INFORMATION RETRIEVAL

Week 12 – Summary

Severin Mills

Today

1

Semester Recap

2

Theory

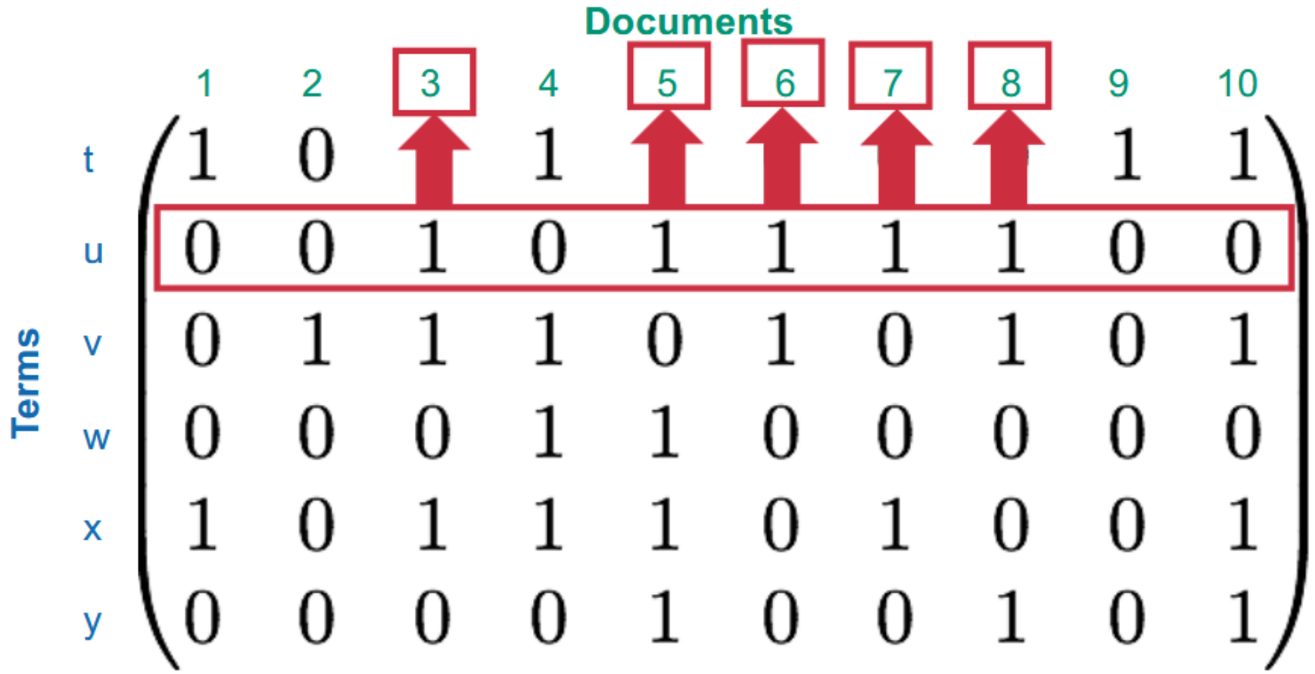
- Large Language Models

3

Kahoot

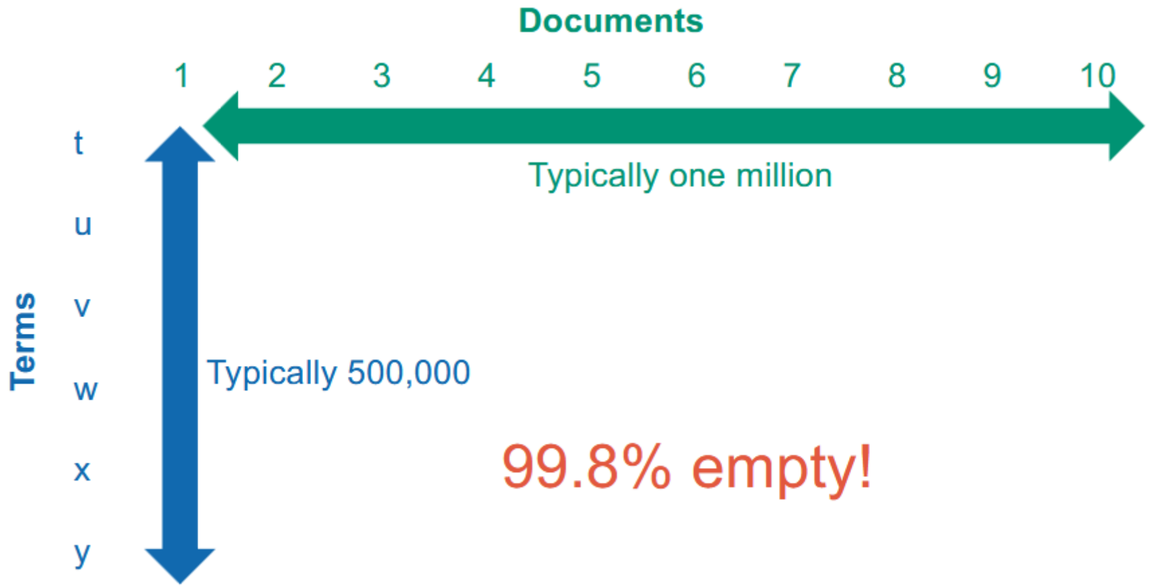
Incidence Matrix

0: Term is **not** in document
1: Term is in document



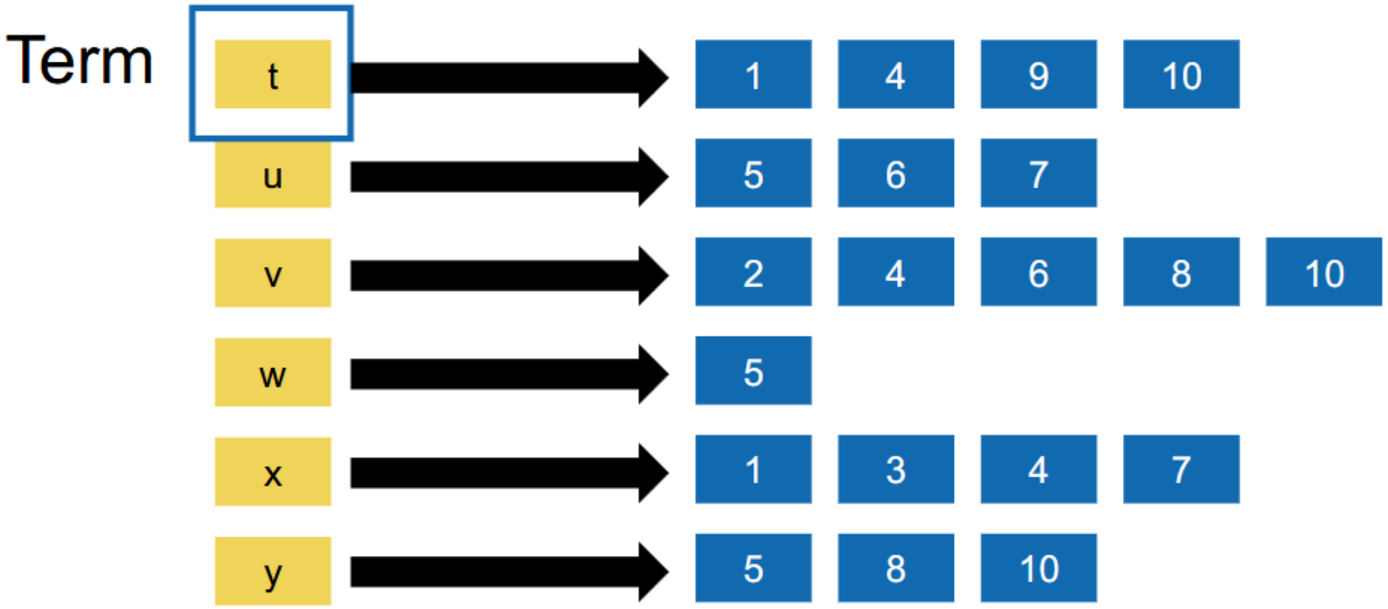
Incidence Matrix

Very inefficient storage usage!



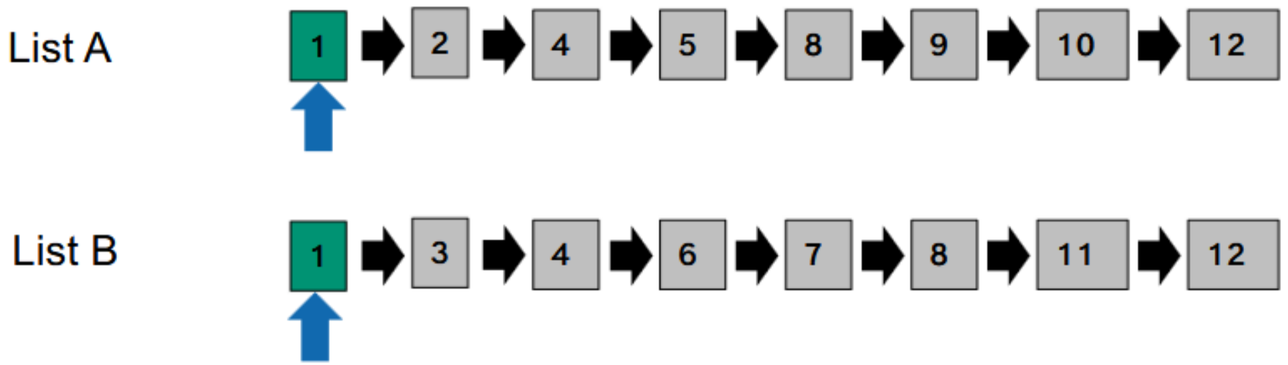
Inverted Index

No storage usage for the zeroes, store documents in lists



Intersection algorithm

Used to find documents containing
both terms A and B.

