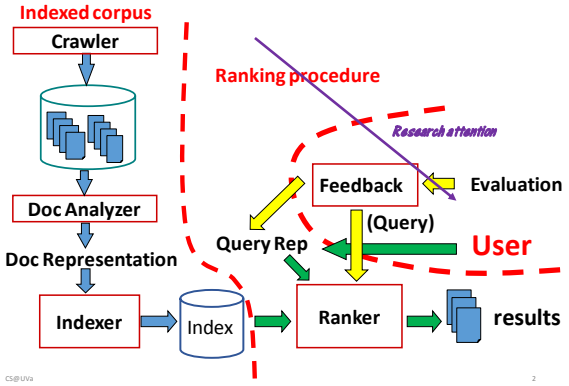
UIT
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

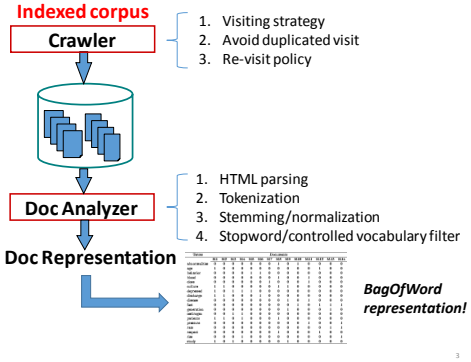
TRUY VẤN THÔNG TIN ĐA PHƯƠNG TIỆN
INFORMATION RETRIEVAL

INVERTED INDEX

1

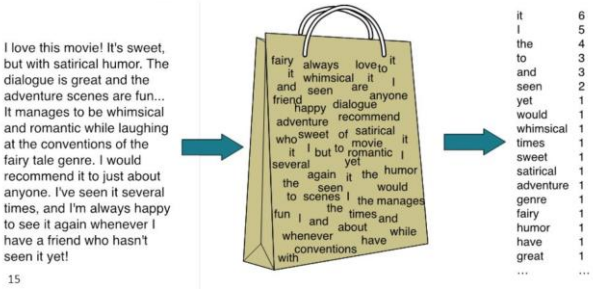


CS@UVa 2

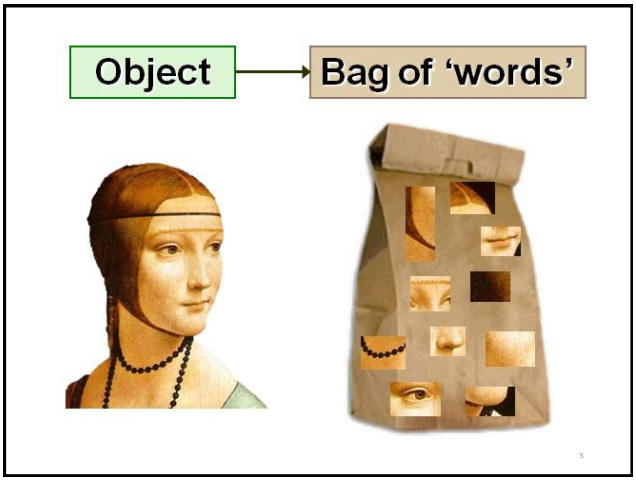


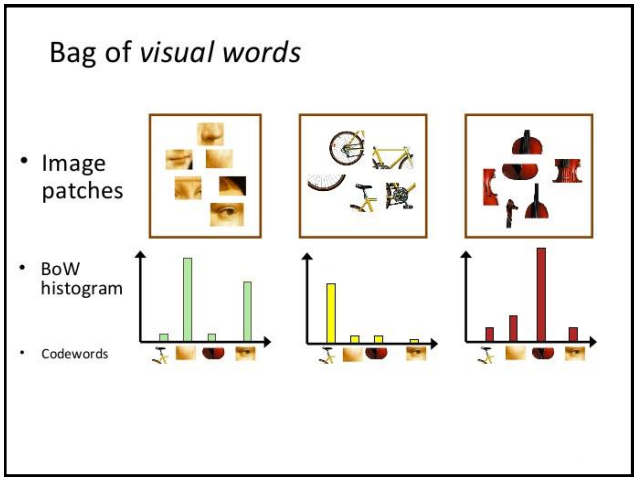
CS@UVa 3

The Bag of Words Representation



4


Object
Bag of 'words'


