INFORMATION RETRIEVAL

Week 11 – Probabilistic Retrieval

Severin Mills

Today

1

Semester Recap

2

Theory

- Probabilistic Retrieval

3

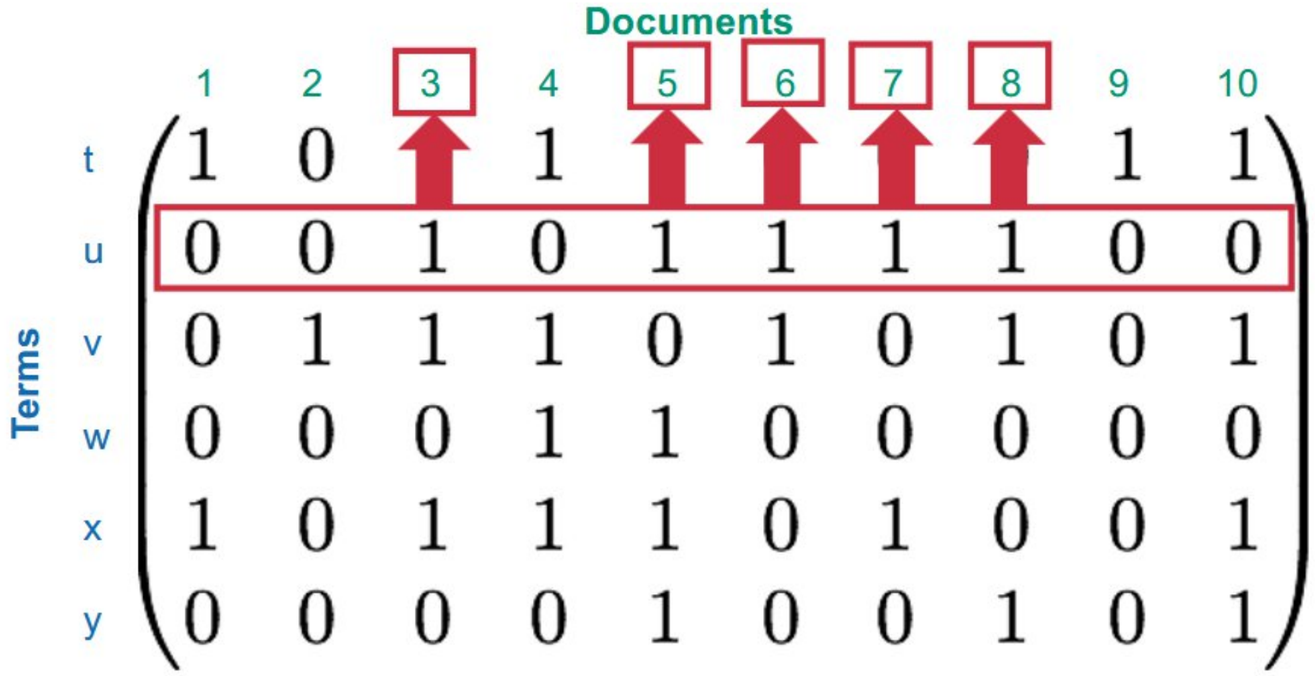
Kahoot

Exercise 10: Probabilistic Retrieval