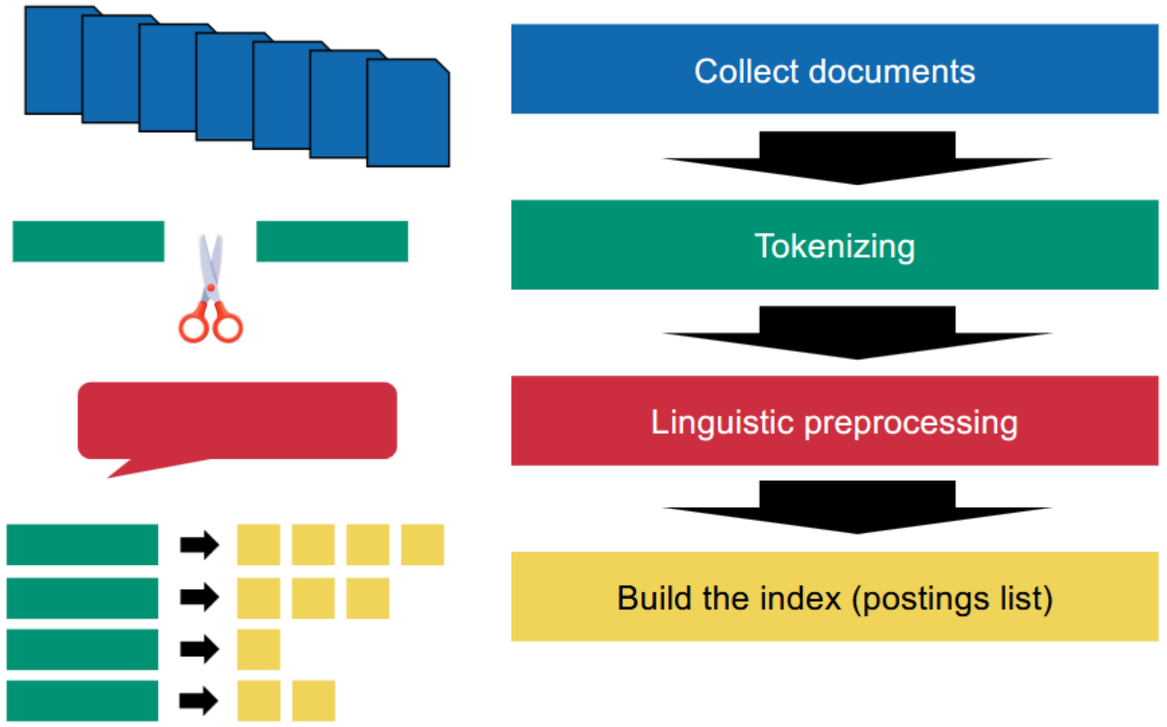


Exam question types

- Querying on Incidence Matrices
- Building the Inverted Index
- Performing queries on the index
- Complexities of query types
- Intersection algorithm (steps, length)

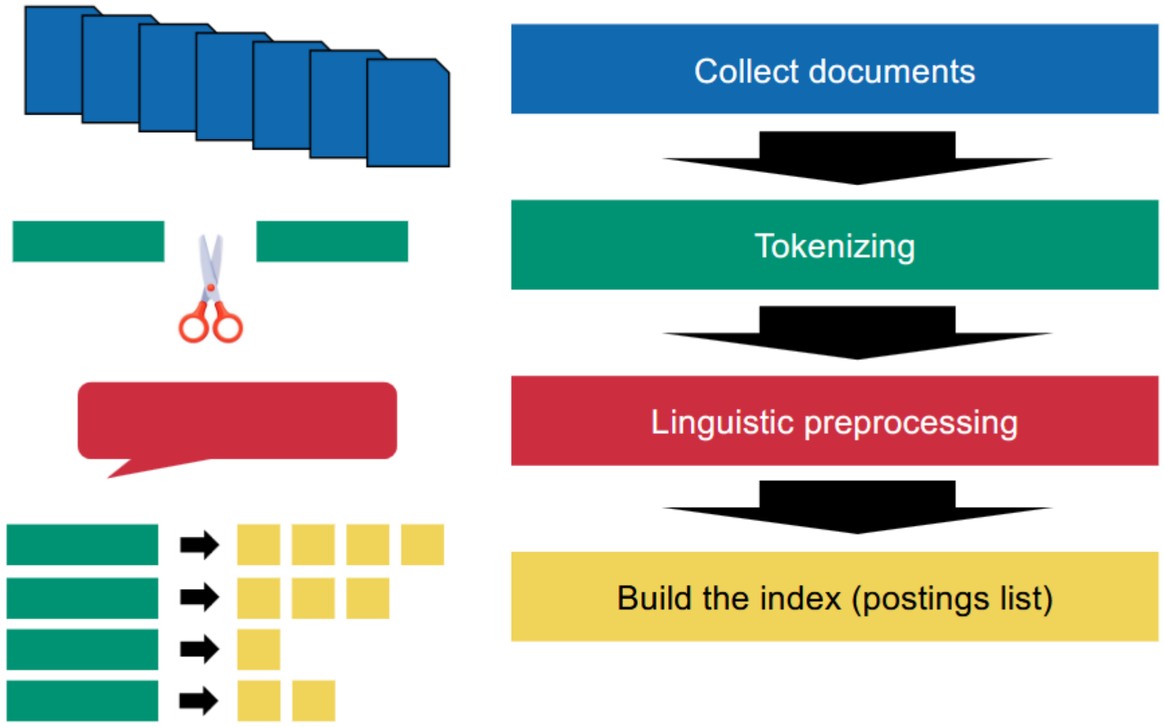
Term Vocabulary

Lot of steps to do before building the index!



Collecting documents

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



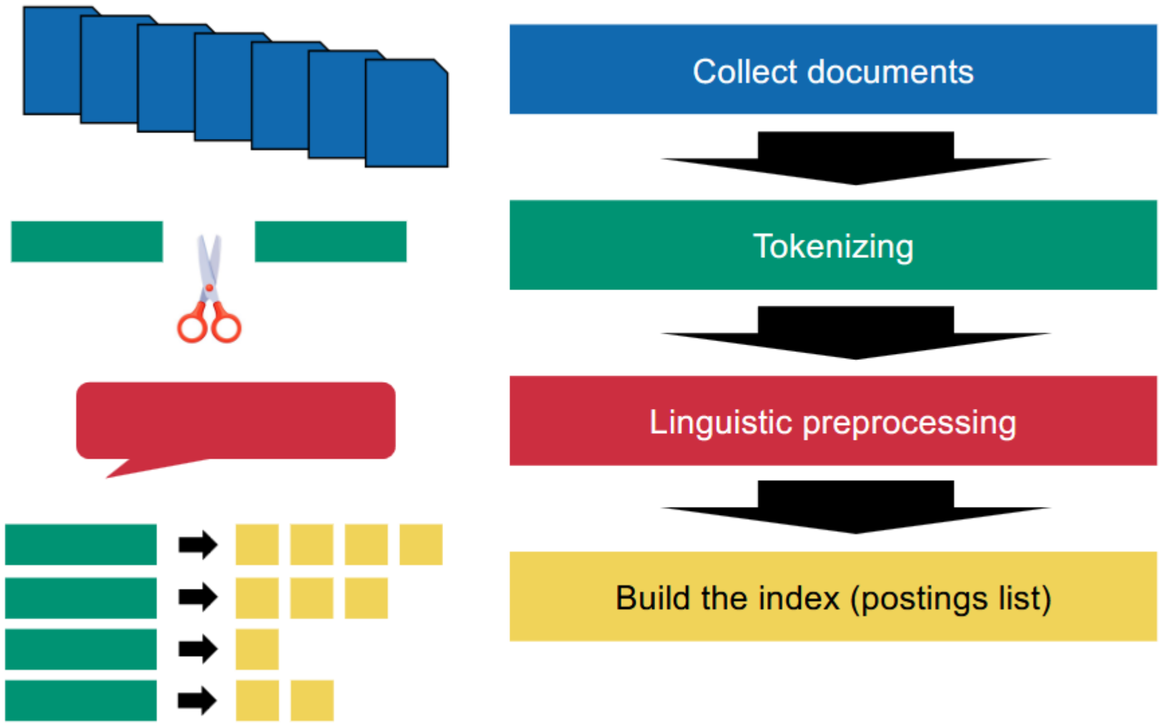
Tokenization

- 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

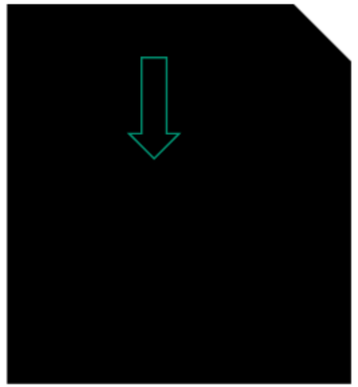
- Normalization
- Expansion
- Lemmatization and Stemming