Bag of *visual words*
Image patches
BoW histogram
Codewords

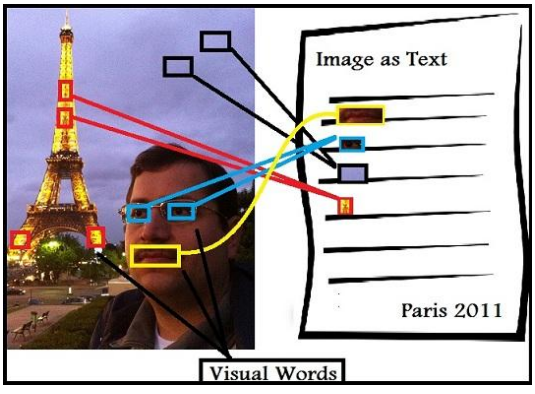
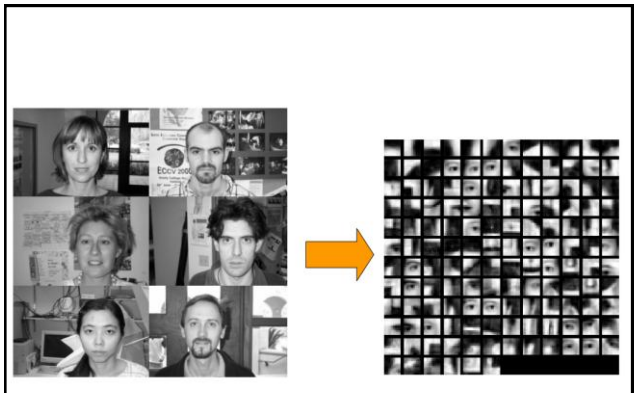
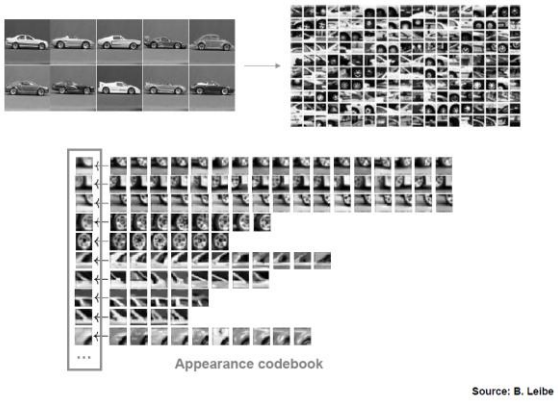
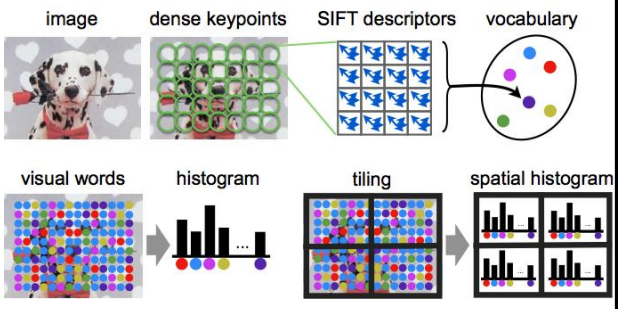
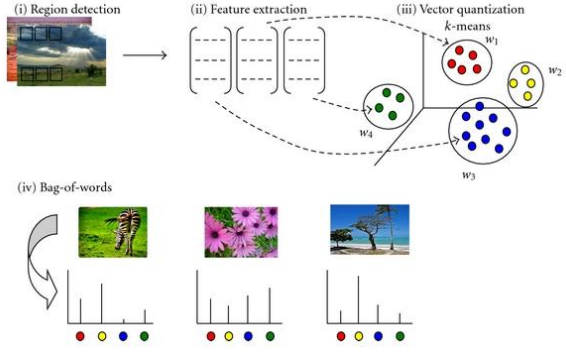


Image as Text
Paris 2011
Visual Words









Recap:

- Documents have been
  - Crawled from Web
  - Tokenized/normalized
  - Represented as Bag-of-Words
- Let's do search!
  - Query: "information retrieval"

	information	retrieval	retrieved	is	helpful	for	you	everyone
Doc1	1	1	0	1	1	1	0	1
Doc2	1	0	1	1	1	1	1	0

Phân tích độ phức tạp - Complexity analysis

- Space complexity analysis
  - $O(D * V)$
    - D is total number of documents and V is vocabulary size
  - Zipf's law: each document only has about 10% of vocabulary observed in it
    - 90% of space is wasted!
  - Space efficiency can be greatly improved by only storing the occurred words

