Information Retrieval SS 2023

Exercise 1: Boolean Retrieval, Phrase and Proximity Queries, Tolerant Retrieval

Benedikt Ebing

partially based on “An Introduction to Information Retrieval” by Manning, Raghavan and Schütze

Boolean Retrieval

Task 1, Task 2

Consider the following document collection (Assume the text is preprocessed by lowercasing and stopword removal. Also consider *Facebook's* as two tokens: *Facebook* and *'s)*.

Doc 1 Facebook's new tool is called Graph Search.
Doc 2 A new social graph for LinkedIn users.
Doc 3 Find friends using search on Facebook.
Doc 4 Google's Knowledge Graph lets you search for things, people, or places.

1. Draw the term-document incidence matrix for this document collection.
2. Draw the inverted index representation for this document collection, showing the dictionary and the postings.

term	Doc 1	Doc 2	Doc 3	Doc 4
facebook	1	0	1	0
‘s	1	0	0	1
new	1	1	0	0
tool	1	0	0	0
called	1	0	0	0
graph	1	1	0	1
social	0	1	0	0
linkedin	0	1	0	0
users	0	1	0	0
find	0	0	1	0
friends	0	0	1	0
using	0	0	1	0
search	1	0	1	1
google	0	0	0	1
knowledge	0	0	0	1
things	0	0	0	1
people	0	0	0	1
places	0	0	0	1

$$\begin{tikzcd}
\text{facebook} \arrow[r] & \boxed{1} \arrow[r] & \boxed{3} \\
\text{‘s} \arrow[r] & \boxed{1} \arrow[r] & \boxed{4} \\
\text{new} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \\
\text{tool} \arrow[r] & \boxed{1} \\
\text{called} \arrow[r] & \boxed{1} \\
\text{graph} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \\
\text{social} \arrow[r] & \boxed{2} \\
\text{linkedin} \arrow[r] & \boxed{2} \\
\text{users} \arrow[r] & \boxed{2} \\
\text{find} \arrow[r] & \boxed{3} \\
\text{friends} \arrow[r] & \boxed{3} \\
\text{using} \arrow[r] & \boxed{3} \\
\text{search} \arrow[r] & \boxed{1} \arrow[r] & \boxed{3} \arrow[r] & \boxed{4} \\
\text{google} \arrow[r] & \boxed{4} \\
\text{knowledge} \arrow[r] & \boxed{4} \\
\text{things} \arrow[r] & \boxed{4} \\
\text{people} \arrow[r] & \boxed{4} \\
\text{places} \arrow[r] & \boxed{4}
\end{tikzcd}$$

Task 3

Use both the term-document incidence matrix and the inverted index to compute the results return for the following queries:

- graph AND search
- graph AND NOT (google OR facebook)

graph AND search:

	1	1	0	1	graph
AND	1	0	1	1	search
	1	0	0	1	graph AND search

$$\begin{tikzcd}
\text{graph} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \\
\text{search} \arrow[r] & \boxed{1} \arrow[r] & \boxed{3} \arrow[r] & \boxed{4} \\
\text{graph AND search} \arrow[r] & \boxed{1} \arrow[r] & \boxed{4} &
\end{tikzcd}$$

Task 3

Use both the term-document incidence matrix and the inverted index to compute the results return for the following queries:

- graph AND search
- graph AND NOT (google OR facebook)

graph AND NOT (google OR facebook)