Phrase search

Biword: False positives
Positional: Can reconstruct whole document

Biword indices



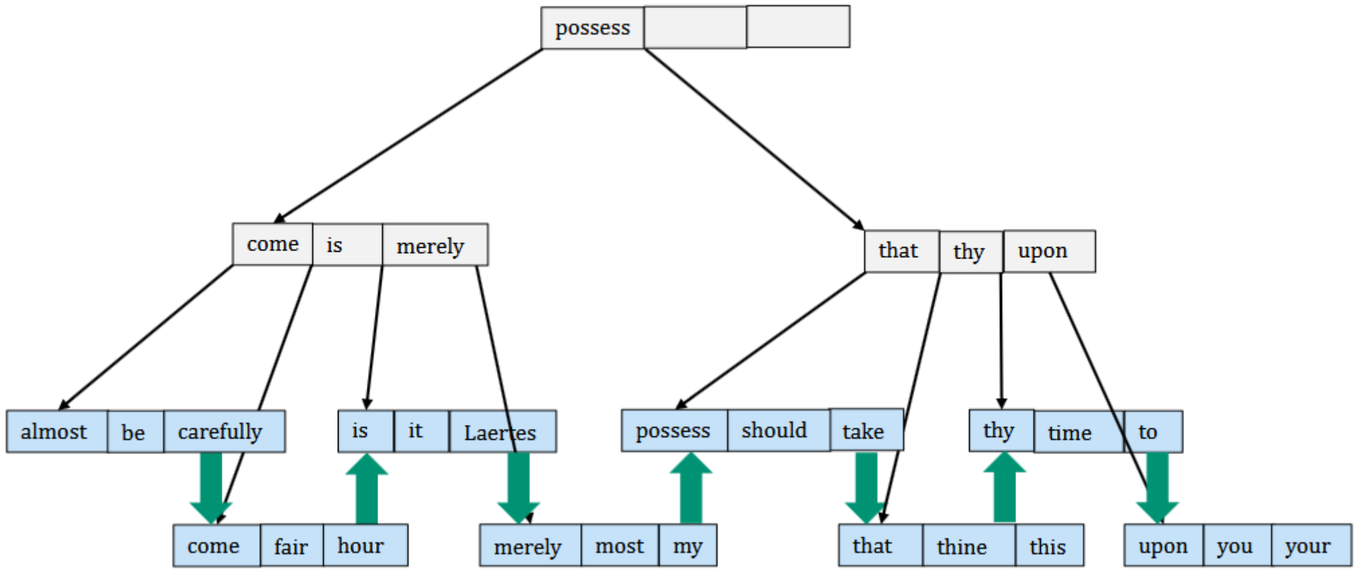
Positional indices

Exam questions

- Applying stemming and lemmatization
- Building a bi-word index
- Querying on bi-word and positional indices
- Reconstruction documents from positional indices

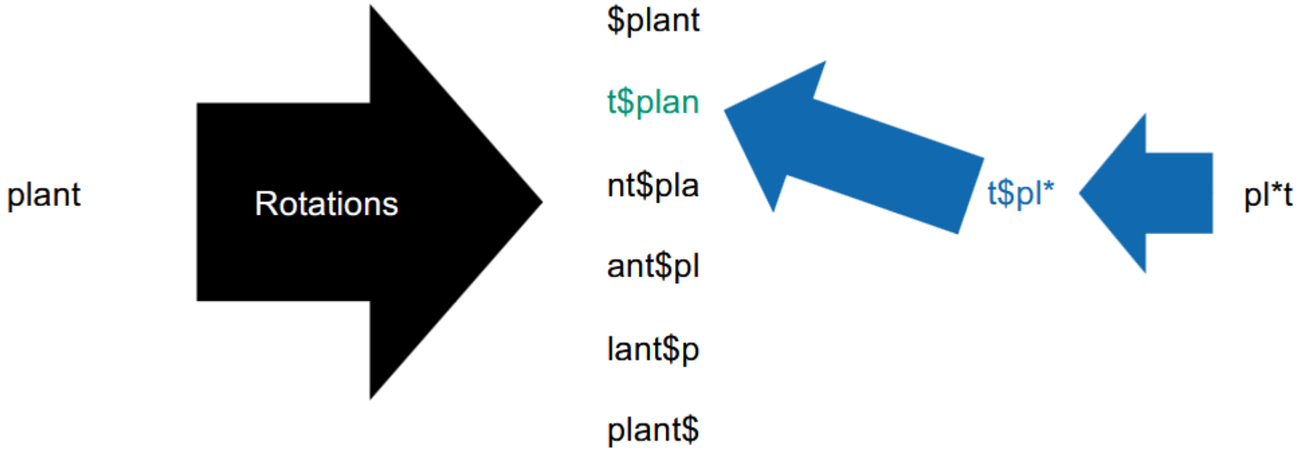
B+-tree

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.

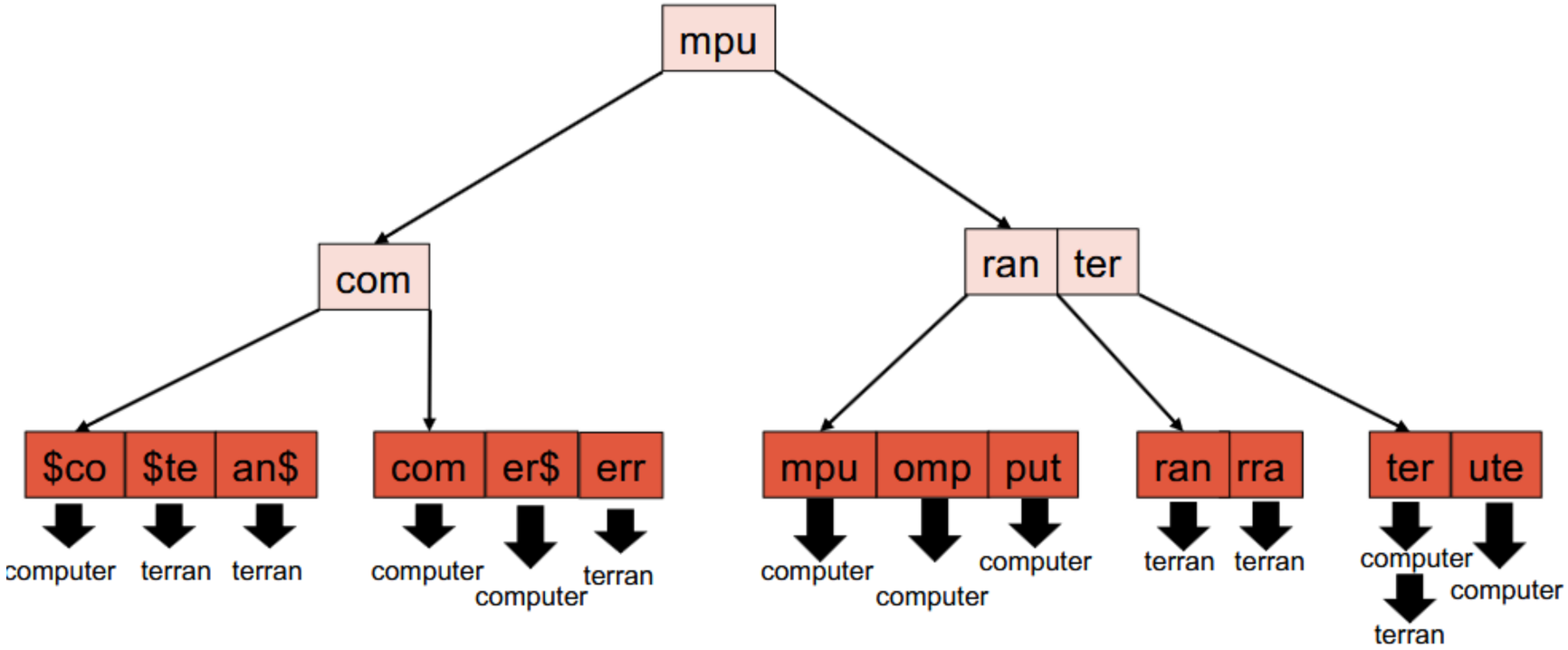


Permuterm index

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

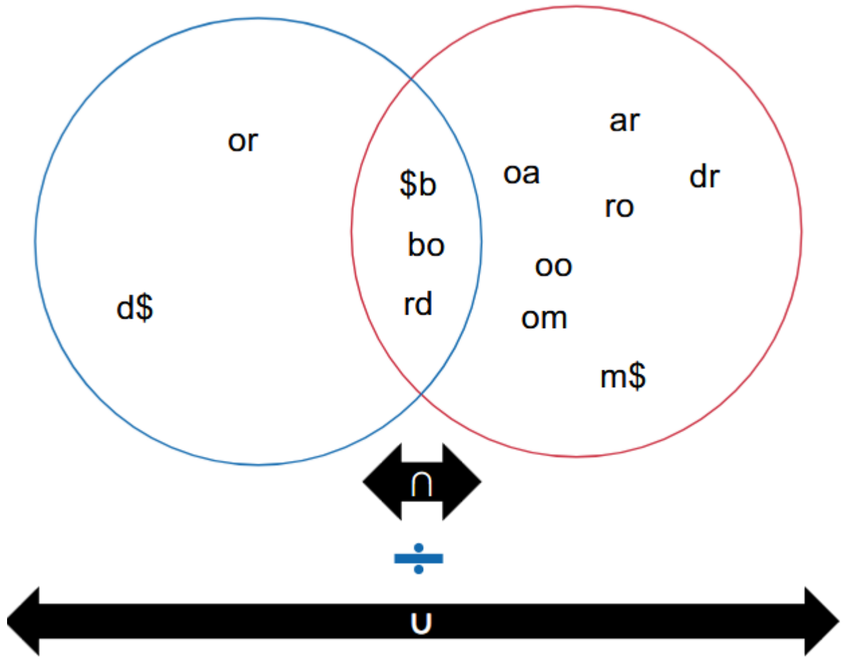
Spell correction

	#	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)



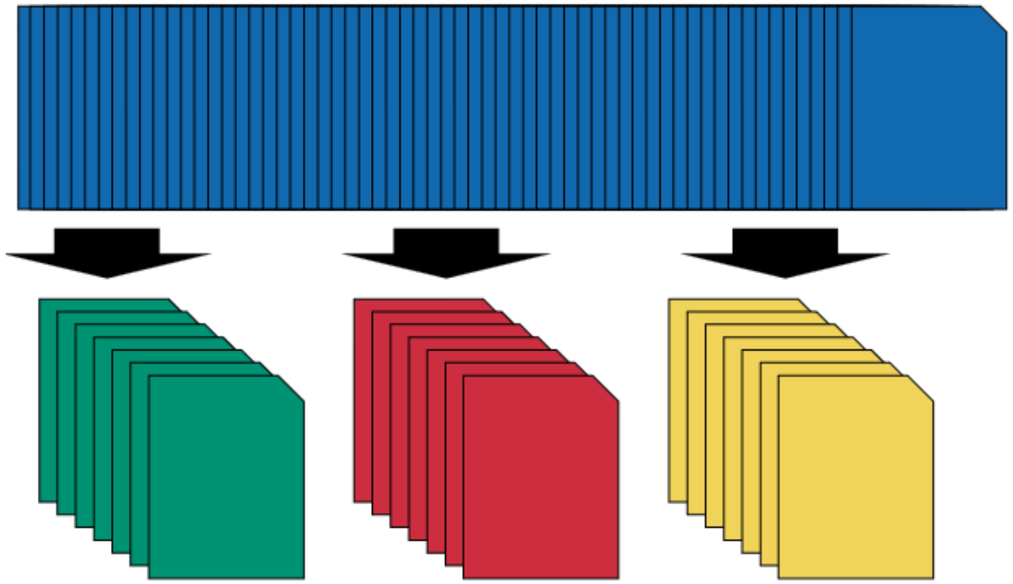
Exam questions

- Calculations using Jaccard coefficient and Edit distance
- Working with k-grams, especially recognizing false positives
- Working with permuterm indices

