



TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií



Vyhledávání

a deep learning

František Kynych
20. 10. 2022 | MVD





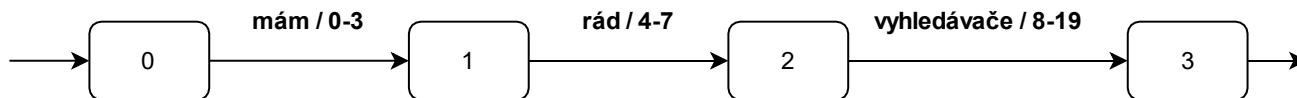
Část I.: Úvod do vyhledávání, základní metody

Co je to vyhledávač?

- Program používaný k získávání informací
 - Získává data -> poskytuje užitečné informace
- Dělíme na:
 - Obecné vyhledávání – Google, DuckDuckGo, Bing, Baidu, ...
 - Vertikální vyhledávání
 - Zaměřují se pouze na určitou oblast – Google Scholar
- Základní úlohy vyhledávače
 - Indexování
 - Efektivní příjem a ukládání dat
 - Dotazování
 - Poskytování funkce vyhledávání
 - Hodnocení
 - Správné hodnocení výsledků pro návrat relevantních výsledků

Základní prvky textové analýzy

- Tokenizér
 - Rozdělí proud textu na slova, fráze nebo jiné prvky (tokeny)
- Token filter
 - Přijímá proud tokenů, které modifikuje, maže, nebo přidává další
- Základní zpracování textu „Mám rád vyhledávače“



- Dělení na slova pomocí mezer, lower case, držení pozice
 - V praxi komplexnější analyzéry – stopwords list, stemming, ...

Indexování

- Dokument
 - Kniha, článek, webová stránka, ...
- Proces zpracování dokumentů a uložení jednotlivých slov (tokenů) s odkazy na cílový dokument
 - Invertovaný soubor (index)

Slovo	Doc id
metoda	1
vytěžování	1, 2
data	1
neuron	3
vyhledávání	1, 2, 3

Vyhledávání

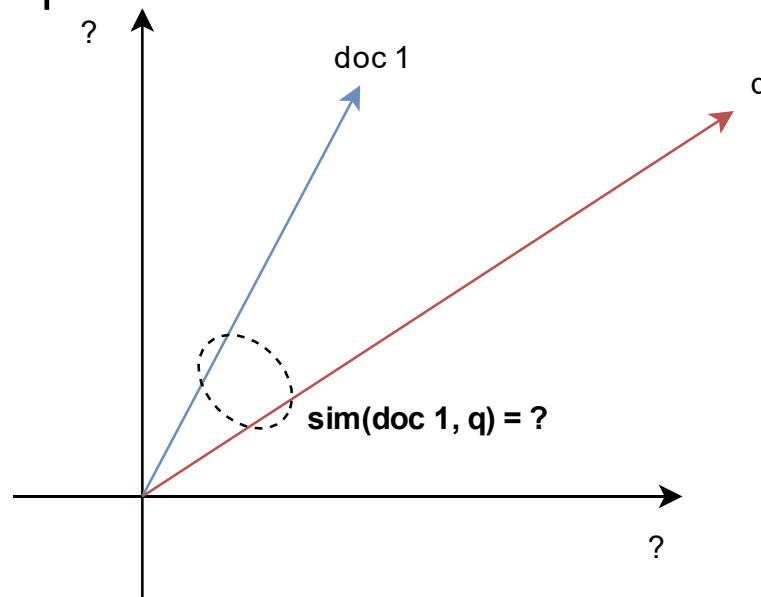
- Na počátku se podporovalo pouze vyhledávání klíčových slov
- Query parser
 - Zodpovědný za správné parsování dotazu
 - Vyhledávání „latest research in artificial intelligence“
 - Měl by poznat, že artificial intelligence je jedna fráze
 - Měl by dokázat převést slovo latest na časové omezení dotazu (např. stáří < 3 měsíce)
- Textová analýza
 - Pipeline podobná jako u indexování (rozdělení slov, lower case, stemming)
- Navrácení výsledků
 - Invertovaný index -> navrácení **všech** výsledků s danými slovy neseřazeně

Hodnocení relevance (ranking)