Incidence Matrix

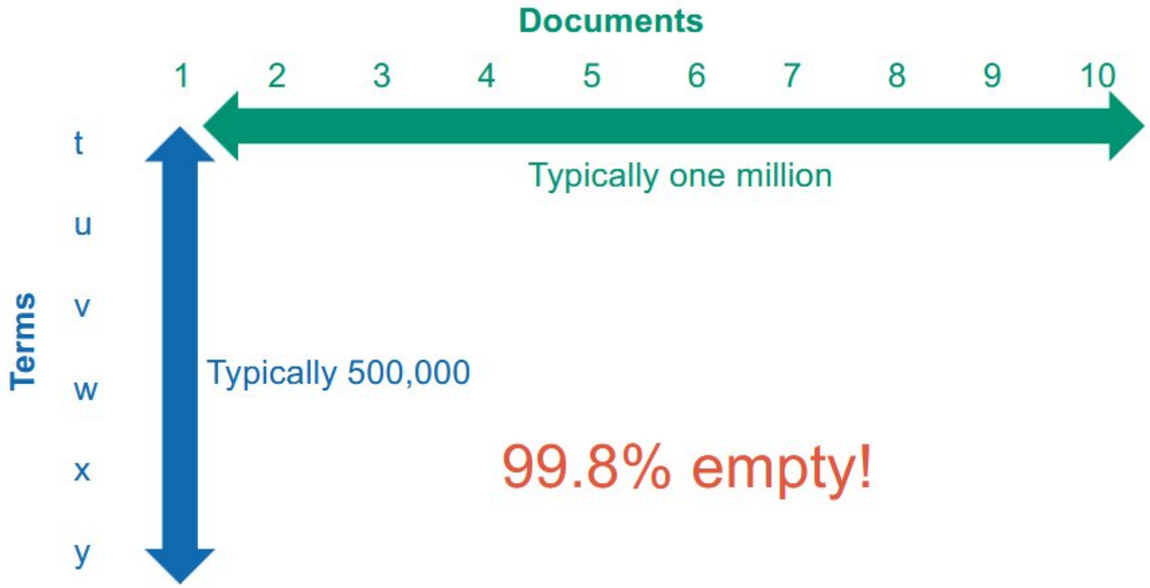
0: Term is **not** in document

1: Term is in document



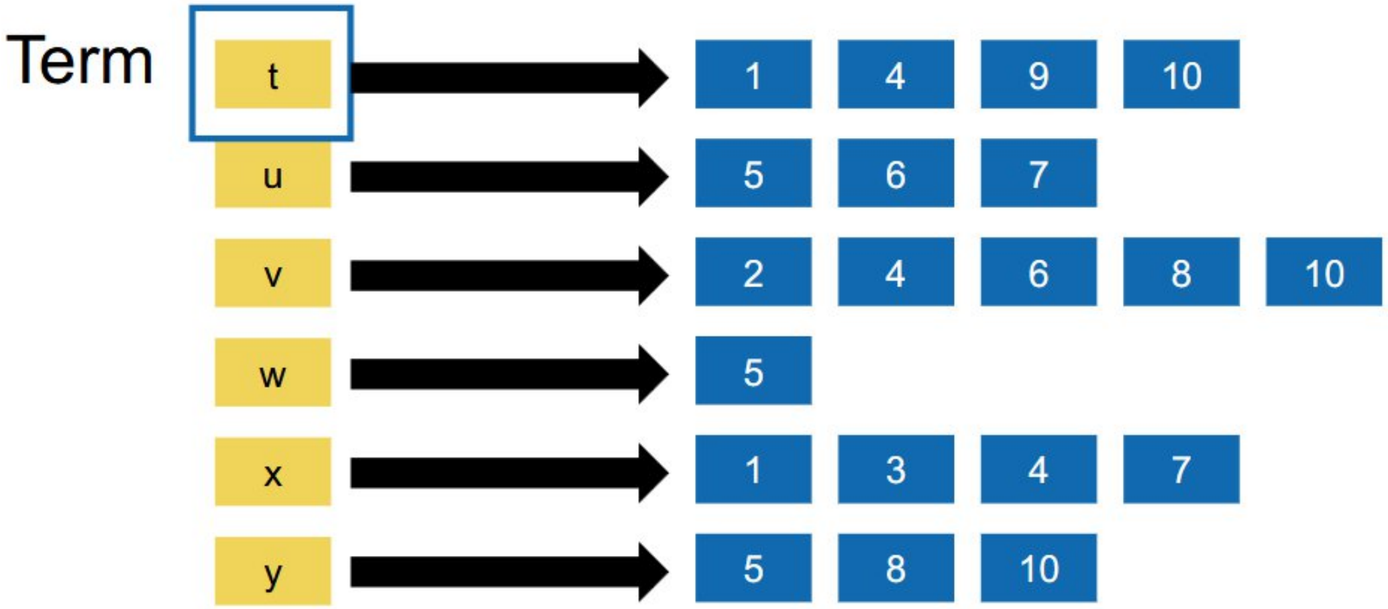
Incidence Matrix

Very inefficient storage usage!



