

Multimedia Technology

Lecture 3 Document Retrieval: searching

Lecturer: *Dr. Wan-Lei Zhao*

Autumn Semester 2022

Outline

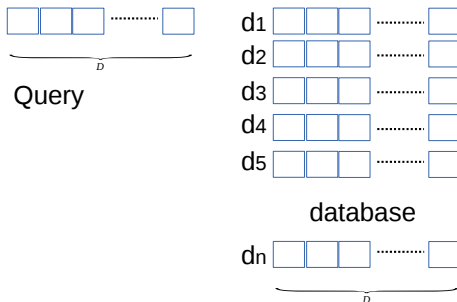
- 1 Inverted files
- 2 Searching
- 3 Word Embedding and Co-occurrence Matrix

Recap

- Documents are undergone a series pre-processing
 - Tokenization, stemming and etc.
 - Only index terms are left
- Boolean model, vector model and probabilistic model
 - Vector model is selected for its simplicity and good performance
- Ready to do the retrieval
- Brute-force comparison between query and all documents in the database is not efficient
 - Dimension of query is high ($> 10,000$), size of database is in billion level
 - You cannot expect users are all as patient as you:)
 - Problem of nearest neighbour search (NNS) in high-dimensional and sparse space

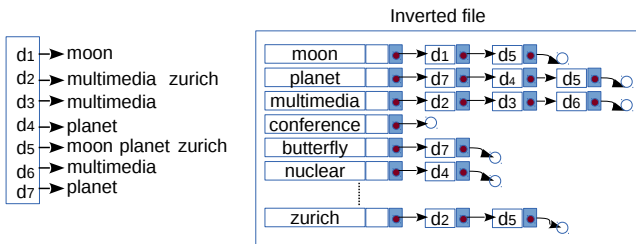
Brute-force search: illustration

- NNS is a very challenging problem, whose complexity is $\mathcal{O}(D * n)$
- Fortunately, the vectors are all sparse!!



Inverted file construction

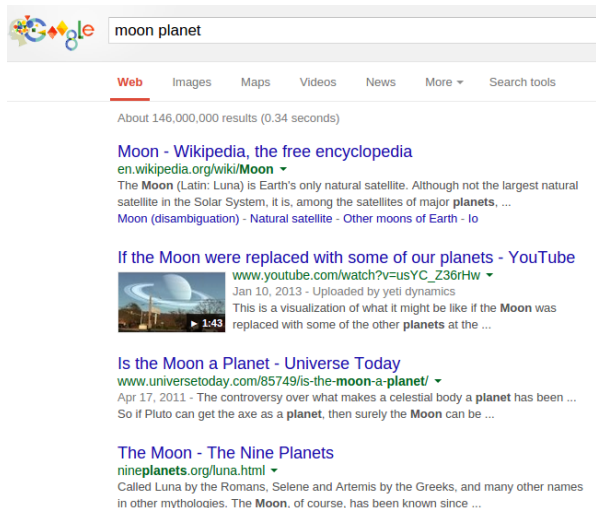
- Idea: term points towards which documents it occurs



- Two representations keep the same amount of information
- Given a query “moon planet”, inverted lists pointed by “moon” and “planet” will be considered
- This way is more efficient

A complete example of inverted file (1)

- Check carefully what we still miss as database of a search engine
- Matched key words are all in **bold**



A screenshot of a Google search results page for the query "moon planet". The search bar at the top shows the query. Below the search bar, the "Web" tab is selected, and the results are displayed. The first result is from Wikipedia, titled "Moon - Wikipedia, the free encyclopedia". The second result is from YouTube, titled "If the Moon were replaced with some of our planets - YouTube". The third result is from Universe Today, titled "Is the Moon a Planet - Universe Today". The fourth result is from Nine Planets, titled "The Moon - The Nine Planets". The search results show the number of results (About 146,000,000 results) and the time taken (0.34 seconds). The search results are sorted by relevance. The search results are displayed in a list format. The search results are displayed in a list format. The search results are displayed in a list format.

moon planet

Web Images Maps Videos News More ▾ Search tools

About 146,000,000 results (0.34 seconds)

Moon - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Moon ▾
 The **Moon** (Latin: Luna) is Earth's only natural satellite. Although not the largest natural satellite in the Solar System, it is, among the satellites of major **planets**, ...
[Moon \(disambiguation\)](#) - [Natural satellite](#) - [Other moons of Earth](#) - [Io](#)

If the Moon were replaced with some of our planets - YouTube
www.youtube.com/watch?v=usYC_Z36rHw ▾
 Jan 10, 2013 - Uploaded by yeti dynamics
 This is a visualization of what it might be like if the **Moon** was replaced with some of the other **planets** at the ...

