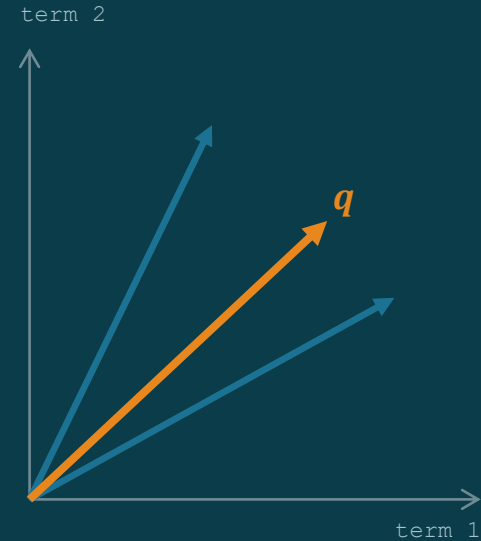


PERTEMUAN 4

Vector Space Model dan Pembobotan TF-IDF

[cocok; tidak cocok] → daftar hasil yang terurut menurut relevansi



Pertemuan 3 Memiliki Satu Masalah

Query Boolean menjawab cocok atau tidak, tetapi pengguna membutuhkan urutan

Korpus yang sama dengan pertemuan 3

D1

banjir landa surabaya

D2

banjir bandang terjang
kota

D3

surabaya kota pahlawan

D4

kota surabaya banjir

banjir AND surabaya

Hasil: D1, D4 tidak memiliki urutan. D2 dan D3 dibuang seluruhnya, padahal masing-masing memuat salah satu kata.

Terlalu sedikit

banjir OR surabaya

Hasil: D1, D2, D3, D4 → seluruh koleksi. Pada jutaan dokumen, pengguna tidak tahu harus mulai membaca dari mana.

Terlalu banyak

Beri tiap dokumen sebuah skor antara 0 dan 1, dan tampilkan dokumen dengan skor tertinggi lebih dulu.

Capaian Pembelajaran Pertemuan Ini

Sub-CPMK-4 — turunan dari CPMK-07.1 dan CPL-TF-07

Sub-CPMK-4

Mahasiswa mampu menerapkan pembobotan term dan Vector Space Model untuk perangkingan dokumen.

Ranah: C3 (menerapkan), P3 (presisi)

Indikator penilaian

- 1 Ketepatan menghitung TF-IDF secara manual dan terprogram
- 2 Ketepatan menerapkan normalisasi panjang dokumen
- 3 Kebenaran perangkingan dokumen berdasarkan cosine similarity

Bentuk penilaian: latihan perhitungan dan laporan praktikum (formatif). Mesin pencari hari ini menjadi baseline Tugas Mandiri 1 pada pertemuan 5.

Bag of Words dan Term Frequency

Dokumen direpresentasikan sebagai kumpulan term beserta jumlah kemunculannya

Banjir kembali merendam Surabaya. Warga Surabaya mengungsi karena banjir setinggi pinggang, dan banjir diperkirakan belum surut hingga besok.

Term frequency $tf(t, d)$

banjir

3

surabaya

2

rendam

1

warga

1

ungsi

1

surut

1

Sebagian term lain disembunyikan agar ringkas.

Asumsi bag of words

Urutan kata diabaikan. Dua kalimat berikut menjadi vektor yang identik:

"Persebaya mengalahkan Arema"

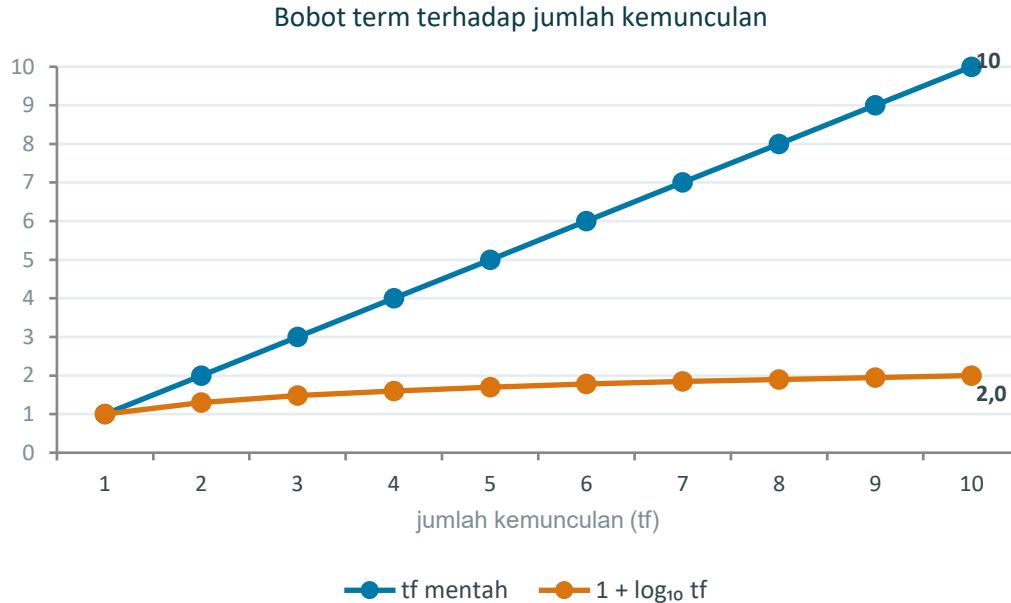
"Arema mengalahkan Persebaya"