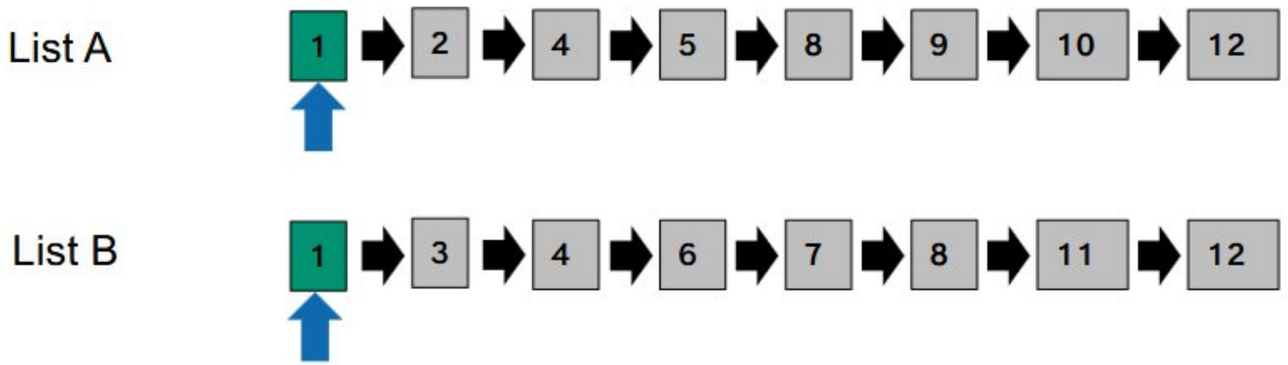
Inverted Index

No storage usage for the zeroes, store documents in lists



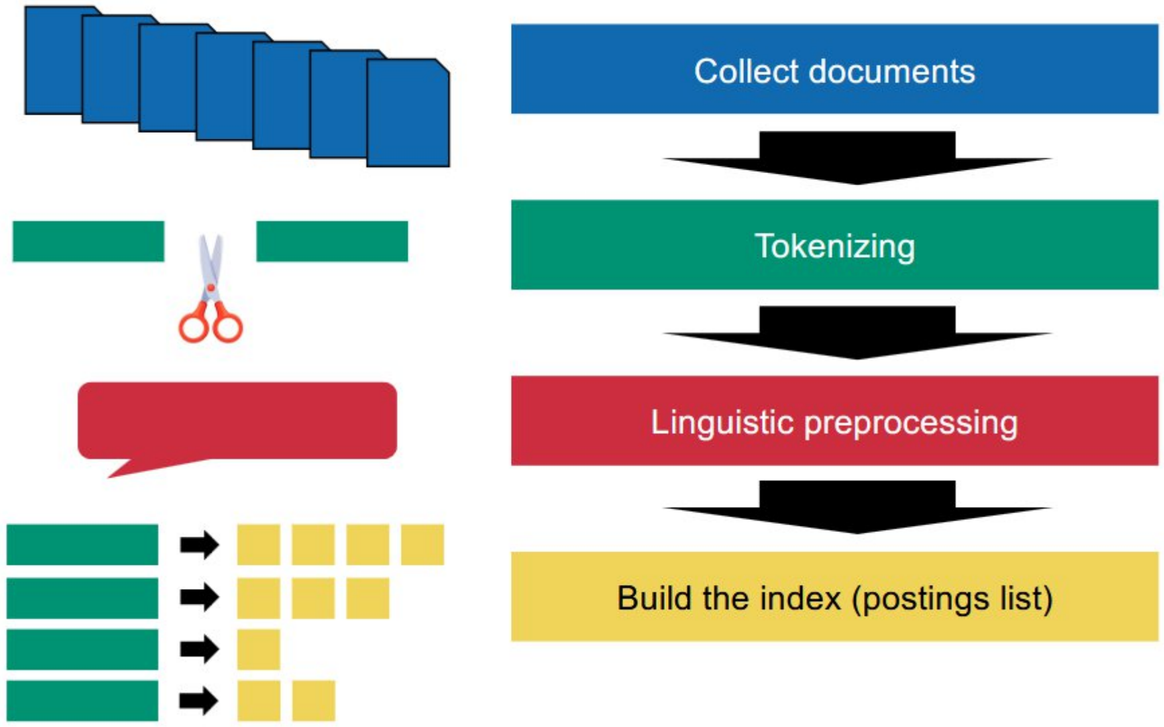
Intersection algorithm

Used to find documents containing
both terms A and B.