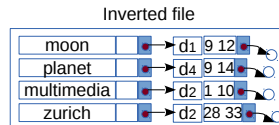
Is the Moon a Planet - Universe Today
www.universetoday.com/85749/is-the-moon-a-planet/ ▾
 Apr 17, 2011 - The controversy over what makes a celestial body a **planet** has been ...
 So if Pluto can get the axe as a **planet**, then surely the **Moon** can be ...

The Moon - The Nine Planets
nineplanets.org/luna.html ▾
 Called Luna by the Romans, Selene and Artemis by the Greeks, and many other names in other mythologies. The **Moon**, of course, has been known since ...

A complete example of inverted file (2)

- How we can achieve that?
- "...keyword..."
- We should know where to insert ""
- This is achieved by keeping the position of one word in the **original** documents
- Notice that document can be viewed as a linear array of words
- The start and rear positions of one word are kept

d1 This is moon.
 d2 Multimedia conf. is held in Zurich.
 d4 It is a planet.



A complete example of inverted file (3)

- Sometimes, the original document is segmented into blocks
- The block ID instead of the word position is kept
- Linear search (KMP) is required to localize the word in the block

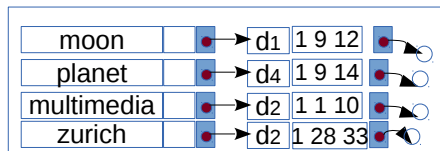


A complete example of inverted file (4)

- For one cell in the inverted list, we should keep
 - position of one word in the original document, two integers
 - document ID (integer)
 - term frequency (TF), (integer/float)
 - URLs for each document
 - Click-in frequency
 - Pagerank



Inverted file



Issues involved in retrieval over inverted files (1)

- The database is now ready
- Given query “moon planet”,
- Shall we compare “moon” with each inverted list leading term?
 - This is a linear scan (time consuming)
 - Any more convenient way??

query

moon	planet
------	--------

for each term in inverted files

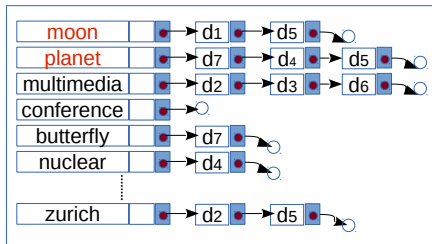
 M = strcmp(“moon”, term)

 if (M)

 break;

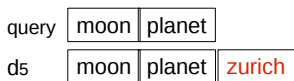
end

Inverted file

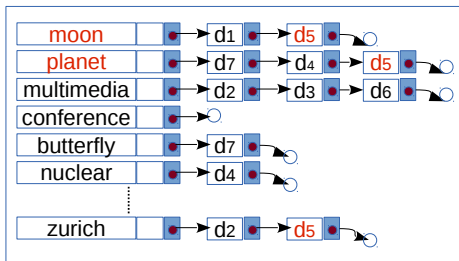


Issues involved in retrieval over inverted files (2)

- How we compute the Euclidean distance between query and docs in the inverted file?
- Notice that “zurich” inverted list will not be considered



Inverted file



Outline

- 1 Inverted files
- 2 Searching
- 3 Word Embedding and Co-occurrence Matrix

Retrieval on Inverted files (1)

- We have now mapped query terms into IDs
- Able to check all matched inverted lists
- How distance between query and docs from database is calculated?
- e.g. ℓ_2

$$d(q, d_j) = \sqrt{\sum_i (w_{iq} - w_{ij})^2} \quad (1)$$

- e.g. *Cosine* distance

$$\cos(q, d_j) = \frac{\sum_i w_{iq} \cdot w_{ij}}{\sqrt{\sum_i w_{iq}^2} \cdot \sqrt{\sum_i w_{ij}^2}} \quad (2)$$