Solution: linked list for each document

Phân tích độ phức tạp - Complexity analysis

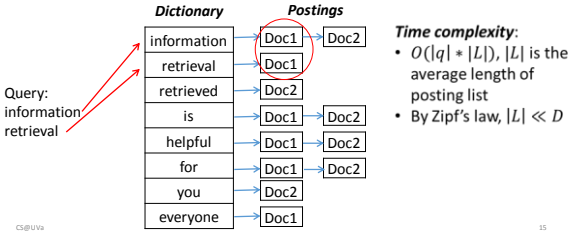
- Time complexity analysis
  - $O(|q| * D * |D|)$
    - $|q|$ is the length of query, $|D|$ is the length of a document

```
doclist = []
for (wi in q) {
    for (d in D) {
        for (wj in d) {
            if (wi == wj) {
                doclist += [d];
                break;
            }
        }
    }
}
return doclist;
```


Giải pháp: inverted index – chỉ mục ngược

Build a look-up table for each word in vocabulary

➔Thay vì từ Document tìm word thì đổi thành từ word kiếm document chứa nó



Time complexity:

- $O(|q| * |L|)$, $|L|$ is the average length of posting list
- By Zipf's law, $|L| \ll D$

Cấu trúc dữ liệu của inverted index

- **Bộ từ điển - Dictionary**: modest size
  - Needs fast random access
  - Stay in memory
    - Hash table, B-tree, trie, …
- Postings: huge
  - Sequential access is expected
  - Stay on disk
  - Contain docID, term freq, term position, …
  - Compression is needed

"Key data structure underlying modern IR"
- Christopher D. Manning

Zipf's law

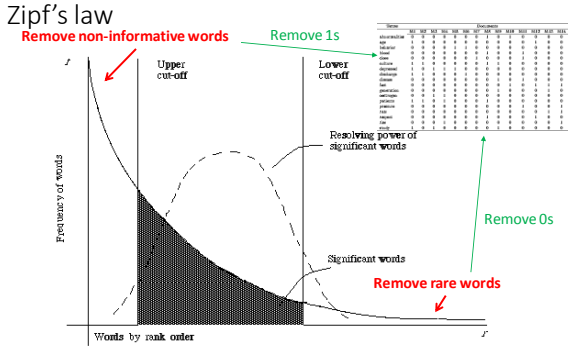


Figure 2.1. A plot of the hyperbolic curve relating f, the frequency of occurrence and r, the rank order (Adapted from Schultz[34] page 120)

TABLE II. SOME TYPICAL VIETNAMESE STOPWORDS

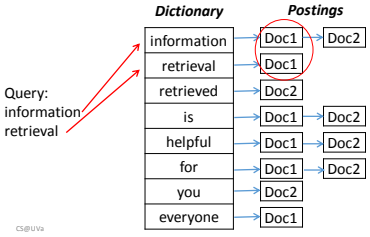
Có thể	Nếu	Vì vậy
Sau khi	Thì	Nếu không
Trước khi	Vì thế	Loại trừ
Tất cả	Cho nên	Một số
Những	Nhưng	Rõ ràng
Phần lớn	Bởi	Với
Hầu như	Là	Với lại

Stop-words appear with high frequency in Vietnamese documents. Hence, a number of past researches on Vietnamese text processing reject stop-words to return a better result. As a result, in our research we rejected stop-words from our

Nhắc lại về chuẩn hóa - Normalization

- Rule-based
  - Xóa dấu chấm, dấu gạch nối
  - Chuyển tất cả thành chữ thường
- Dictionary-based
  - Construct equivalent class
    - Car -> "automobile, vehicle"
    - Mobile phone -> "cellphone"
- Stemming

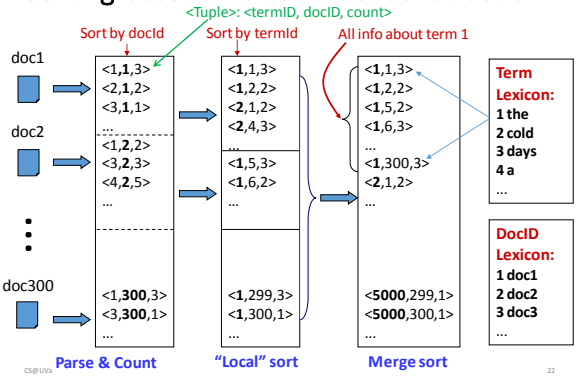
inverted index



Time complexity:

- $O(|q| * |L|)$, $|L|$ is the average length of posting list
- By Zipf's law, $|L| \ll D$