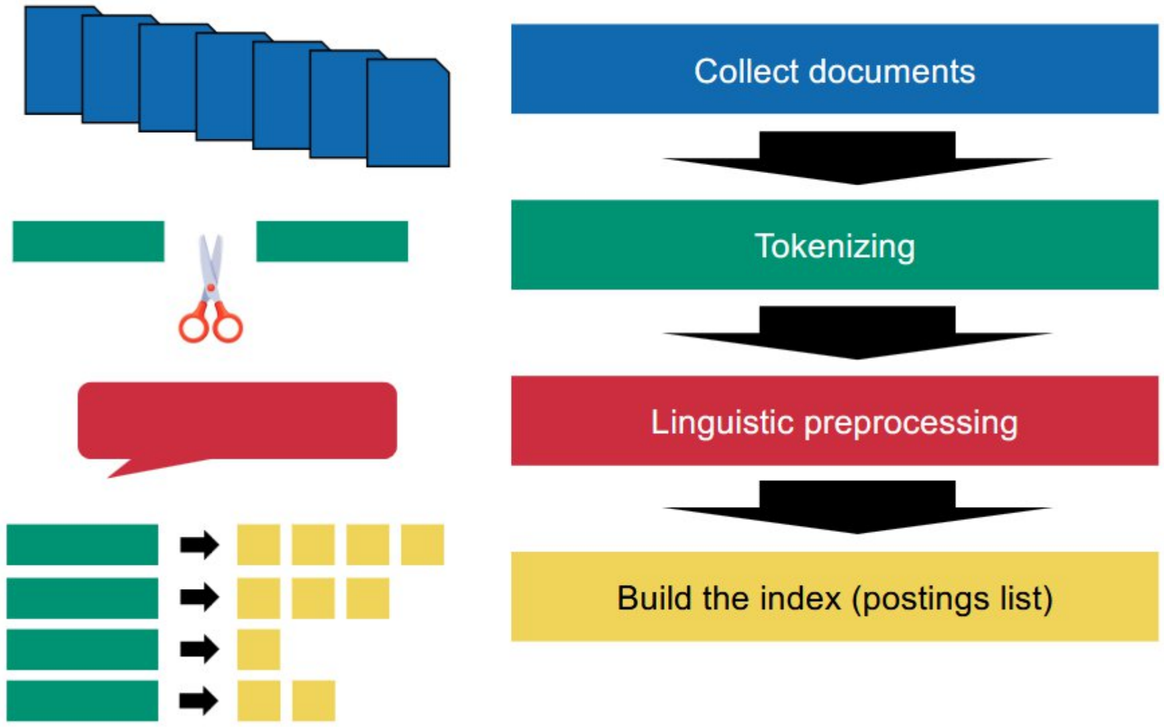
Term Vocabulary

Lot of steps to do before building the index!



Collecting documents

- What encoding type?
- What language?
- In what context?



