Search Engines

Gertjan van Noord

October 14, 2024

Pagerank

How to determine if two vectors are (almost) identical?

Pagerank

```
# version 1

diff = sum(abs(new_ranks[i] - ranks[i]) for i in range(size))
if diff > ALFA:
    return True


# version 2

for i, j in zip(l1, l2):
    if abs(i - j) > ALFA:
        return True
return False
```

1. Boolean retrieval

- “inverted file”: for each term: in which document does it occur
- ordered list of identifiers: “postings list”
- efficient “merge” algorithm (or use efficient implementation of sets)
- efficiency: expressed as a function of the size of the input
- choice of appropriate data-structure (considerations of both speed and memory usage)

Efficiency

- expressed as a function of the size of the input

Example: constant C = 2, polynomial p = 4. Input size is n.

	function of n	10	100	1000	
constant	C	20	20	20	membership of hash
logarithmic	C*log(n)	2	4	6	binary search
linear	C*n	20	200	2000	membership unordered list
	C*n*log(n)	20	400	6000	merge sort
quadratic	C*n^2	200	20K	2G	intersection unordered lists
polynomial	C*n^p	20000	200M	1000G	context-free parsing (p=3)
exponential	C*2^n	2048	2.5e30	2.1e301	traveling salesman

2. Phrase queries

- Term vocabulary, tokenization, normalization
- phrases (sequences of words, “new york”)
- for each term, in which document does it occur, and in which positions

3. Suffix Arrays

- data-structure to represent a given corpus
- sorted sequence of all suffixes
- each suffix is represented by its starting position
- can be used to answer very long phrase queries
- and various other questions (longest repeated subsequences, frequency of very long subsequences, . . .)

4. Tolerant retrieval

- wildcard queries
- K-gram index: for each K-gram, return all words that contain that K-gram
- spell correction, Levenshtein distance
- dynamic algorithm to compute Levenshtein distance (and variants!)

5. Ranked retrieval

- tfidf
- representing documents by normalized vectors of tfidf values
- representing query as normalized vector of tfidf values
- best document $=$ highest cosine similarity

6. Evaluation

- Precision and recall.
- F-score = harmonic mean
- Precision and recall at rank
- Interpolated precision
- n-pt average precision, p@n, r@n, R-precision
- Mean Average Precision (MAP)
- Annotator Agreement, Kappa-score

7. Pagerank

- rank “better” websites higher
- consider random walk, all links equally probable
- small probability that you teleport to some place else
- together: probability matrix P
- probability vector x, start in random webpage
- apply P on x until no more changes: Pagerank
- higher probability in vector = more popular website
- weighted links

Literature

literature: Manning, Raghavan, Schütze, Introduction to Information Retrieval. Cambridge University Press 2008.

Chapters:

chapter 1

chapter 2

chapter 3

chapter 6 But not 6.4

chapter 8 But not 8.6 and 8.7

chapter 21 But not 21.3

In addition, all material discussed during lectures (refer to the slides). Wikipedia articles on suffix arrays and pagerank also provide relevant material.

Example Exam

Question 1

Boolean retrieval, posting lists, positional index. Consider the following documents:

```
Doc 1: Google drops plans for Berlin campus after protests
Doc 2: Google pulls plug on Berlin campus
Doc 3: Google reportedly drops plans for Berlin campus
Doc 4: Google drops plans for German campus
Doc 5: after protests, Google drops plans for the Berlin campus
```

1. Which documents will be retrieved by a boolean retrieval system for the query $(campus\ AND\ Berlin\ AND\ NOT(protests))$

2. Which documents will be retrieved by a boolean retrieval system for the query $(NOT(Google\ AND\ Berlin)\ OR\ protests)$

3. Give an equivalent but shorter query for $NOT(NOT(a)\ OR\ NOT(b))$

4. Which documents will be retrieved by a retrieval system supporting phrases for the phrasal query *drops plans*

5. Intersection of posting lists. Suppose you have two posting lists, l1 and l2. The lists contain the following documents:

```
l1 = [doc1,doc3,doc5,doc7,doc8,doc10,doc12,doc14]
l2 = [doc2,doc4,doc6,doc8,doc11,doc12,doc13]
```

   How many comparisons does it take in an efficient matching algorithm to find out that the intersection of the two lists contains the documents `doc8` and `doc12`?