- Nejdůležitější část vyhledávače
 - Prvních 10 výsledků by mělo být nejvíce relevantních
 - Uživatel nemusí procházet všechny dokumenty – menší zátěž systému
- Retrieval model
 - Zodpovědný za ohodnocení všech výsledků vyhledávání
 - Každý dokument obdrží skóre -> seřazení výsledků
- Jak vybrat správný model?
 - Velké množství metod pro hodnocení
 - Často se relevance dokumentu mění s časem
 - Potřeba průběžně přizpůsobovat

Model vektorového prostoru

- Označujeme jako **VSM** (Vector Space Model)
- Dokumenty i dotazy jsou reprezentovány pomocí vektorů
- Základní princip:



VSM – bag-of-words implementace

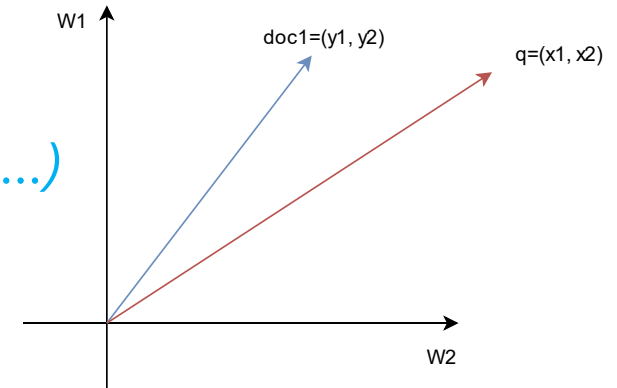
- Bag-of-words
 - „Metody vytěžování dat“ -> [“Metody”, “vytěžování”, “dat”]
 - BoW = {“Metody”: 1, “vytěžování”: 1, “dat”: 1}
- Pro zjednodušení pouze binární přístup a relevance pomocí skalárního součinu

• Máme $q=(x_1, x_2, x_3, \dots)$ a $doc1=(y_1, y_2, y_3, \dots)$

- Slova $x_i, y_i \in \{0, 1\}$
 - 0: slovo není v textu
 - 1: slovo je v textu

• $sim(q, doc1) = x_1y_1 + x_2y_2 + x_3y_3 + \dots = \sum_{i=1}^N x_iy_i$

- Dimenze vektoru = N (velikost slovníku)



VSM – bag-of-words příklad

$q = \text{„Metody vytěžování dat“}$

$doc1 = \text{„... využité metody ...“}$

$doc2 = \text{„... metody vytěžování dat ...“}$

Slovník $V = \{\text{využité, metody, vytěžování, dat}\}$

$q = (0, 1, 1, 1)$

$doc1 = (1, 1, 0, 0)$

$doc2 = (0, 1, 1, 1)$

$\text{sim}(q, doc1) = 0 \times 1 + 1 \times 1 + 0 \times 0 + 0 \times 0 = 1$

$\text{sim}(q, doc2) = 0 + 1 + 1 + 1 = 3$

TF-IDF

- V praxi potřebujeme použít počet výskytů
- TF = term frequency
 - Kolikrát se výraz vyskytuje v dokumentu
- Problém s TF
 - Pokud se slovo vyskytuje často v každém dokumentu (spojky, předložky, ...)
- IDF = inverse document frequency
 - V jaké míře se slovo vyskytuje v celé kolekci
 - Využito k penalizaci častých slov

TF-IDF

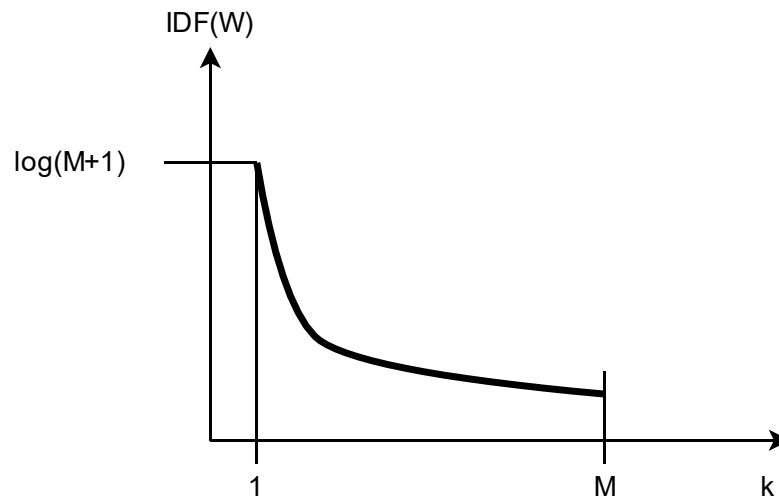
$$x_i = c(W_i, q) \quad (\text{počet slov } W_i \text{ v dotazu } q)$$

$$y_i = c(W_i, d) \times IDF(W_i)$$

$$IDF(W) = \log\left(\frac{M+1}{k}\right)$$

M = celkový počet dokumentů v kolekci

k = celkový počet dokumentů obsahující slovo W



TF-IDF

- Výsledná rovnice při použití TF-IDF a počítání relevance pomocí skalárního součinu:

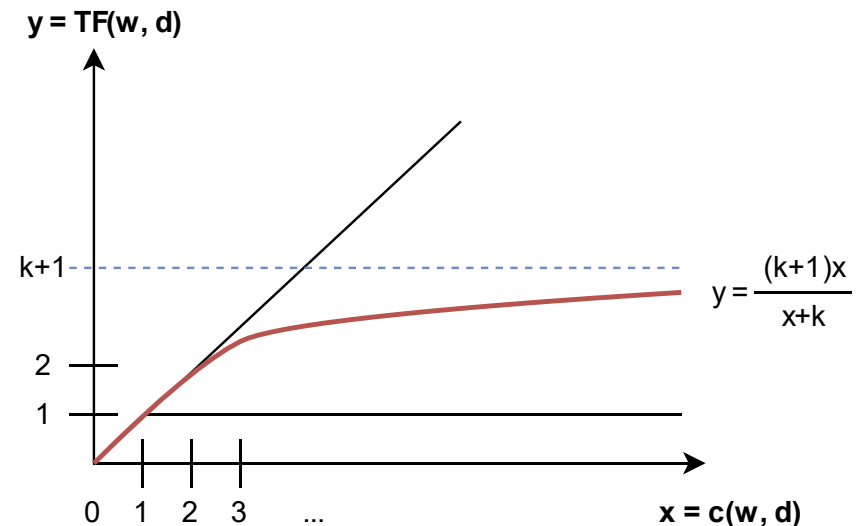
$$\text{sim}(q, d) = \sum_{i=1}^N x_i y_i = \sum_{w \in q \cap d} c(w, q) c(w, d) \log\left(\frac{M + 1}{df(w)}\right)$$

M = celkový počet dokumentů v kolekci

$df(w)$ = celkový počet dokumentů obsahující slovo w (document frequency)

Okapi BM25

- Úprava TF-IDF pomocí TF transformace ($TF(w, d)$)
- Problém TF-IDF
 - Jedno časté slovo z dotazu může dominovat nad ostatními
 - Např. často v nějakém nerelevantním dokumentu, ale méně často v celé kolekci
- Parametr $k \geq 0$
 - $k = 0$: binární TF
 - Příliš velké k : $y = x$ (TF)



Okapi BM25

- Výsledná rovnice při použití TF transformace a počítání relevance pomocí skalárního součinu:

$$\text{sim}(q, d) = \sum_{i=1}^N x_i y_i = \sum_{w \in q \cap d} c(w, q) \frac{(k+1)c(w, d)}{c(w, d) + k} \log\left(\frac{M+1}{df(w)}\right)$$

- Kromě TF transformace se používá normalizace délky dokumentu
 - Dlouhý dokument bude mít větší šanci vyhledání uživatelskými dotazy