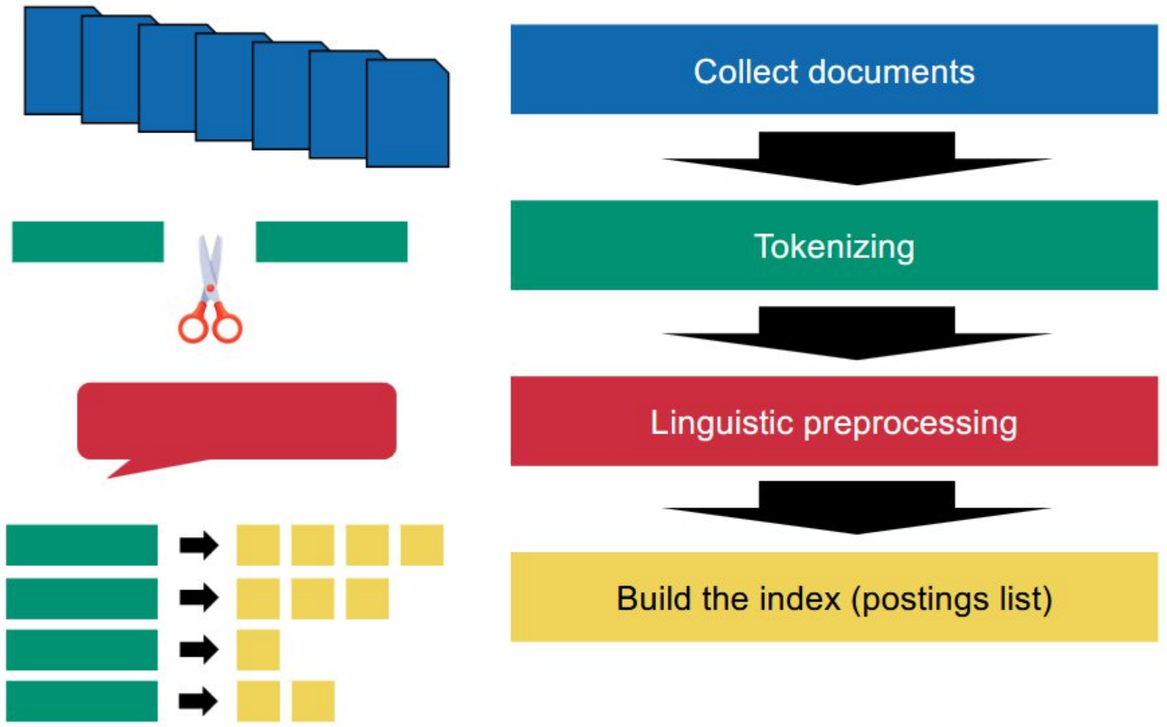
Tokenization

- Punctuation
- Stop words
- Careful of corner cases
- Know the vocabulary!

Raw or processed	Tied to document	Full name	Simplified/casual
raw	tied with position	positional token	token (implicitly positional)
raw	tied without position	non-positional token	
raw	not tied	word, non-normalized type	type (implicitly non-normalized in the book) token (compiler community)
processed	tied with position	positional posting	
processed	tied without position	non-positional posting	posting (implicitly non-positional)
processed	not tied	normalized type, term (if in index)	