Retrieval on Inverted files (2)

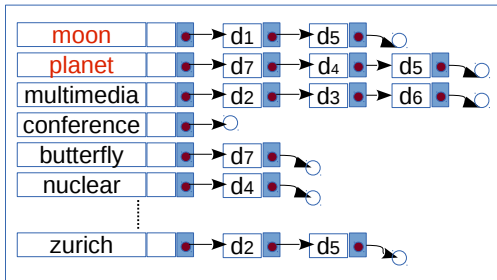
- Problem: Not all w_{ij} s will be included
- If word i does not appear in the query, w_{ij} is missing
- How distance between query and docs from database is calculated?

$$d(q, d_j) = \sqrt{\sum_i (w_{iq} - w_{ij})^2} \quad (3)$$

Retrieval on Inverted files (3)

- Maintain a rank list in the memory
- Update entries according to visited inverted lists

moon planet



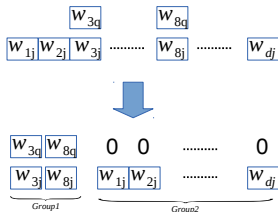
Rank list

d1		$+= (w_{id} - w_{iq})^2$
d2		
d3		
d4		$+= (w_{id} - w_{iq})^2$
d5		$+= (w_{id} - w_{iq})^2$
...		
dn-2		
dn-1		
dn		

Retrieval on Inverted files (4)

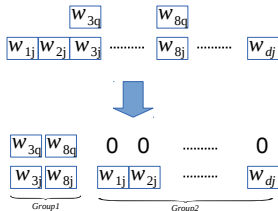
- Solution: a rank list is kept in the memory
- on which we can update from time to time
- w_{ij} s are partitioned into two groups
- Group1: those appear in the query; Group2: those do not

$$d(q, d_j) = \sqrt{\sum_i (w_{iq} - w_{ij})^2} \quad (4)$$



Retrieval on Inverted files (5)

- Solution: a rank list is kept in the memory
- on which we can update from time to time
- w_{ij} s are partitioned into two groups
- Group1 (G1): those appear in the query; Group2 (G2): those do not

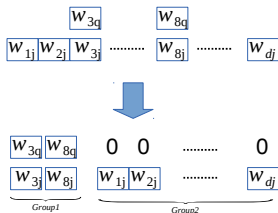


$$d^2(q, d_j) = \sum_{w_i \in G1} (w_{iq} - w_{ij})^2 + W_j - \sum_{w_i \in G1} w_{ij}^2, \quad (5)$$

where $W_j = \sum_i w_{ij}^2$

Retrieval on Inverted files (5)

- In *Cosine* distance
- Only terms that co-occur in the query and docs are considered
- $\sqrt{\sum_i w_{ij}^2}$ is pre-computed
- $\sqrt{\sum_i w_{iq}^2}$ is computed online



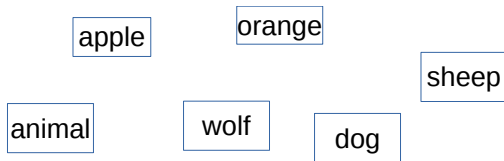
$$\cos(q, d_j) = \frac{\sum_i w_{iq} \cdot w_{ij}}{\sqrt{\sum_i w_{iq}^2} \cdot \sqrt{\sum_i w_{ij}^2}} \quad (6)$$

Outline

- 1 Inverted files
- 2 Searching
- 3 Word Embedding and Co-occurrence Matrix

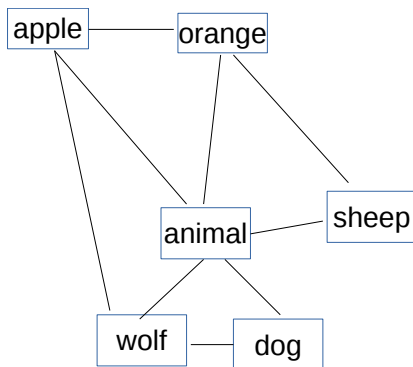
Overview on Word Embedding (1)

