
 - Values: term frequency or binary (0/1)
 - All documents $\rightarrow$ collection matrix

2	1	0	2	0	0	1	← D1: He likes to wink, he likes to drink
1	3	0	1	0	0	0	← D2: He likes to drink, and drink, and drink
1	1	1	1	0	1	0	← D3: The thing he likes to drink is ink
1	1	1	1	1	0	0	← D4: The ink he likes to drink is pink
1	1	1	1	1	0	1	← D5: He likes to wink, and drink pink ink

number of occurrence of
a term in a document

Walid Magdy, TTDS 2026/2027



8

Inverted Index

- Represent terms as vectors
 - Vector $\rightarrow$ term, cell $\rightarrow$ document
 - Transpose of the collection matrix
 - Vector: inverted list

he	drink	ink	likes	pink	thing	wink	
2	1	0	2	0	0	1	← D1: He likes to wink, he likes to drink
1	3	0	1	0	0	0	← D2: He likes to drink, and drink, and drink
1	1	1	1	0	1	0	← D3: The thing he likes to drink is ink
1	1	1	1	1	0	0	← D4: The ink he likes to drink is pink
1	1	1	1	1	0	1	← D5: He likes to wink, and drink pink ink

Boolean Search

- Boolean: exist / not-exist
- Multiword search: logical operators (AND, OR, NOT)
- Example
 - Collection: search Shakespeare's Collected Works
 - Boolean query: Brutus AND Caesar AND NOT Calpurnia
- Build a **Term-Document Incidence Matrix**
 - Which term appears in which document
 - Rows are terms
 - Columns are documents

Collection Matrix

	Documents					
	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

1 if **document** contains **term**, 0 otherwise

Query: Brutus AND Caesar AND NOT Calpurnia
 Apply on rows: **110100** AND **110111** AND **!(010000) = 100100**

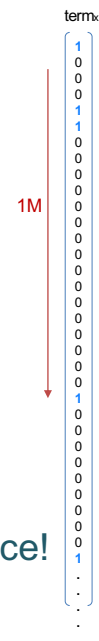
Walid Magdy, TTDS 2026/2027

THE UNIVERSITY of EDINBURGH

11

Bigger collections?

- Consider $N = 1$ million documents, each with about 1000 words.
- $n = 1M \times 1K = 1B$ words
 $\rightarrow$ Heap's law $\rightarrow v \approx 500K$
- Matrix size = 500K unique terms x 1M documents
 = 0.5 trillion 0's and 1's entries!
- If all words appear in many documents
 $\rightarrow \max\{\text{count}(1's)\} = N * \text{doc. length} = 1B$
- Actually, from Zip's law $\rightarrow 250k$ terms appears once!
- Collection matrix is extremely **sparse**. (mostly 0's)



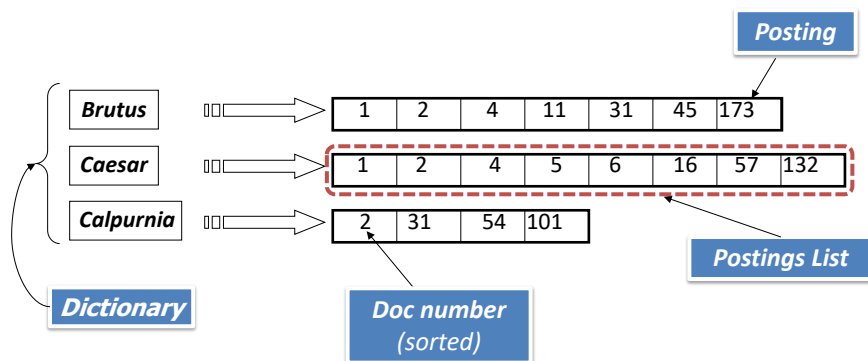
Walid Magdy, TTDS 2026/2027

THE UNIVERSITY of EDINBURGH

12

Inverted Index: Sparse representation

- For each term t , we must store a list of all documents that contain t .
 - Identify each by a **docID**, a document serial number