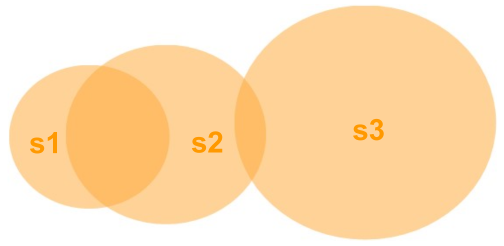
	0	0	0	1	google
OR	1	0	1	0	facebook
NOT	1	0	1	1	google OR facebook
	0	1	0	0	NOT (google OR facebook)
AND	1	1	0	1	graph
	0	1	0	0	graph AND NOT (google OR facebook)

$$\begin{tikzcd}
\text{google} \arrow[r] & \boxed{4} & & \\
\text{facebook} \arrow[r] & \boxed{1} \arrow[r] & \boxed{3} & \\
\text{google OR facebook} \arrow[r] & \boxed{1} \arrow[r] & \boxed{3} \arrow[r] & \boxed{4} \\
\text{NOT (google OR facebook)} \arrow[r] & \boxed{2} & & \\
\text{graph} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \\
\text{graph AND NOT (google OR facbook)} \arrow[r] & \boxed{2} & &
\end{tikzcd}$$

Task 4

For a conjunctive query, is processing postings lists in order of size guaranteed to be optimal? Explain why it is, or give an example where it isn't.

The order is not guaranteed to be optimal. Consider three terms with postings list sizes s1 = 100, s2 = 105 and s3 = 110. Suppose the intersection of s1 and s2 has length 100 and the intersection of s1 and s3 length 0. The ordering s1, s2, s3 requires 100+105+100+110=315 steps through the postings lists. The ordering s1, s3, s2 requires 100+110+0+0=210 steps through the postings lists.



Task 5

What is the complexity (in big O notation) for a query x AND y when the postings lists are sorted?

O(x+y)

Task 5 (cont.)

What if they aren't?

O(x*y)

Task 6

How should the Boolean query x AND NOT y be handled?

```
MERGE(x, y, AND NOT)
    answer <- ()
    while x != NIL and y != NIL
        do if docID(x) = docID(y)
            then x <- next(x)
            y <- next(y)
        else if docID(x) < docID(y)
            then ADD(answer, docID(x))
            x <- next(x)
        else
            y <- next(y)

    while x != NIL do
        ADD(answer, docID(x))
    return(answer)
```

Task 6 (cont.)

Why is the naive evaluation of the query, i.e. evaluating NOT y first and then x AND NOT y, normally very expensive?

Calculating (NOT y) first takes O(N) time, and then it will be merged with x. The overall complexity is O(N).

Task 6 (cont.)

How expensive is a Boolean query x OR y?

O(x+y)

Boolean Retrieval - Skip Pointers

Task 1

Why are skip pointers not useful for queries in the form x OR y?

It is essential to visit every docID in the postings list of either terms.

Task 2

Work out how many comparisons would be done to intersect the following two postings lists with the following two strategies mentioned in (i) and (ii).

p1 ← [4, 6, 10, 12, 14, 16, 18, 20, 22, 32, 47, 81, 120, 122, 157, 180], p2 ← [47]

(i) Using standard postings lists.
(ii) Using postings lists stored with skip-pointers, with a skip length of $\sqrt{L}$, as suggested in the lecture.

(i) Comparisons will be made unless either of the postings list end, i.e. till we reach 47 in the upper postings list, after which the lower list ends. Number of comparisons: 11

(ii) Number of comparisons: 6. The following comparisons will be made:

1. (4, 47),
2. (14, 47)
3. (22, 47)
4. (120, 47)
5. (32, 47)
6. (47, 47)

Task 3

Consider a postings intersection between this postings list, with skip pointers:

$$\begin{tikzcd}
3 \arrow[rrrr, bend left] & 5 & 9 & 15 & 24 \arrow[rrrr, bend left] & 39 & 60 & 68 & 75 \arrow[rrrr, bend left] & 81 & 84 & 89 & 92 \arrow[rrrr, bend left] & 96 & 97 & 100 & 115
\end{tikzcd}$$

and the following intermediate result postings list (which hence has no skip pointers):

3 5 89 95 97 99 100 101

(a) How often is a skip pointer followed?
(b) How many postings comparisons will be made by this algorithm while intersecting the two lists?
(c) How many postings comparisons are made if the postings lists are intersected without the use of skip pointers?