Sorting-based inverted index construction

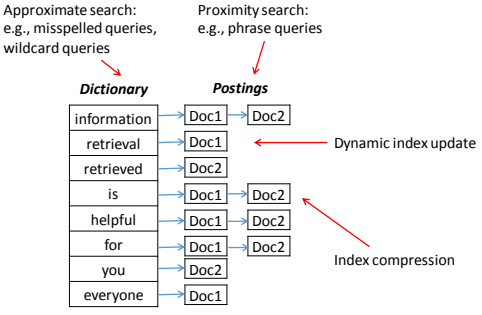


Sorting-based inverted index

- Challenges
  - Document size exceeds memory limit
- Key steps
  - Local sort: sort by termID
    - For later global merge sort
  - Global merge sort
    - Preserve docID order: for later posting list join

Can index large corpus with a single machine! Also suitable for MapReduce!

A close look at inverted index



Dynamic index update

- Periodically rebuild the index
  - Acceptable if change is small over time and penalty of missing new documents is negligible
- Auxiliary index
  - Keep index for new documents in memory
  - Merge to index when size exceeds threshold
    - Increase I/O operation
    - Solution: multiple auxiliary indices on disk, logarithmic merging

Index compression

- Benefits
  - Save storage space
  - Increase cache efficiency
  - Improve disk-memory transfer rate
- Target
  - Postings file

Tìm kiếm trên inverted index

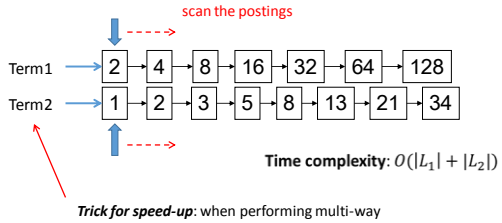
- Query processing
  - **Parse query syntax**
    - E.g., Barack AND Obama, orange OR apple
  - **Perform the same processing procedures as on documents** to the input query
    - Tokenization->normalization->stemming->stopwords removal

Tìm kiếm trên inverted index

- Procedures
  - Lookup query term in the dictionary
  - Retrieve the posting lists
  - Operation
    - AND: intersect the posting lists
    - OR: union the posting list
    - NOT: diff the posting list

Tìm kiếm trên inverted index

- Example: AND operation



Time complexity: $O(|L_1| + |L_2|)$

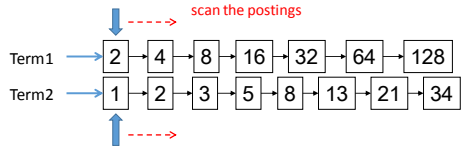
Trick for speed-up: when performing multi-way join, starts from lowest frequency term to highest frequency ones

Một số vấn đề với - Phrase query

- "computer science"
  - "He uses his computer to study science problems" is not a match!
  - We need the phase to be exactly matched in documents
  - N-grams **generally does not work for this**
    - Large dictionary size, how to break long phrase into N-grams?
  - We **need term positions in documents**
    - We can store them in inverted index

Một số vấn đề với - Phrase query

- Generalized postings matching
  - Equality **condition check with requirement of position pattern between two query terms**
    - e.g., T2.pos-T1.pos = 1 (T1 must be immediately before T2 in any matched document)
  - Proximity query: |T2.pos-T1.pos| ≤ k



More and more things are put into index

- Document structure
  - Title, abstract, body, bullets, anchor
- Entity annotation
  - Being part of a person's name, location's name

Một số vấn đề với - Spelling correction

- Tolerate the misspelled queries
  - "barck obama" -> "barack obama"
- Principles
  - Of various alternative correct spellings of a misspelled query, choose the ***nearest*** one
  - Of various alternative correct spellings of a misspelled query, choose the ***most common*** one

Một số vấn đề với - Spelling correction

- Proximity between query terms
  - Edit distance
    - Minimum number of edit operations required to transform one string to another
    - Insert, delete, replace
    - Tricks for speed-up
      - Fix prefix length (error does not happen on the first letter)
      - Build character-level inverted index, e.g., for length 3 characters
      - Consider the layout of a keyboard
        - E.g., 'u' is more likely to be typed as 'y' instead of 'z'

Một số vấn đề với - Spelling correction

- Proximity between query terms
  - Query context
    - "flew form Heathrow" -> "flew from Heathrow"
  - Solution
    - Enumerate alternatives for all the query terms
    - Heuristics must be applied to reduce the search space

Tài liệu tham khảo

Slide được tham khảo từ:

- http://www.cs.virginia.edu/~hw5x/Course/IR2015/_site/lectures/
- https://nlp.stanford.edu/IR-book/newslides.html
- https://course.ccs.neu.edu/cs6200s14/slides.html