Linguistic preprocessing

- Normalization
- Expansion
- Lemmatization and Stemming

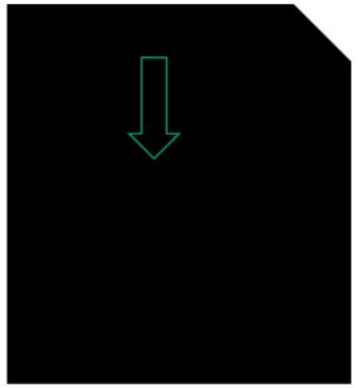


Phrase search

Biword: False positives

Positional: Can reconstruct whole document

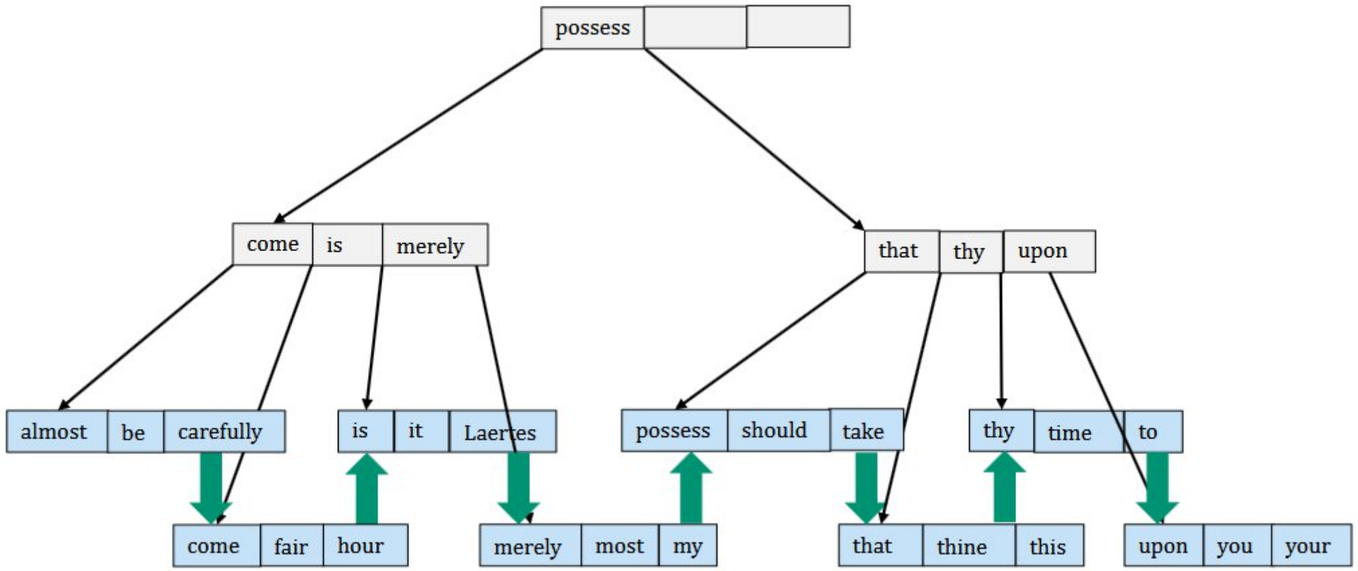
Biword indices



Positional indices

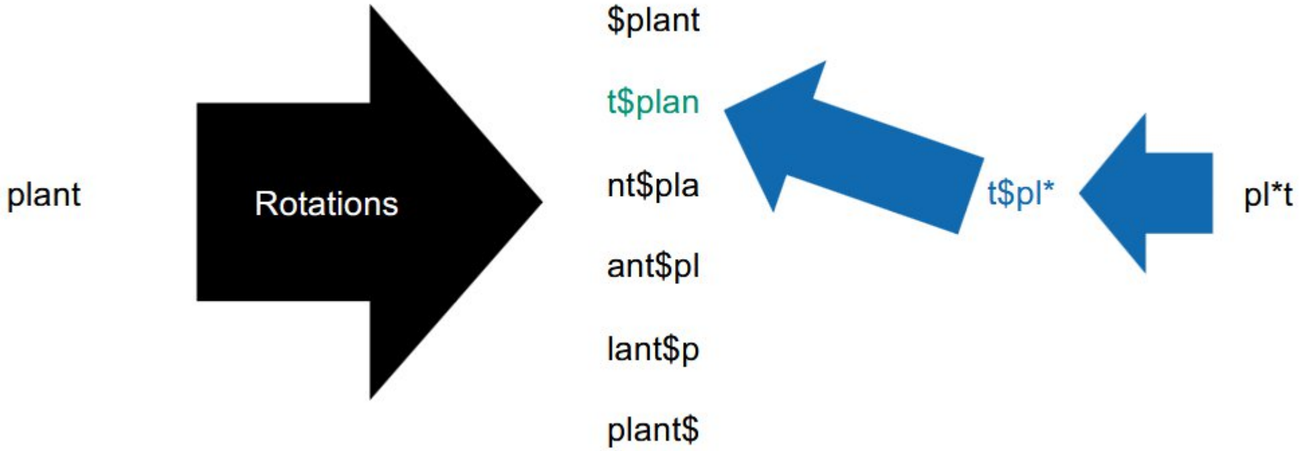
B+-tree

A n-m B+-tree has between **n** and **m** children and between **n – 1** and **m – 1** keys.
All leaves at same depth.
Usually have extra pointers in postings lists.