6. Consider a retrieval system which supports phrases. For a particular query a b, the system found out that document 21 contains both a and b. The position lists for the terms a and b represent the positions where respectively a and b occur in document 21. What is a good data-structure for position lists to ensure that it is efficient to check if a phrase such as a b occurs in a given document?

Question 2

1. Give the minimum edit distance between *foot* en *float*, assuming that insertion and deletion has a cost of 2, and substitution has a cost of 3.

2. Provide the well-known matrix which contains the minimum edit distance between all prefixes of the given two words (using the same costs as in the previous question).

3. Indicate in the given matrix the cell which represents the minimum edit distance between *flo* and *fo*

Question 3

tf-idf$_{t,d}$ is the frequently used formula to assign a weight to a term t in a document d.

1. What is the definition of tf-idf$_{t,d}$?

2. Suppose we have 10 million documents, and a particular term a occurs in 10 thousand of those documents. Furthermore, for a particular document d46, the term a occurs 2 times. What is tf-idf$_{a,d46}$?

3. If a query contains multiple terms, how do we normally combine the tf-idf weights of all these terms to score documents?

Question 4

We are given a test-set containing 1000 documents. We have two queries. For query q1 we have checked that there are 8 relevant documents. For query q2 there are 6 relevant documents. Those documents are:

```
q1: d10 d20 d30 d40 d60 d80 d90 d100
q2: d5 d16 d38 d40 d42 d44
```

A particular retrieval system returns the following documents for these queries, in the given order:

```
q1: d60 d30 d40 d10 d11 d12 d20 d21 d22 d90 d80 d100
q2: d42 d44 d1 d2 d38 d3 d40 d4 d16 d5
```

1. Give Precision, Recall and F-score of this system for q1.
2. Give Precision, Recall and F-score of this system for q2.
3. What is de Mean Average Precision of this system for these queries?

4. Kappa is used to measure annotator agreement. It is defined as:

$$kappa = \frac{P(A) - P(E)}{1 - P(E)}$$

   where P(A) is the actual agreement, and P(E) is “chance”-agreement.

   Consider the following judgements:

```
ann1: blue red red red blue red yellow red    red yellow
ann2: blue red red red blue red yellow yellow red yellow
```

   What is kappa in this case? Please indicate how you obtain your answer.

5. How should we interpret this particular value of kappa? (if you were not able to compute kappa, assume that the resulting kappa value was 0.85).

Question 5

Suppose we have a web graph consisting of four nodes (1,2,3,4). The following links exist between the nodes: $1 \longrightarrow 2$, $1 \longrightarrow 4$, $2 \longrightarrow 1$, $2 \longrightarrow 3$ $3 \longrightarrow 1$ $4 \longrightarrow 3$

1. Give the transition probability matrix for a random surfer's walk with teleporting, where the teleport rate $\alpha = 0.3$.

2. Suppose the initial probability distribution vector $\vec{x_0} = (0, 0, 0, 1)$. Compute the vectors $\vec{x_1}$ en $\vec{x_2}$. (If you were not able to answer the previous question, you may assume an arbitrary transition probability matrix, where none of the transitions have a probability of 0. Of course, specify which matrix you assume).

Question 6

1. Suppose we have a string “achteraan”. Give its suffix array (as a sequence of integers).

2. How can a suffix array be used to find the longest repeated subsequence of a large corpus? Please describe in pseudo-code.

3. How much memory is required to represent the suffix array of a corpus of a given size X?