- In some scenario, we may need to evaluate the distance between words?
- For instance, how “apple” is similar to “orange”, in contrast to “dog”



- We are looking for a vector representation for a word
- Based on which the semantic distance between different words could be measured

Overview on Word Embedding (2)



- We are looking for a vector representation for a word
- Based on which the semantic distance between different words could be measured

Review on Bag-of-Words

	D1	D2	D3	Dn
apple	1	1	1		1
animal		1	1		1
dog					3
orange	2		2	
sheep		1			1
wolf					4

- We see above model before
- Each document is represented by frequencies of words

Word Embedding by Bag-of-Words

	D1	D2	D3	Dn
apple	1	1	1		1
animal		1	1		1
dog					3
orange	2		2	
sheep		1			1
wolf					4

- If we look at each row
- Each word is represented by n documents
- This way actually works!!

Motivation of Co-occurrence Matrix

- Besides Bag-of-Word model, there is another popular way
- It considers the co-occurrence rate of word appear in the same context

John likes apple and orange

Tom likes banna and cherries

- “apple” and “orange” play similar roles in the sentences of the same structure
- We have good reason to guess that they are something semantically similar or parallel
- “Co-occurrence” means “occur together”

Co-occurrence Matrix (1)

john like apple orange. john like football basketball.

Vocab.	john	like	apple	orange	football	basketball
john	0	2	0	0	0	0
like	2	0	1	0	1	0
apple	0	1	0	1	0	0
orange	0	0	0	0	0	0
football	0	1	0	0	0	1
basketball	0	0	0	0	1	0

- There is one mistake, please point it out for me:)

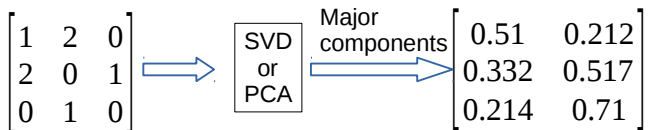
Co-occurrence Matrix (2)

john like apple orange. john like football basketball.

Vocab.	john	like	apple	orange	football	basketball
john	0	2	0	0	0	0
like	2	0	1	0	1	0
apple	0	1	0	1	0	0
orange	0	0	1	0	0	0
football	0	1	0	0	0	1
basketball	0	0	0	0	1	0

- 1 The matrix is symmetric
- 2 The diagonal elements are all zeros
- 3 It is sparse!!

From Co-occurrence Matrix to Word Embedding (1)



- Each row in the resulting matrix gives vector representation of one word
- This is the conventional way of word embedding

From Co-occurrence Matrix to Word Embedding (2)

$$C = \begin{bmatrix} 0 & 2 & 0 & 0 & 0 & 0 \\ 2 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\Downarrow$$

$$[U \ S \ V] = \text{svd}(C)$$

- Given $U'_{n \times k}$ takes the first k columns of U , $S'_{k \times k}$ takes the first k rows and columns of S
- Each row of F is the vector representation for a corresponding word

$$F = U'_{n \times k} \times S'_{k \times k} \quad (7)$$

From Co-occurrence Matrix to Word Embedding (3)

$$[U \ S \ V] = \text{svd}(C)$$

- Given $U'_{n \times k}$ takes the first k columns of U , $S'_{k \times k}$ takes the first k rows and columns of S
- Each row of F is the vector representation for a corresponding word

$$F = U'_{n \times k} \times S'_{k \times k}$$

$$d(i, j) = \cos(F(i, :), F(j, :)) \quad (8)$$

- We are now able to calculate the distance (usually *Cosine* distance) between words
- The larger the corpus is, the more it makes sense

Window Size for the Co-occurrence Matrix

- We can actually set the window size with different widths

john like apple orange. jonh like football basketball.

- When width equals to 3, we study of the co-occurrence of consecutive two words in the window

john like apple orange. jonh like football basketball.

Other Approaches for Word Embedding

- Recently, deep learning approaches based on Recurrent Neural Network is quite popular
 - 1 Continuous Bag-of-Word: $P(\text{word}|\text{context})$
 - 2 Skip-Gram: $P(\text{context}|\text{word})$

Q & A

Thanks for your attention!