a) The skip pointer is followed once. (from 24 to 75)

skip-pointer

look-ahead

b) 18 comparisons are made: (3,3), (5,5), (9,89), (15,89), $\overset{\curvearrowright}{(24,89), (75,89)}$, (92,89), (81,89), (84,89), (89,89), (92,95), (115,95), (96,95), (96,97), (97,97), (100,99), (100,100), (115,101)

c) 19

Boolean Retrieval - Phrase & Proximity Queries

Task 1

Shown below is a portion of a positional index in the format: term: doc1: <position1, position2,...>; doc2: <position1, position2, ...>; etc.

angels:	2: [36,174,252,651];	4: [12,22,102,432];	7: [17];
fools:	2: [1,17,74,222];	4: [8,78,108,458];	7: [3,13,23,193];
fear:	2: [87,704,722,901];	4: [13,43,113,433];	7: [18,328,528];
in:	2: [3,37,76,444,851];	4: [10,20,110,470,500];	7: [5,15,25,195];
rush:	2: [2,66,194,321,702];	4: [9,69,149,429,569];	7: [4,14,404];
to:	2: [47,86,234,999];	4: [14,24,774,944];	7: [199,319,599,709];
tread:	2: [57,94,333];	4: [15,35,155];	7: [20,320];
where:	2: [67,124,393,1001];	4: [11,41,101,421,431];	7: [16,36,736];

Which document(s), if any, meet each of the following queries, where each expression within quotes is a phrase query

(i) "fools rush in"
(ii) "fools rush in" AND "angels fear to tread"

(i) All three documents (2, 4 and 7).

(ii) Only document 4.

Task 2

Consider the following fragment of a positional index with the same format:

Gates:	1: [3];	2: [6];	3: [2,17];	4: [1]	
IBM:		4: [3];	7: [14];		
Microsoft:	1: [1];	2: [1, 21];	3: [3];		5: [16,22,51]

(a) Describe the set of documents that satisfy the query Gates /2 Microsoft
(b) Describe each set of values for k for which the query Gates /k Microsoft returns a different set of documents as the answer.

a) Documents 1 and 3

b) {k=1} and K ∈ {x: x ≥ 5} return a different set than {1,3}

c) → {k=1} results in {3};
d) → K ∈ {x: x ≥ 5} results in {1, 2, 3}.

e) k ∈ {2, 3, 4} returns the result same set {1, 3}

Tolerant Retrieval

Task 1

Write down the entries in the permuterm index dictionary that are generated by the term *Retrieval*.

Retrieval$, etrieval$R, trieval$Re, rieval$Ret, ieval$Retr, eval$Retri, val$Retrie, al$Retriev, l$Retrieva, $Retrieval

Task 2

If you want to search for s*ng in a permuterm wildcard index, what key(s) would one do the lookup on?

ng$s*

Task 3

Consider the following example of a postings list in a 3-gram index.

$$\begin{tikzcd}
\text{etr} \arrow[r] & \boxed{\text{beetroot}} \arrow[r] & \boxed{\text{metric}} \arrow[r] & \boxed{\text{petrify}} \arrow[r] & \boxed{\text{retrieval}}
\end{tikzcd}$$

