

Ex 4) We are looking for all occurrences of Gates and Microsoft in the same documents separated by a maximum of one word.

* For each positional index in Gates we check if the same docID appears in the positional index of Microsoft

For each matching documentID we compare each ~~pos~~ position in the positional index of Gates ~~with~~ for this docID with each position in the positional index of Microsoft for the ~~same~~ matching docID.

$\{(1: (<3>, <1>)), (3: (<2>, <3>))\}$

Ex 5)

Permuterm index is simple but needs a considerable number of rotations per term. This can lead to a dramatic increase in disc and memory space.

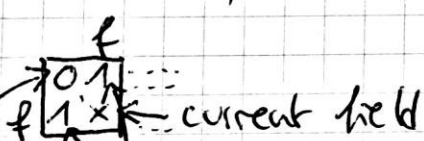
k-gram querying can be expensive because of all the additional lookup (special index, filtering and inverted index). However k-gram index is more space efficient.

Ex 6) Levenshtein distance of "flower" and "flour"

	f	l	o	w	e	r
f	0	1	2	3	4	5
l	1	0	1	2	3	4
w	2	1	0	1	2	3
e	3	2	1	0	1	2
r	4	3	2	1	1	2
	5	4	3	2	2	2
	6	5	4	3	3	2

- We write an incrementing sequence of number in the first row and column starting with 0.

- We always look at four fields at the same time



- if one of these values is the minimum, ~~take the~~ $x = \text{the min value} + 1$

- if this value is the minimum, ~~we~~ $x = \text{this value} + \text{substitution cost}$ where substitution cost = 1 if the ~~same~~ character in the current row and column don't match, else 0

$\text{Lev}(\text{flower}, \text{flour}) = 2$