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

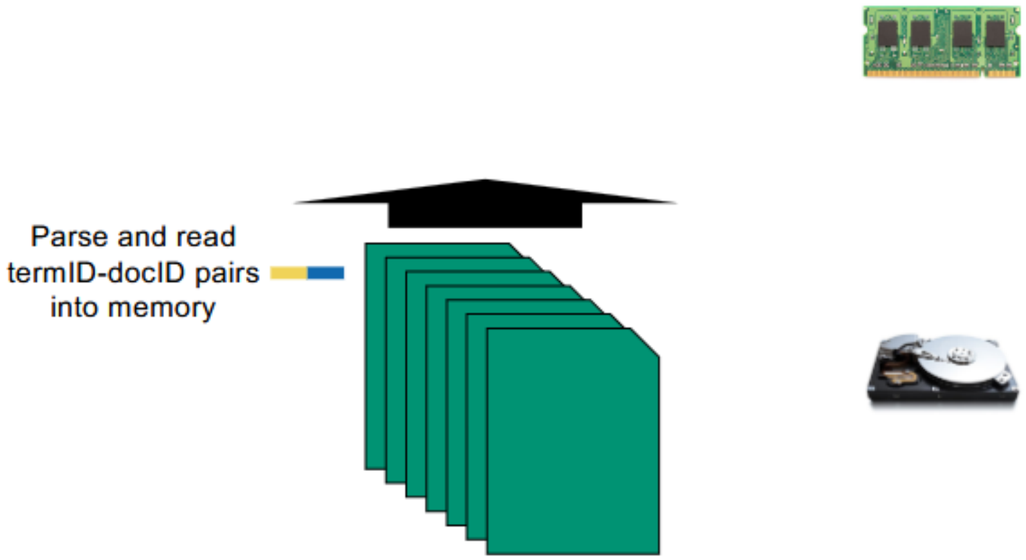
Complexity: O(T log T)



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

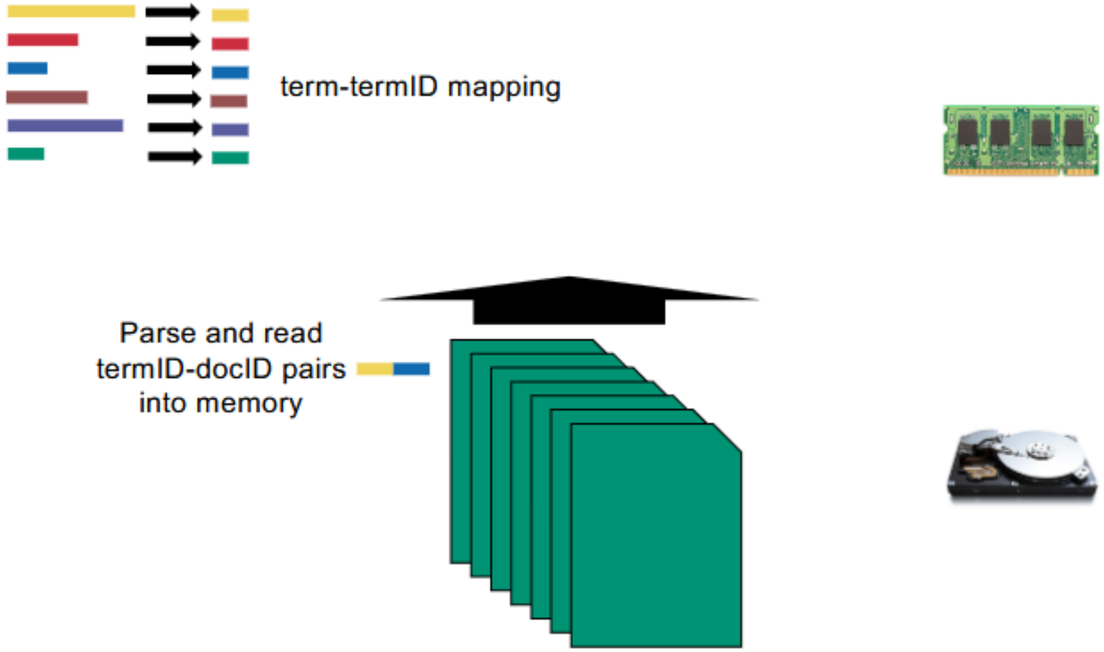
Complexity: O(T log T)



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

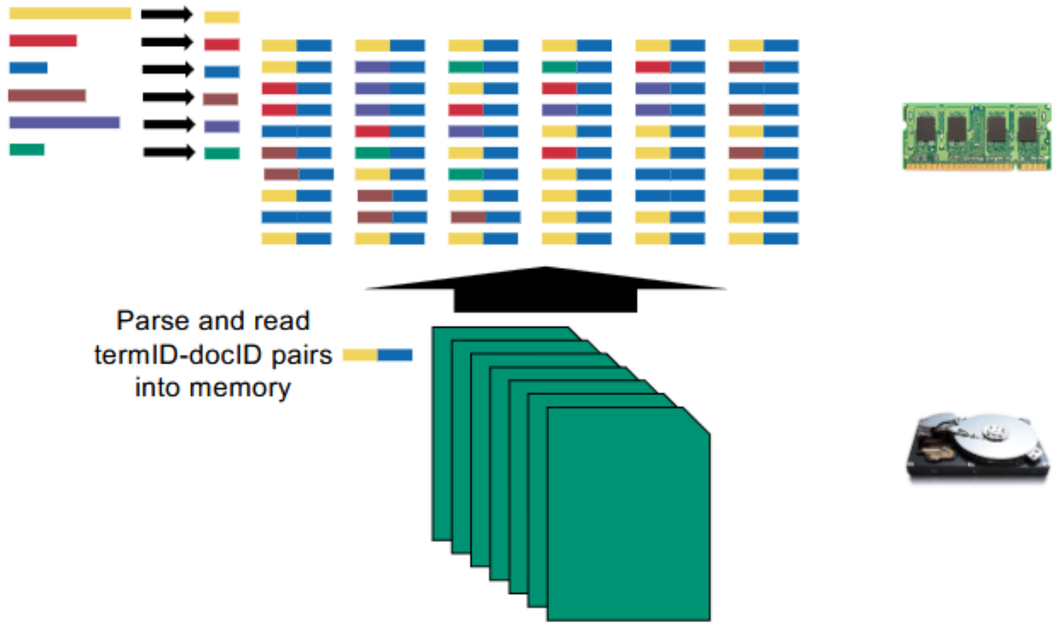
Complexity: O(T log T)



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

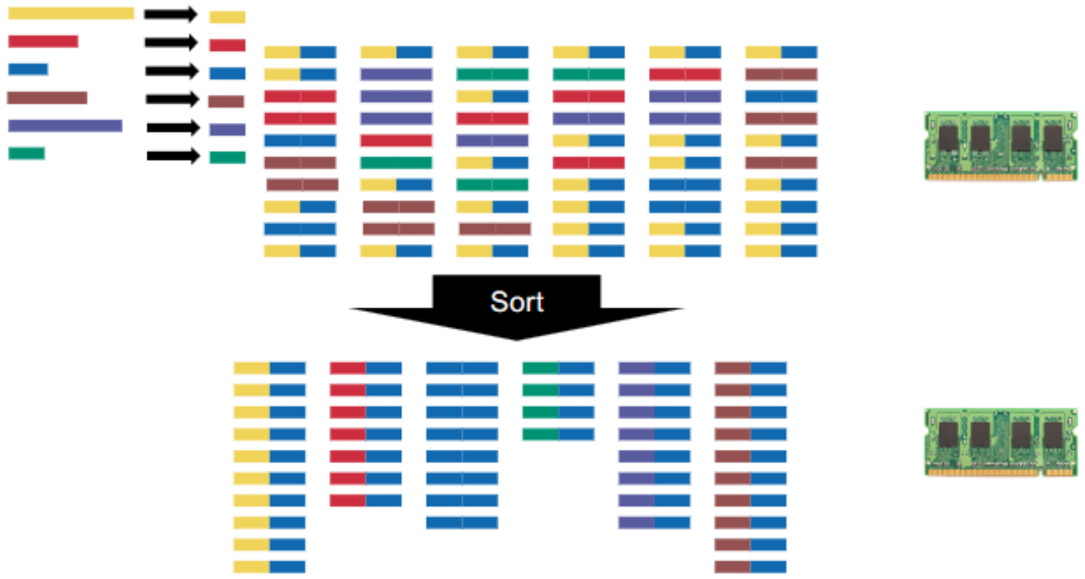
Complexity: O(T log T)