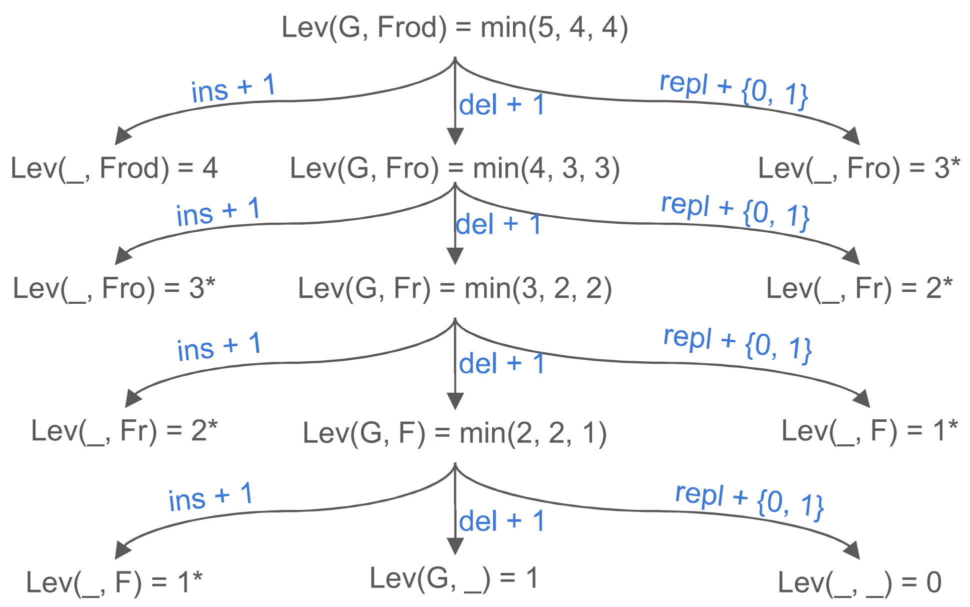
Why is it useful to have the vocabulary terms in the postings lexicographically ordered?

We want to use our n-gram index for filtering dictionary candidates -> Need to intersect n-gram postings. Hence, ordering the vocabulary terms allows for intersection in O(x + y) steps.

Task 4

We want to compute the Levenshtein edit distance between **Frodo** and **Gondor**. Consider the sub-problem of computing the distance between **G** and **Frod**. What are the costs for insertion, deletion and replacement respectively.

Recursive approach:



(insertion, deletion, replacement)

Insertion:	5
Deletion:	4
Replacement:	4

Interpretation of Lev(G, Frod):

Insertion: If we knew the distance of the sub-problem Lev(_, Frod), we could translate Frod to _, and have + 1 ins cost for inserting G.

Deletion: If we knew the distance of the sub-problem Lev(G, Fro), we could translate Fro to G and would be left with d. We'd need one delete operation (+1 del) for removing d.

Replacement: If we knew the distance of the sub-problem Lev(_, Fro), then we could translate Fro to _, and replace G with d (+1 repl, because G ≠ d).

* redundant computations

Explanation

Goal: Translate "Frodo" to "Gondor"

Lev(Frodo, Gondor)

1. Lev(Frod, Gondor) = x
--> If we knew how to translate "Frod" to "Gondor" we could do it and would be left "Gondor", x+1 cost for inserting "o"

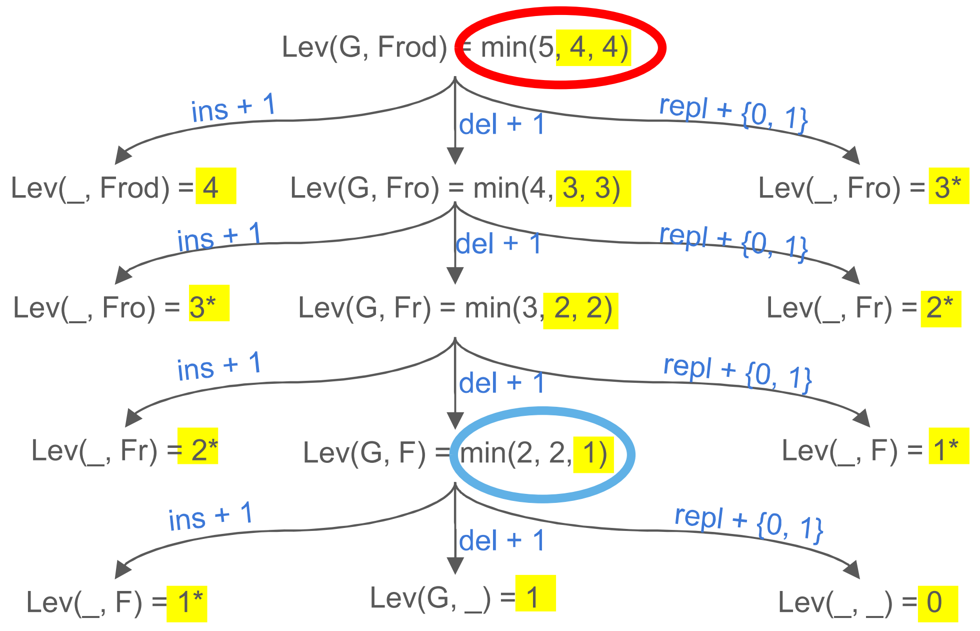
2. Lev(Frodo, Gondo) = y
--> If we knew how to translate "Frodo" to "Gondo" we could do it and would be left with "Gondo", y+1 cost for deleting "r„ from Gondor