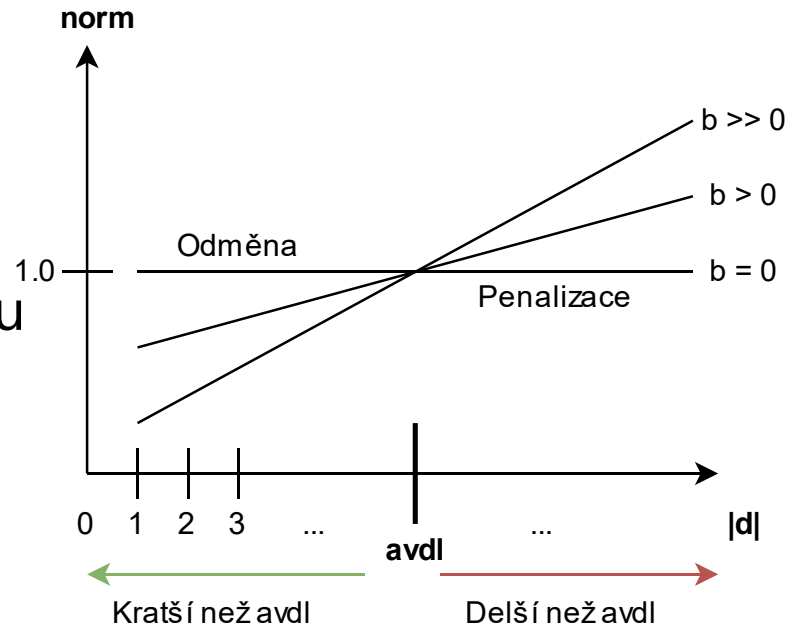
Okapi BM25

- Penalizace s pomocí parametru $b \in [0, 1]$

$$norm = 1 - b + b \frac{|d|}{avdl}$$

$|d|$ = délka dokumentu

$avdl$ = průměrná délka dokumentu



Okapi BM25

- Výsledná rovnice BM25 při počítání relevance pomocí skalárního součinu:

$$\text{sim}(q, d) = \sum_{i=1}^N x_i y_i = \sum_{w \in q \cap d} c(w, q) \frac{(k+1)c(w, d)}{c(w, d) + k(1 - b + b \frac{|d|}{\text{avdl}})} \log\left(\frac{M+1}{\text{df}(w)}\right)$$

- V praxi se používá rozšířený BM25
 - BM25F pro strukturované dokumenty (F = fields)
 - BM25+ zabraňuje příliš velké penalizaci dlouhých dokumentů
- BM25 používá defaultně [Elasticsearch](#)

Limitace předchozích metod

- Bag-of-words
 - Rozdělení textu na jednotlivá slova
 - Nezáleží na pořadí slov v dotazu
- Fungují hůře při vyhledávání nad delšími dokumenty
- Problém při špatném definování dotazu
 - Nelze nalézt synonyma
- Pokud není potřeba řešit některý ze zmíněných problémů, tak je BM25(F/+) dostačující
 - Jinak se používá jako baseline



Část III.: Vyhodnocení vyhledávání

Vyhodnocení vyhledávání

- Pro obecné vyhodnocení se používají datasety z Text Retrieval Conference (NIST [TREC](#))
- Při použití Apache Lucene lze vyhodnocení vytvořit pomocí [Lucene4IR](#) nástroje
 - Vyhodnocuje nad [CACM](#) datasetem (názvy a abstrakty článků)
 - Obsahuje nástroj k vyhodnocení nad TREC datasetem

Co vyhodnocovat?

- Efektivnost / přesnost
 - Měříme schopnost vyhledávání relevantních dokumentů
- Výkonnost
 - Jak dlouho trvá získání výsledků?
 - Jak moc je systém zatížený?
- Použitelnost
 - Je vytvořený systém vhodný pro danou aplikaci?

Precision + Recall

- Precision
 - Podíl navrácených výsledků, které jsou relevantní
- Recall
 - Podíl relevantních dokumentů, které jsou navraceny

Dokument	Navracený	Nenavracený
Relevantní	a	b
Nerelevantní	c	d

$$Precision = \frac{a}{a + c}$$

$$Recall = \frac{a}{a + b}$$

- Ideální stav -> Recall = Precision = 1
- Často definován cutoff (např. precision @ 10 docs)

F-Measure

- Kombinace Precision (P) a Recall (R)
- Parametr β , většinou hodnota 1
 - Ovládá jaký důraz je kladen na Precision nebo Recall

$$F_{\beta} = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}$$

$$F1 = \frac{2PR}{P + R}$$

Precision Recall křivka

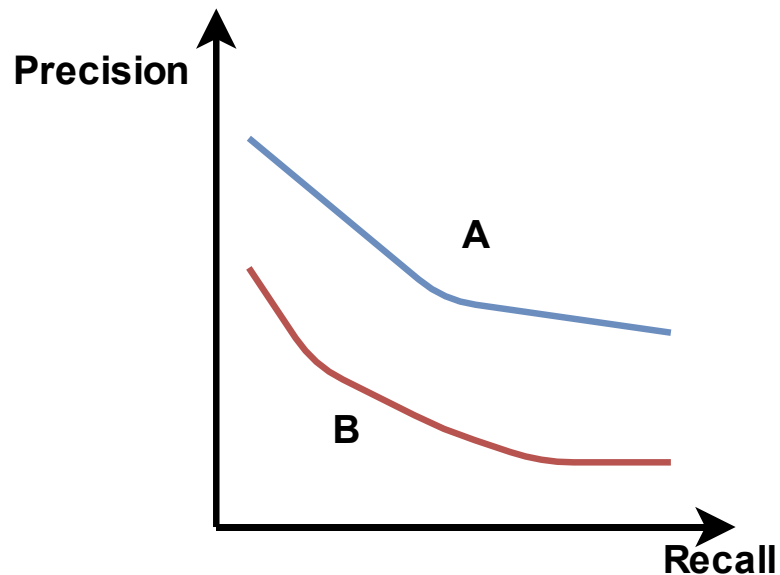
- Např. máme 10 relevantních dokumentů (+ relevantní, - nerelevantní)