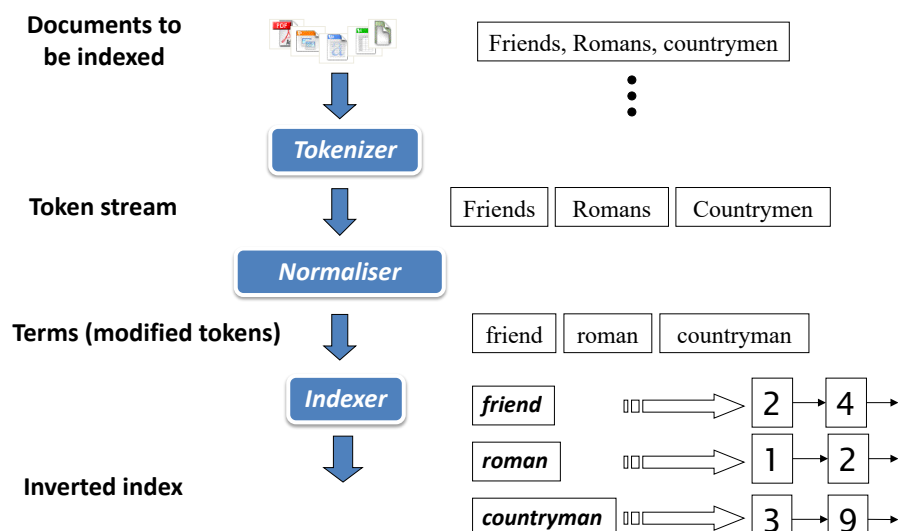


Walid Magdy, TTDS 2026/2027



13

Inverted Index Construction



Walid Magdy, TTDS 2026/2027



14

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

Walid Magdy, TTDS 2026/2027



THE UNIVERSITY
of EDINBURGH

15

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

Walid Magdy, TTDS 2026/2027



THE UNIVERSITY
of EDINBURGH

16

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

Walid Magdy, TTDS 2026/2027



17

Inverted Index: matrix → postings

he	drink	ink	likes	pink	think	wink	
2	1	0	2	0	0	1	← D1: He likes to wink, he likes to drink
1	3	0	1	0	0	0	← D2: He likes to drink, and drink, and drink
1	1	1	1	0	1	0	← D3: The thing he likes to drink is ink
1	1	1	1	1	0	0	← D4: The ink he likes to drink is pink
1	1	1	1	1	0	1	← D5: He likes to wink, and drink pink ink

he	→	1	2	3	4	5
drink	→	1	2	3	4	5
ink	→	3	4	5		
pink	→	4	5			
thing	→	3				
wink	→	1	5			

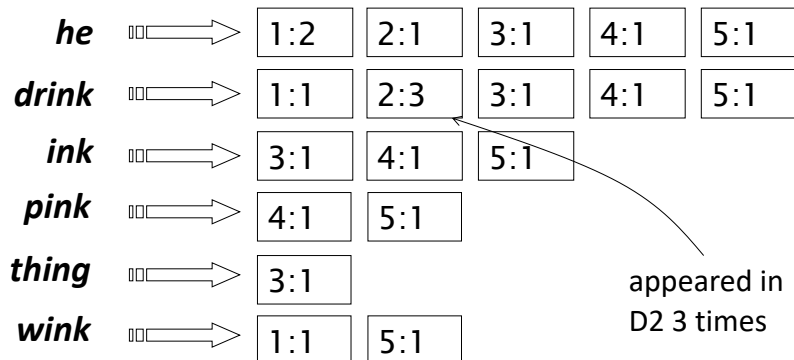
Walid Magdy, TTDS 2026/2027



18

Inverted Index: with frequency

- Boolean: term $\rightarrow$ DocIDs list
- Frequency: term $\rightarrow$ tuples (DocID,count(term)) lists



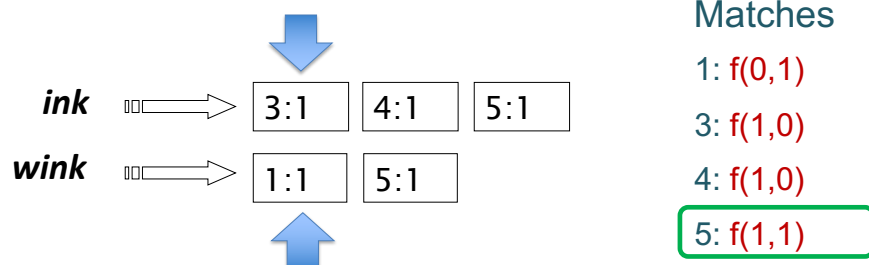
Walid Magdy, TTDS 2026/2027



19

Query Processing

- Find documents matching query {ink **AND** wink}
 1. Load inverted lists for each query word
 2. Merge two postings lists $\rightarrow$ **Linear merge**
- Linear merge $\rightarrow O(n)$
 n : total number of posts for all query words