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

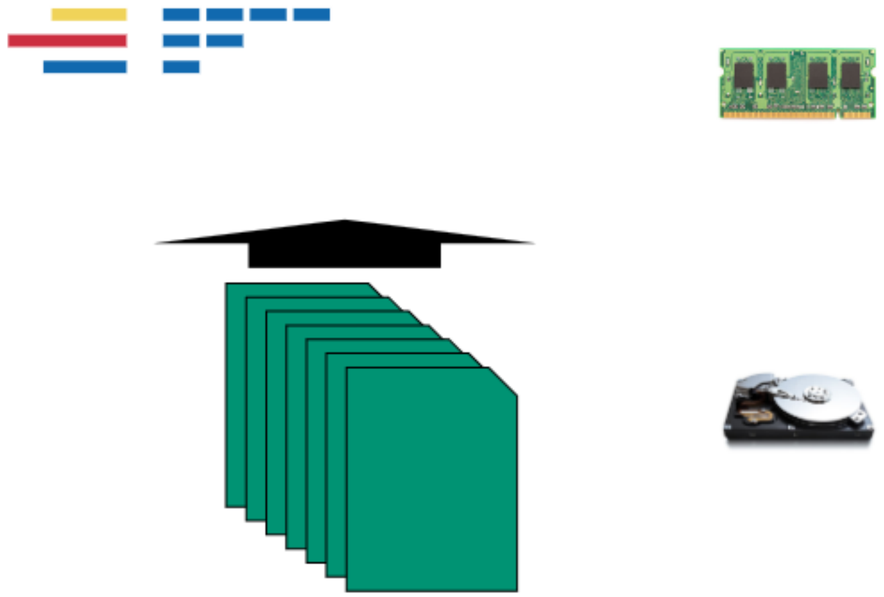
Complexity: O(T log T)



Single-Pass In-Memory Indexing

1. Shard the collection of documents
2. Process each block one by one in memory
- Create intermediary index
- Sort index by term
- Write back intermediate index to disk

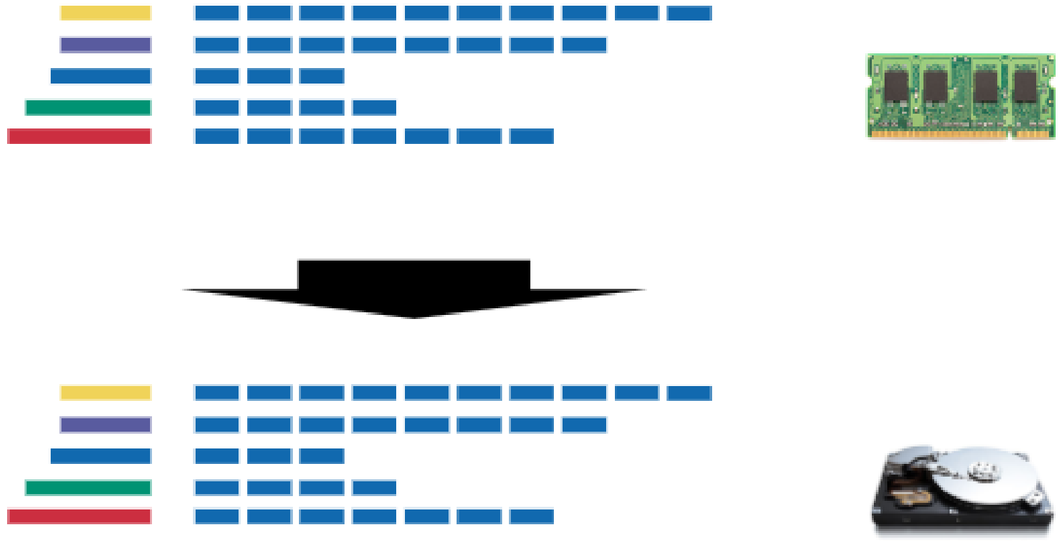
Complexity: O(T log M)



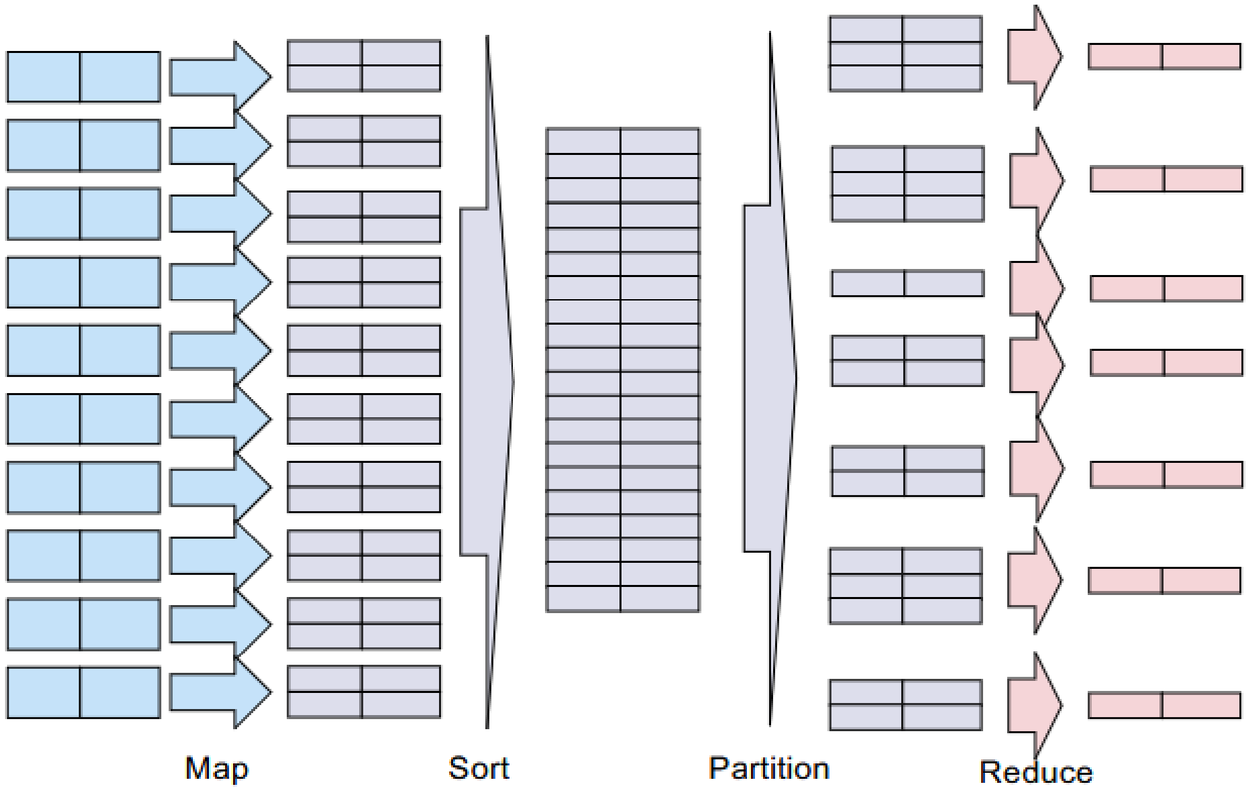
Single-Pass In-Memory Indexing

1. Shard the collection of documents
2. Process each block one by one in memory
- Create intermediary index
- Sort index by term
- Write back intermediate index to disk

Complexity: O(T log M)



MapReduce



Exam questions

- Know the complexities and algorithms of BSBI and SPIMI
- Performing logarithmic merging

Compression

Heap’s Law, Zipf’s law

$$M = k\sqrt{T}$$

Compression methods:

- Encoding gaps
- Unary
- Variable length
- Fixed length
- UTF-8
- Gamma

$$Frequency = \frac{k}{Rank}$$

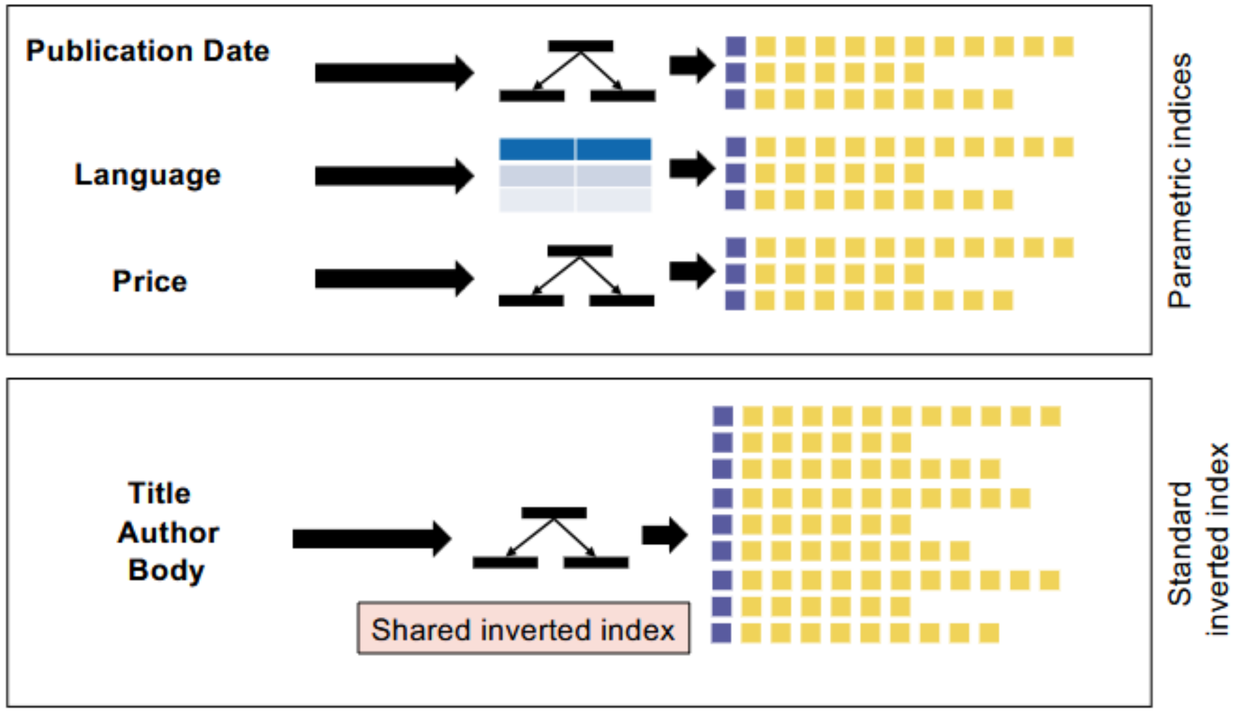
Compression

- Applying Heap’s and Zipf’s law
- Mainly applying the various compression methods and knowing how they work