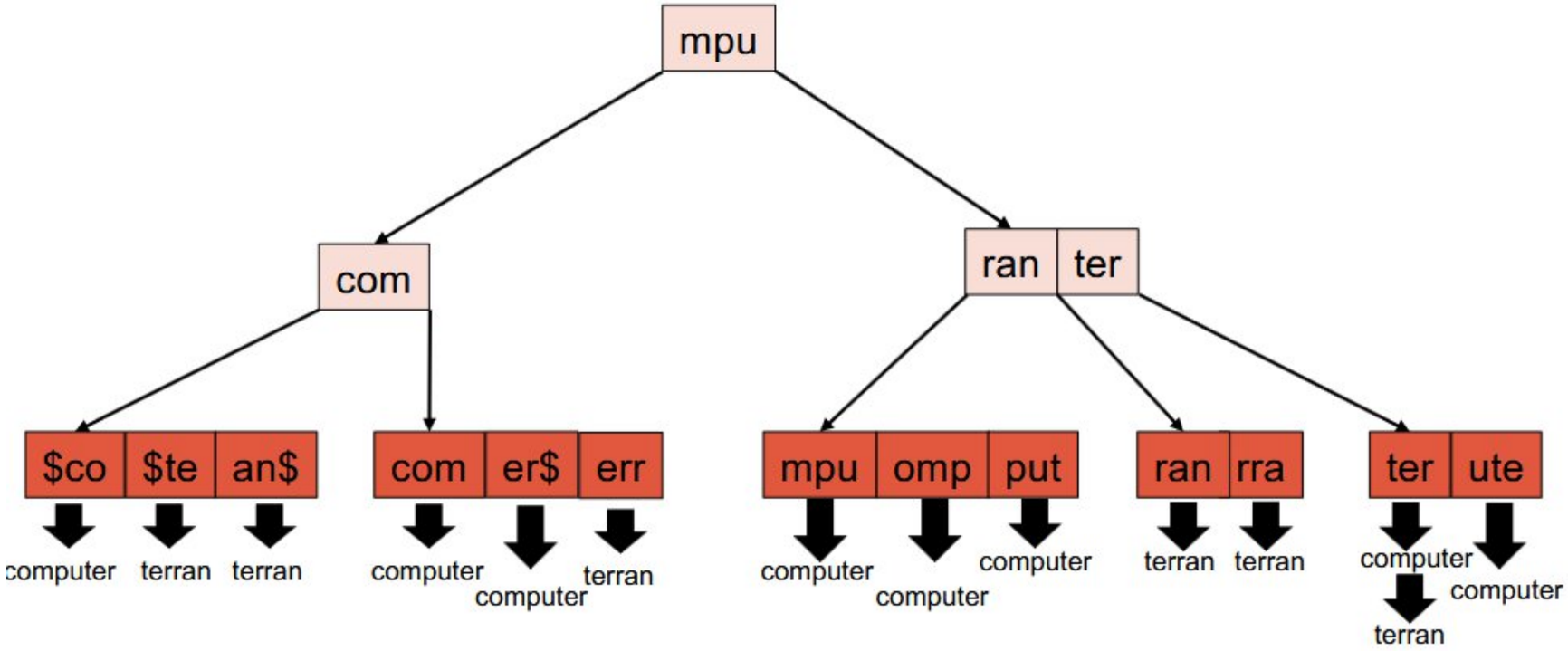


Permuterm index

Use a B+-tree to store all rotations.



k-gram index


mpu
com
ran
ter
$co
$te
an$
com
er$
err
mpu
omp
put
ran
rra
ter
ute
computer
terran
terran
computer
computer
terran
computer
computer
computer
terran
terran
computer
computer
terran

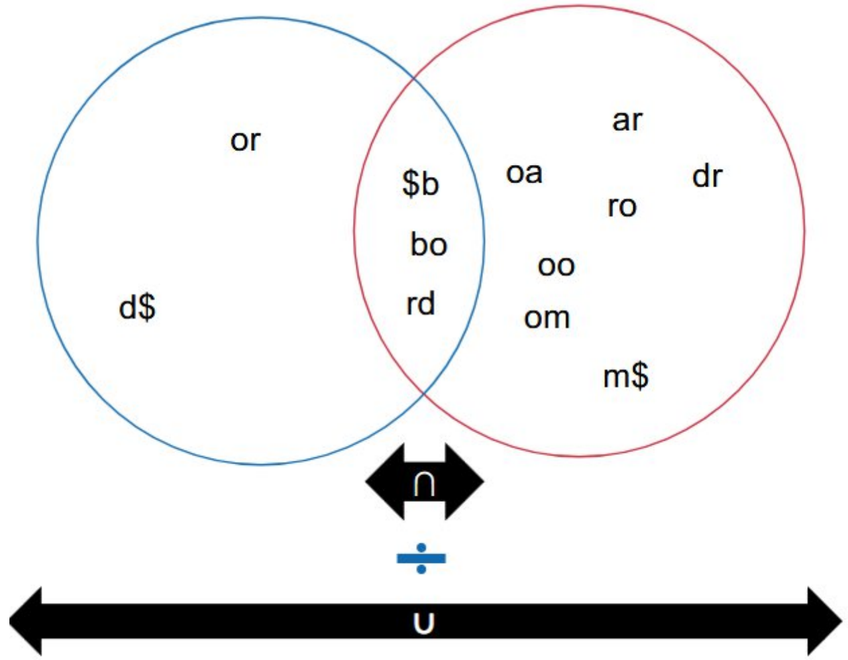
Spell correction

	#	a	t	e
#	0	1	2	3
c	1	1	2	3
a	2	min(3, 1, 2)		
t	3			

0 (do nothing)