3. Lev(Frod, Gondo) = z
--> If we knew how to translate "Frod" to "Gondo" we could do it and would have to replace the letter "o" with "r"

⇒ Interpretations of 1. and 2. swap if we translate "Gondor" to "Frodo" instead.

Task 4

We want to compute the Levenshtein edit distance between **Frodo** and **Gondor**. Consider the sub-problem of computing the distance between **G** and **Frod**. What are the costs for insertion, deletion and replacement respectively.



Dynamic programming approach:

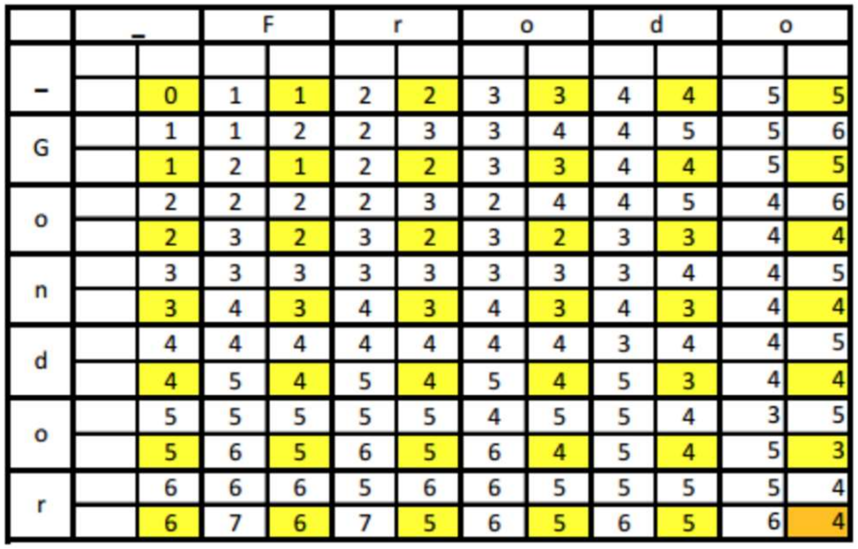
	_		F		r		o		d	
_										
		0	1	1	2	2	3	3	4	4
G		1	1	2	2	3	3	4	4	5
		1	2	1	2	2	3	3	4	4

replacement	insertion
Deletion	minimum

* redundant computations

Task 5

Write down the full 6x5 array of distances between all prefixes as shown in the lecture 3. What is the minimum edit-distance between **Frodo** and **Gondor**.