Dokumenty	Precision	Recall
D1 +	1/1	1/10
D2 +	2/2	2/10
D3 -	2/3	2/10
D4 -
D5 +	3/5	3/10
D6 -
D7 -
D8 +	4/8	4/10
?	?	10/10

- Z těchto bodů vytvoříme křivku

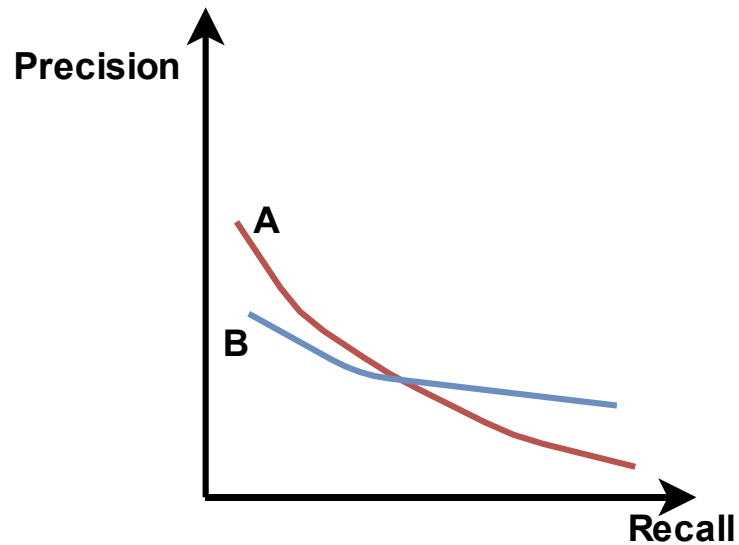
Precision Recall křivka

- Máme dva systémy A a B, který je lepší?



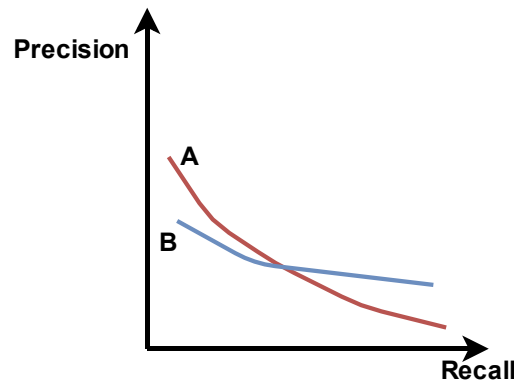
Precision Recall křivka

- Máme dva systémy A a B, který je lepší?



Precision Recall křivka

- Máme dva systémy A a B, který je lepší?



- Záleží na aplikaci
 - Uživatel chce mít při vyhledávání prvních pár výsledků nejrelevantnějších -> systém A
 - Např. chci zjistit, co se dnes děje
 - Někdo chce naopak projít co nejvíce výsledků -> systém B
 - Např. chci zjistit, zda můj nápad už někdo realizoval dříve

Normalized Discounted Cumulative Gain (NDCG)

- Vyhodnocení pro různé úrovně relevance (1 = nerelevantní, 2 = částečně relevantní, 3 = relevantní)

Doc	Gain	Cumulative gain	Discounted Cumulative Gain
D1	3	3	3
D2	2	3+2	$3 + \frac{2}{\log 2}$
D3	1	3+2+1	$3 + \frac{2}{\log 2} + \frac{1}{\log 3}$
D4	1	3+2+1+1	...
D5	3	3+2+1+1+3	...

$$NDCG = \frac{DCG@5}{IdealDCG@5}$$

$$DCG@5 = 3 + \frac{2}{\log 2} + \dots + \frac{3}{\log 5}$$

$$IdealDCG@5 = 3 + \frac{3}{\log 2} + \frac{3}{\log 3} + \frac{3}{\log 4} + \frac{2}{\log 5}$$



Užitečná literatura / kurzy

- TEOFILI, Tommaso. *Deep learning for search*. Shelter Island: Manning, [2019]. ISBN 978-161-7294-792.
- Coursera kurz [Text Retrieval and Search Engines](#)