+1 (remove a)

+1 (add a)



Blocked Sort-Based Indexing

1. Shard the collection of documents
2. Process each block one by one in memory

- Parse termID-docID pairs
- Sort pairs according to termID
- Write back intermediate results

Vector Space Model

Q2

1. distance
2. leader

Calculating the cosine distance from a query **Q** to all documents **D** is an expensive operation. Cluster pruning attempts to reduce this cost by selecting a subset of $\sqrt{N}$ leaders at random, and partitioning all documents into clusters of approximately $\sqrt{N}$ documents each. To process a query, we only compute the [ ] from the query vector to the [ ] of each [ ], and then search for the [ ] within that cluster. This is a heuristic for solving the nearest-neighbour problem. As a [ ], however, it is [ ] to give the correct answer.

distance | leader | cluster | nearest document | not guaranteed | heuristic | guaranteed | optimization

Champion Lists

Q2

1. distance
2. leader
3. cluster

Calculating the cosine distance from a query **Q** to all documents **D** is an expensive operation. Cluster pruning attempts to reduce this cost by selecting a subset of $\sqrt{N}$ leaders at random, and partitioning all documents into clusters of approximately $\sqrt{N}$ documents each. To process a query, we only compute the [ ] from the query vector to the [ ] of each [ ], and then search for the [ ] within that cluster. This is a heuristic for solving the nearest-neighbour problem. As a [ ], however, it is [ ] to give the correct answer.

distance | leader | cluster | nearest document | not guaranteed | heuristic | guaranteed | optimization

Evaluation

Q2

1. distance
2. leader
3. cluster
4. nearest document

Calculating the cosine distance from a query **Q** to all documents **D** is an expensive operation. Cluster pruning attempts to reduce this cost by selecting a subset of $\sqrt{N}$ leaders at random, and partitioning all documents into clusters of approximately $\sqrt{N}$ documents each. To process a query, we only compute the [ ] from the query vector to the [ ] of each [ ], and then search for the [ ] within that cluster. This is a heuristic for solving the nearest-neighbour problem. As a [ ], however, it is [ ] to give the correct answer.

distance | leader | cluster | nearest document | not guaranteed | heuristic | guaranteed | optimization

Q2

1. distance
2. leader
3. cluster
4. nearest document
5. heuristic

Calculating the cosine distance from a query **Q** to all documents **D** is an expensive operation. Cluster pruning attempts to reduce this cost by selecting a subset of $\sqrt{N}$ leaders at random, and partitioning all documents into clusters of approximately $\sqrt{N}$ documents each. To process a query, we only compute the [ ] from the query vector to the [ ] of each [ ], and then search for the [ ] within that cluster. This is a heuristic for solving the nearest-neighbour problem. As a [ ], however, it is [ ] to give the correct answer.

distance | leader | cluster | nearest document | not guaranteed | heuristic | guaranteed

optimization

Q2

Calculating the cosine distance from a query **Q** to all documents **D** is an expensive operation. Cluster pruning attempts to reduce this cost by selecting a subset of $\sqrt{N}$ leaders at random, and partitioning all documents into clusters of approximately $\sqrt{N}$ documents each. To process a query, we only compute the [ ] from the query vector to the [ ] of each [ ], and then search for the [ ] within that cluster. This is a heuristic for solving the nearest-neighbour problem. As a [ ], however, it is [ ] to give the correct answer.

distance | leader | cluster | nearest document | not guaranteed | heuristic | guaranteed | optimization

1. distance
2. leader
3. cluster
4. nearest document
5. heuristic
6. not guaranteed

https://create.kahoot.it/details/duplicate-of-information-retrieval-ex-07-vector-space-models-mschoeb/ef383953-b43a-4abd-af2a-d9ebf2ad1019