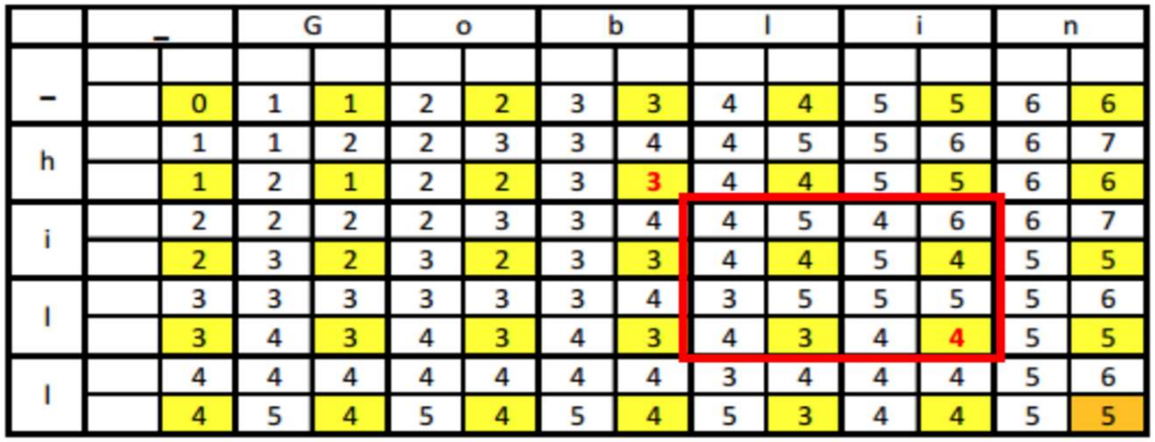
	_		F		r		o		d		o	
_		0	1	1	2	2	3	3	4	4	5	5
G		1	1	2	2	3	3	4	4	5	5	6
		1	2	1	2	2	3	3	4	4	5	5
o		2	2	2	2	3	2	4	4	5	4	6
		2	3	2	3	2	3	2	3	3	4	4
n		3	3	3	3	3	3	3	3	4	4	5
		3	4	3	4	3	4	3	4	3	4	4
d		4	4	4	4	4	4	4	3	4	4	5
		4	5	4	5	4	5	4	5	3	4	4
o		5	5	5	5	5	4	5	5	4	3	5
		5	6	5	6	5	6	4	5	4	5	3
r		6	6	6	5	6	6	5	5	5	5	4
		6	7	6	7	5	6	5	6	5	6	4

replacement	insertion
Deletion	minimum

Minimum edit-distance(Frodo, Gondor) = Lev(Frodo,Gondor) = 4

Task 6

What is the Levenshtein-Damerau distance between **hill** and **goblin**? Write down the solution in the same tabular format from the previous task.



	_		G		o		b		l		i		n	
–														
		0	1	1	2	2	3	3	4	4	5	5	6	6
h		1	1	2	2	3	3	4	4	5	5	6	6	7
		1	2	1	2	2	3	3	4	4	5	5	6	6
i		2	2	2	2	3	3	4	4	5	4	6	6	7
		2	3	2	3	2	3	3	4	4	5	4	5	5
l		3	3	3	3	3	3	4	3	5	5	5	5	6
		3	4	3	4	3	4	3	4	3	4	4	5	5
l		4	4	4	4	4	4	4	3	4	4	4	5	6
		4	5	4	5	4	5	4	5	3	4	4	5	5

Consider transposition only because the following condition is satisfied:

$$\underbrace{(a_{i-1} = b_j}_{l} \quad \& \quad \underbrace{a_i = b_{j-1})}_{i}$$

Levenshtein-Damerau(Gobli,hil) = min(insertion=5, replacement=5, deletion=4, transposition=3+1)

Levenshtein-Damerau(Goblin,hill) = 5

practice with your own examples: http://www.let.rug.nl/kleiweg/lev/

Task 7

What is the Jaccard coefficient between the word **bord** and each of **lord**, **morbid**, and **sordid** when we treat them as bigrams?

Jaccard(bord, lord)	= Jaccard({bo, or, rd,}, {lo, or, rd})	= 2 / 4
Jaccard(bord, morbid)	= Jaccard({bo, or, rd}, {mo, or, rb, bi, id})	= 1 / 7
Jaccard(bord, sordid)	= Jaccard({bo, or, rd}, {so, or, rd, di, id})	= 2 / 6