Walid Magdy, TTDS 2026/2027



20

Phrase Search

- Find documents matching query “pink ink”
 - Find document containing both words
 - Both words has to be a phrase

- Bi-gram Index:

He likes to wink, and drink pink ink Convert to bigrams →

He_likes likes_to to_wink wink_and and_drink drink_pink pink_ink

- Bi-gram Index, issues:

- Fast, but index size will explode!
- What about trigram phrases?
- What about proximity? “ink is pink”

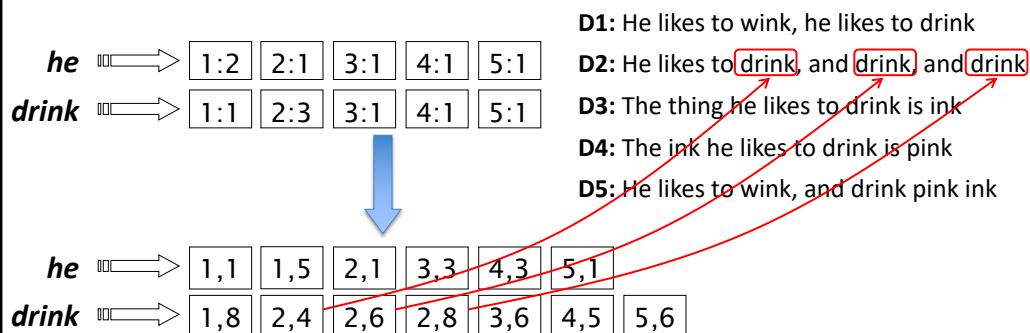
Walid Magdy, TTDS 2026/2027



21

Proximity Index

- Terms positions is embedded to the inv. Index
 - Called proximity/positional index
 - Enables phrase and proximity search
 - Tuples (DocID, term position)



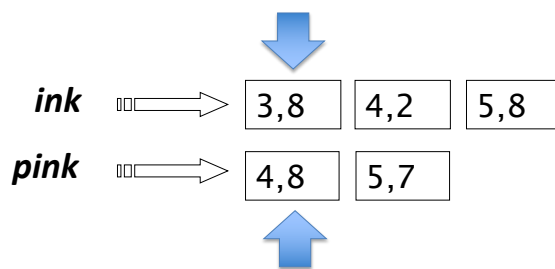
Walid Magdy, TTDS 2026/2027



22

Query Processing: Proximity

- Find documents matching query “pink ink”
 - Use **Linear merge**
 - Additional step: check terms positions
- Proximity search:**
 $pos(term1) - pos(term2) < |w| \rightarrow \#5(pink, ink)$



Matches

3: $f(1,0) = 0$

4: $f(1,1) = ? =$
 $pos(ink) - pos(pink) == 1?$

5: $f(1,1) = ? =$
 $pos(ink) - pos(pink) == 1?$

Query Processing: Simple implementation

- For each term in query → retrieve **posting list**
- AND** → Intersection ($\cap$) between posting lists
- OR** → Union ($\cup$) of posting lists
- NOT** → inverse of posting list (all docs not in the list)
 - Usually only combined with AND operator.
- Phrase search** → AND + check $pos2 - pos1 == 1$
- Proximity search** → AND + check $|pos2 - pos1| \leq n$

Proximity search: data structure

- Possible data structure:

```
<term: df;
  DocNo: pos1, pos2, pos3
  DocNo: pos1, pos2, pos3
  ..... >
```
- Example:

```
<scotland: 25;
  94: 212
  351: 23,34,1354,1779,1838
  370: 981
  3330: 115
  3334: 121,316
  3532: 59,111,162,265 ...>
```

Practical

Summary

- Document Vector
- Term Vector
- Inverted Index
- Collection Matrix
- Posting
- Proximity Index
- Query Processing → Linear merge

Walid Magdy, TTDS 2026/2027



27

Resources

- Textbook 1: Intro to IR, Chapter 1 & 2.4
- Textbook 2: IR in Practice, Chapter 5
- [Lab 2](#)

Walid Magdy, TTDS 2026/2027



28