Zone Search

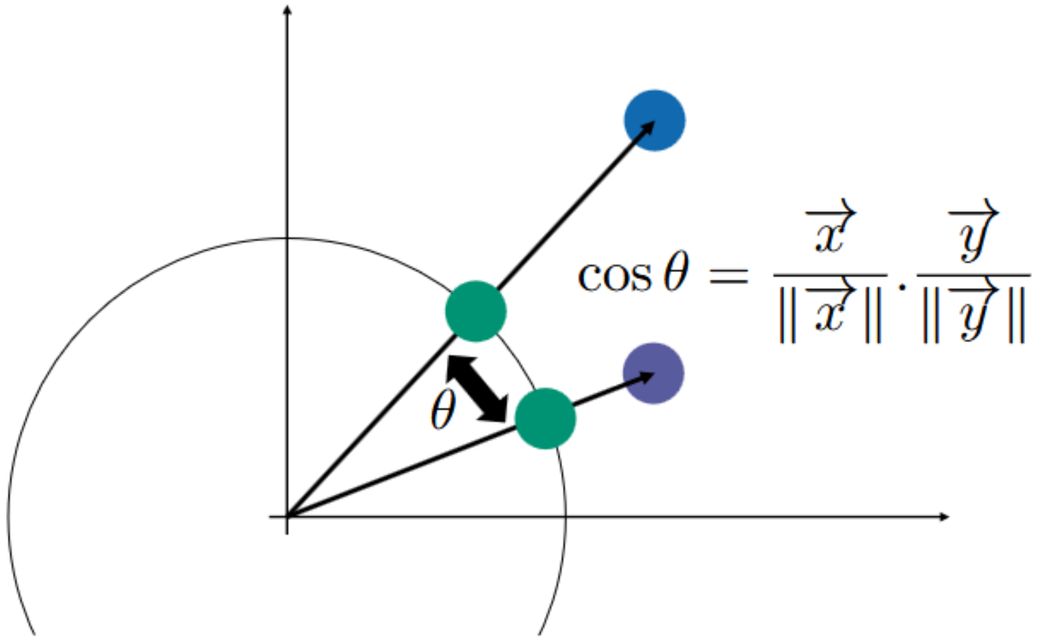
- Parametric indices: Each parameter gets a index
- Shared index: All parametres in one index, but each posting is flagged by parametre

Both: Add weights to zones, rank by weights.



Vector Space Model

Encode documents and queries as vectors. Use tf-idfs as entries to vector
Renormalized inner product gives similarity between vectors in the vector space.



SMART Notation

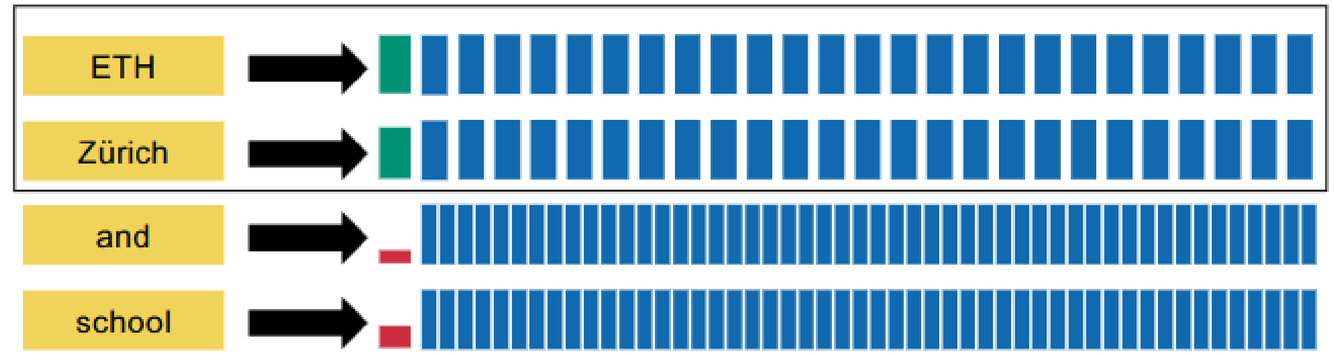
Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1 + \log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5 + \frac{0.5 \times \mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$ (Section 6.4.4)
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/CharLength^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t \in d}(\mathrm{tf}_{t,d}))}$				

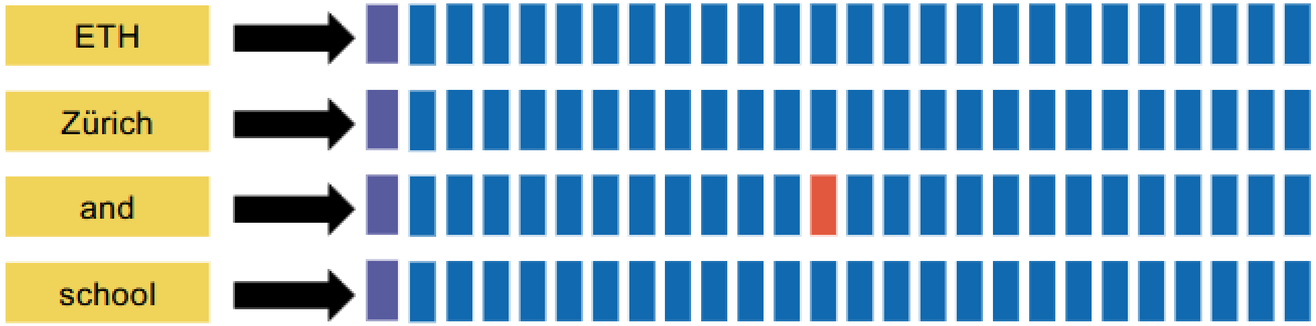
Exam questions

- Know how to calculate tf-idfs
- Know SMART Notation
- Compute scores between documents and queries

Index Elimination

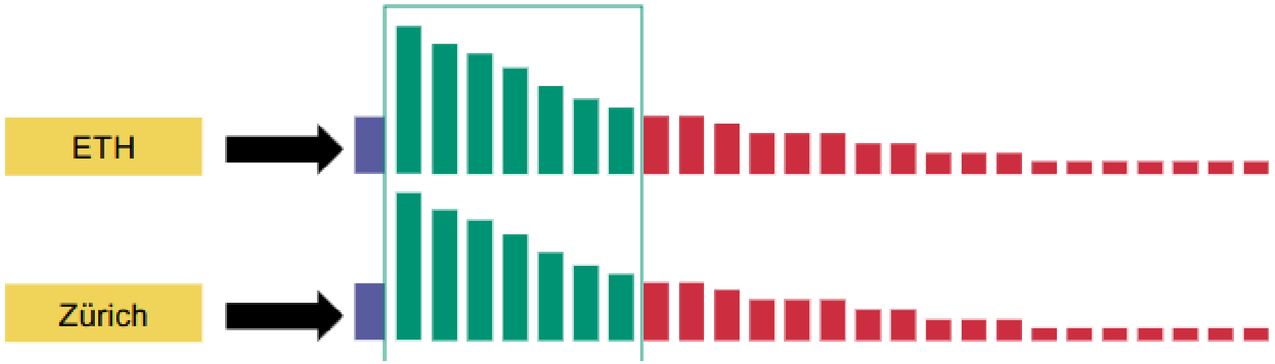
1. Remove postings lists with many postings
2. Remove postings with only a few terms





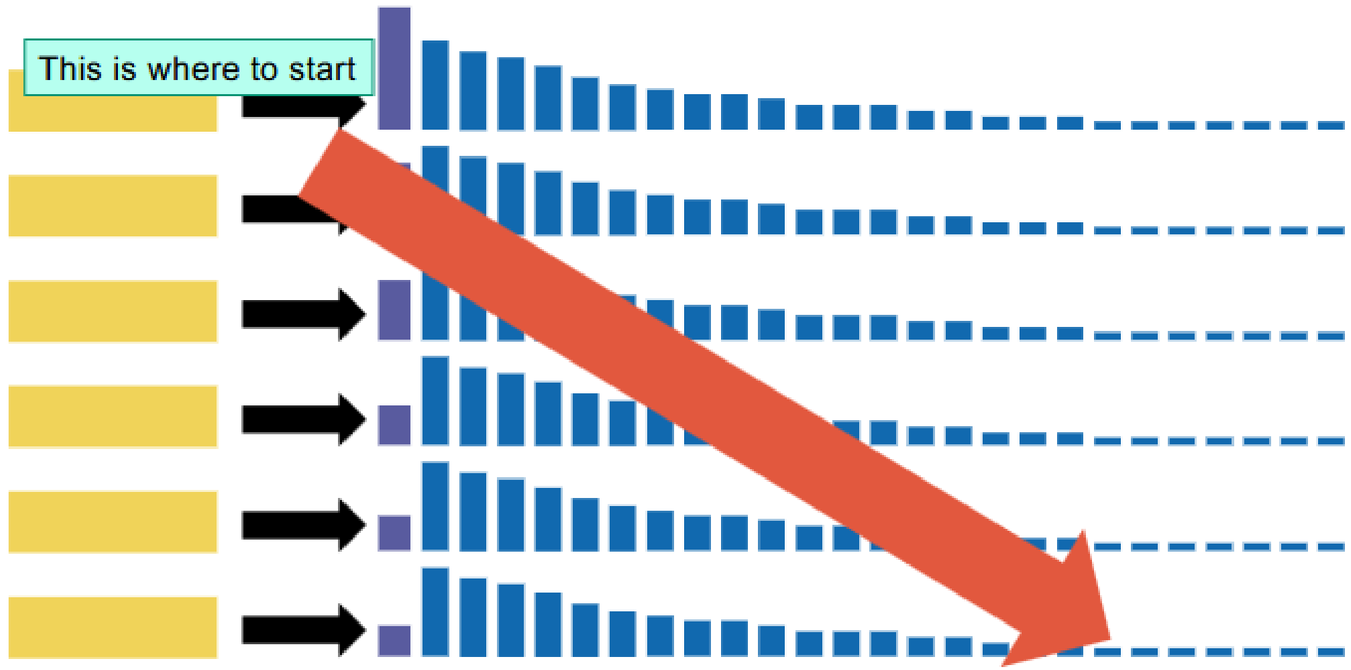
Champion Lists

1. Sort PLs by decreasing tf
2. Keep top r documents from each postings list.
3. Union top r from each term



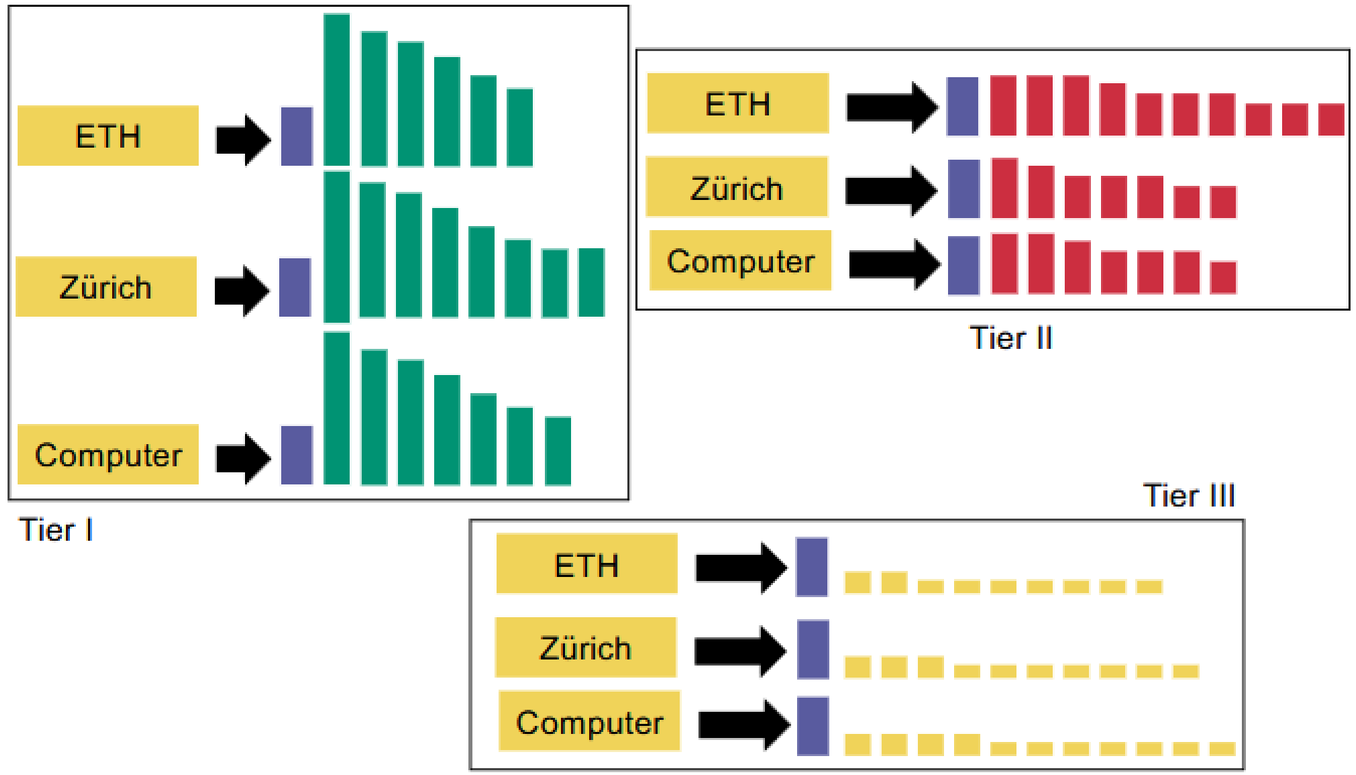
Impact Ordering

1. Build Champion Lists
2. Sort terms by idf
3. Traverse term-at-a-time

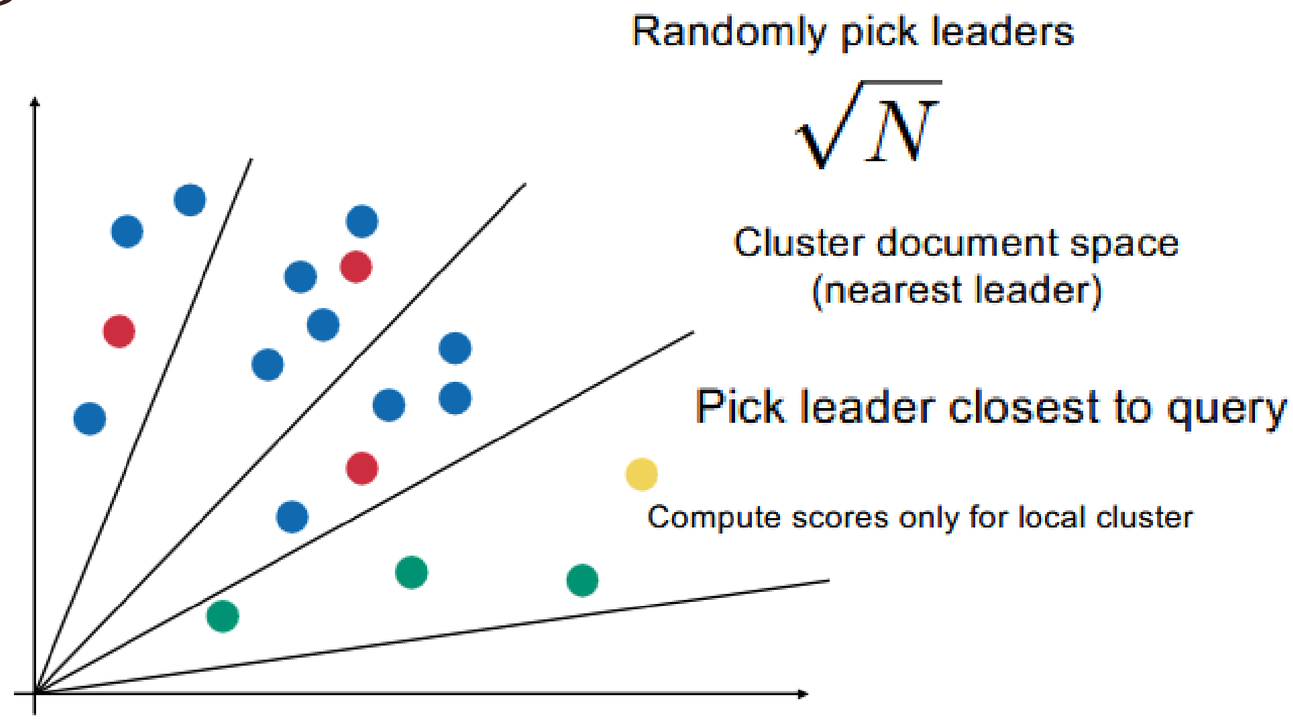


Tiered indices

1. Build Champion Lists
2. Split up into tiers by term frequencies
3. Go through T1, if not enough, go to T2, etc.



Clustering


Randomly pick leaders
$\sqrt{N}$
Cluster document space
(nearest leader)
Pick leader closest to query
Compute scores only for local cluster

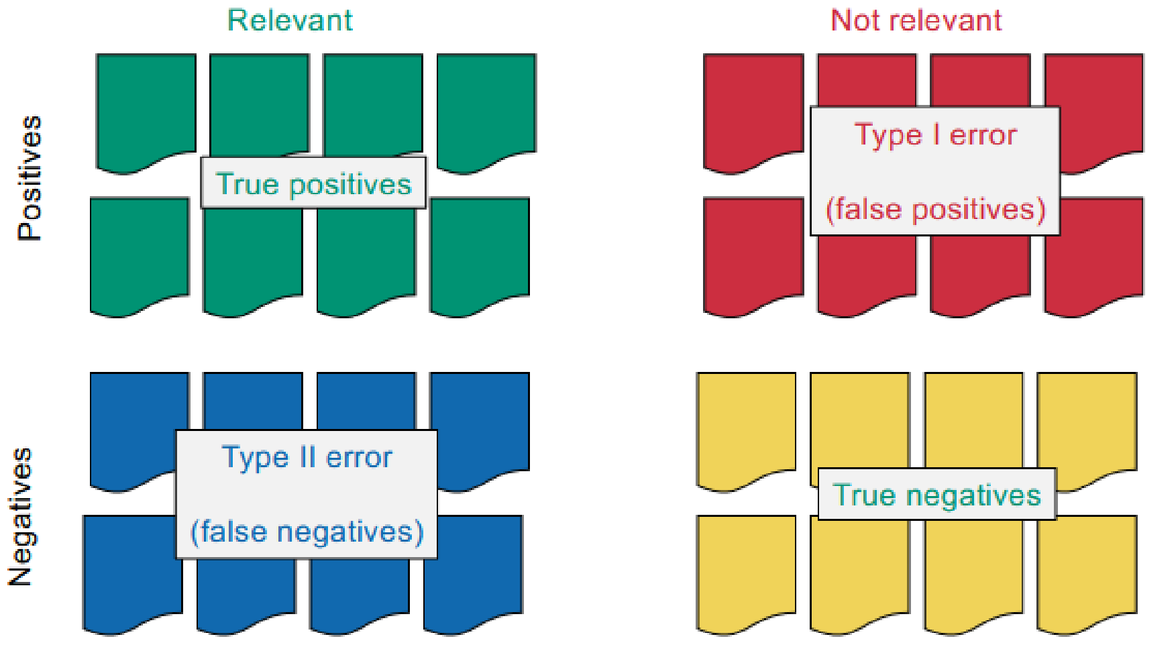
Exam questions

- Know exactly how the different scoring systems work

Metrics

Precision, Recall,
Specificity, Accuracy
F-Measure

$$F_\alpha = \frac{1}{\frac{\alpha}{P} + \frac{1-\alpha}{R}}$$



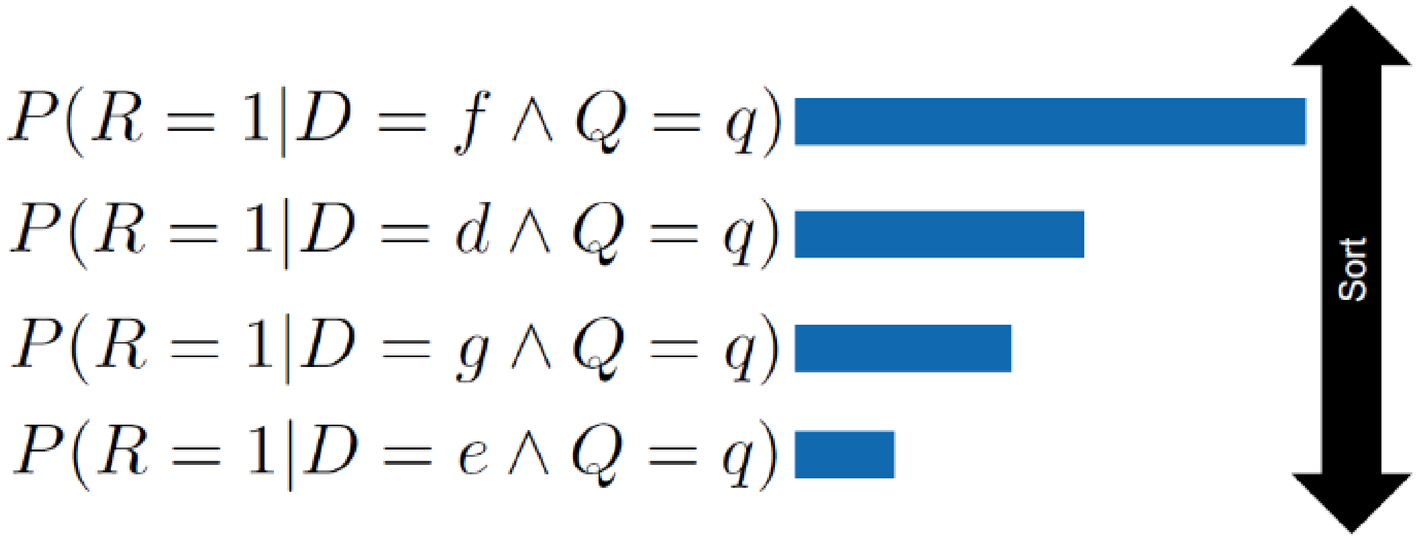
Exam questions

- Know the different metrics by heart
- Differentiate between false positives and false negatives
- Use the F-Measure
- Understand the ROC Curve

$$P(R = 1 | D = d \wedge Q = q)$$

Probability that, for a query q and a document d,
d is relevant for query q

Probability ranking principle



Retrieval Status Value

$$RSV_d = \sum_{k|d_k=1 \wedge q_k=1} \log \frac{N}{\mathrm{df}_t}$$

$$RSV_d = \sum_{k|d_k=1 \wedge q_k=1} \log \frac{p_k}{1-p_k} - \log \frac{u_k}{1-u_k}$$

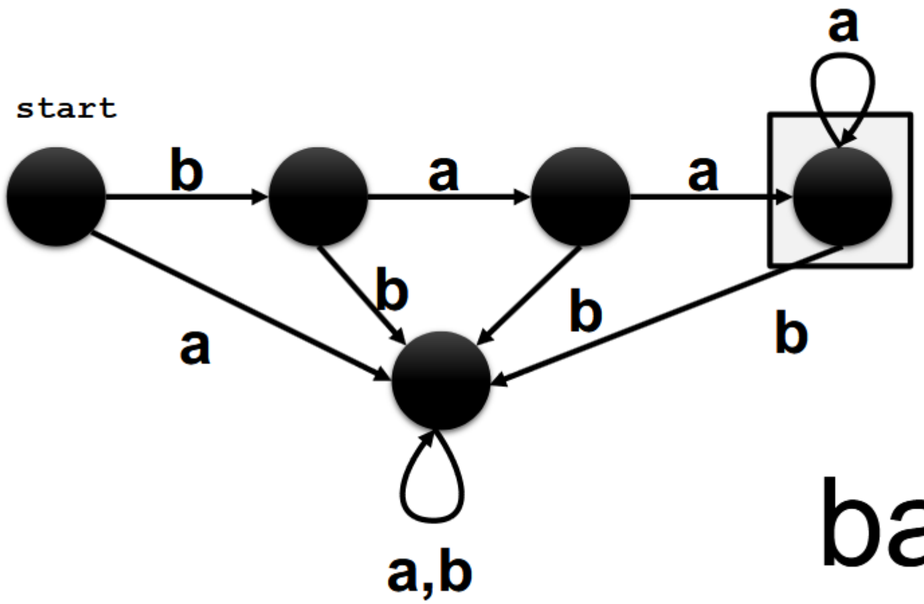
Odds of containing term k in relevant documents

Odds of containing term k in non-relevant documents

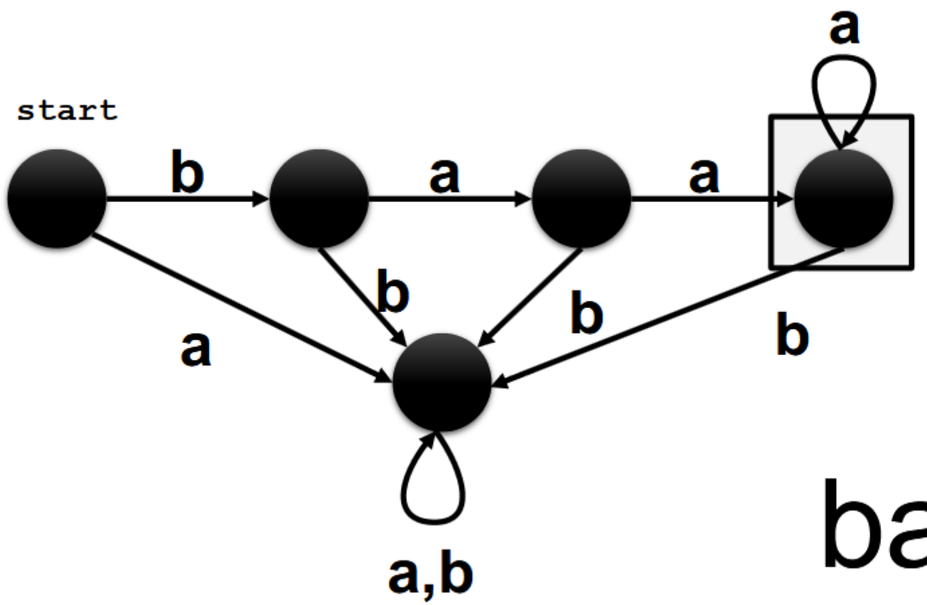
Exam questions

- Know the notations
- Apply Bayes' Rule

Finite State Automata



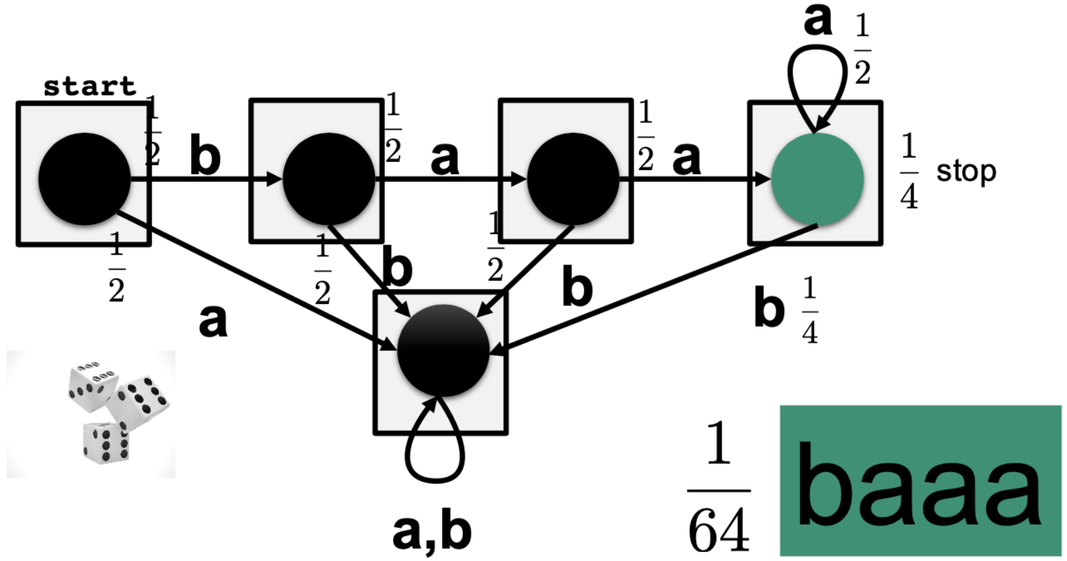
baba



baaaa

How do we generate words?

- Assign probabilities!



Generating a document

“Probability that document D consists of d_1d_2d_3d_4”

$$P(D = (d_1, d_2, d_3, d_4)) =$$
$$(1 - p_{stop})P(D_1 = d_1)$$
$$(1 - p_{stop})P(D_2 = d_2|D_1 = d_1)$$
$$(1 - p_{stop})P(D_3 = d_3|D_1 = d_1 \wedge D_2 = d_2)$$
$$(1 - p_{stop})P(D_4 = d_4|D_2 = d_2 \wedge D_3 = d_3)p_{stop}$$

Exam questions

- Calculate probabilities of generating sentences
- Beware of stop probabilities

Kahoot

https://create.kahoot.it/details/121b6b6e-6890-490e-a0ef-69e84347f04f

30.05.2025