Kasar, tetapi terbukti cukup baik untuk perangkingan.

Intuisi pertama: dokumen yang lebih sering menyebut term pada query kemungkinan besar lebih relevan terhadap query tersebut.

Mengapa Frekuensi Saja Tidak Cukup

Dokumen yang menyebut “banjir” 20 kali tidak 20 kali lebih relevan daripada yang menyebutnya sekali



Pembobotan logaritmik

$$w = 1 + \log_{10} tf \quad \text{jika } tf > 0$$
$$w = 0 \quad \text{jika } tf = 0$$

tf 1 $\rightarrow$ 1,0

tf 10 $\rightarrow$ 2,0

tf 1000 $\rightarrow$ 4,0

Kemunculan tambahan tetap menaikkan bobot, tetapi makin lama makin kecil pengaruhnya. Ini juga meredam dokumen spam yang mengulang kata kunci.

Inverse Document Frequency

Term yang muncul di hampir semua dokumen tidak banyak membantu membedakan dokumen

$$\text{idf}(t) = \log_{10} (N / \text{df}(t))$$

N = jumlah dokumen, $\text{df}(t)$ = jumlah dokumen yang memuat t

term	df	idf
banjir	3	0,125
kota	3	0,125
surabaya	3	0,125
bandang	1	0,602
landa	1	0,602
pahlawan	1	0,602
terjang	1	0,602

Korpus contoh, $N = 4$

Membaca tabel

“pahlawan” hanya ada di D3, sehingga kemunculannya sangat informatif. “banjir” ada di tiga dari empat dokumen, sehingga bobotnya hampir lima kali lebih kecil.

Mengapa df, bukan total kemunculan?

Term yang muncul 50 kali di satu dokumen dan term yang muncul sekali di 50 dokumen punya total sama, tetapi yang pertama jauh lebih membedakan. df menangkap persebaran; total kemunculan tidak.

Menggabungkan Keduanya: TF-IDF

Skema pembobotan paling dikenal dalam sejarah Information Retrieval

$$w(t, d) = (1 + \log_{10} tf(t, d)) \times \log_{10} (N / df(t))$$

Tinggi

Term sering muncul di dokumen ini, tetapi jarang di dokumen lain

“pahlawan” di D3

Rendah

Term jarang di dokumen ini, atau tersebar di banyak dokumen

“banjir” di D1

Nol

Term tidak muncul di dokumen, atau muncul di semua dokumen

$df = N \rightarrow idf = 0$

Faktor tf dihitung per dokumen; faktor idf dihitung sekali untuk seluruh koleksi dan dapat disimpan di dictionary.

Latihan: Matriks Bobot TF-IDF Korpus Contoh

Seluruh *tf* bernilai 1, sehingga bobot setiap term sama dengan *idf*-nya

term	D1	D2	D3	D4
bandang	–	0,602	–	–
banjir	0,125	0,125	–	0,125
kota	–	0,125	0,125	0,125
landa	0,602	–	–	–
pahlawan	–	–	0,602	–
surabaya	0,125	–	0,125	0,125
terjang	–	0,602	–	–
panjang $\ d\ $	0,627	0,870	0,627	0,216

Panjang vektor

$$\|d\| = \sqrt{(\sum w^2)}$$

Contoh D1:

$$\sqrt{(0,125^2 + 0,602^2 + 0,125^2)} = 0,627$$

Coba sendiri

Hitung ulang panjang D2 dan D4. Mengapa D4 jauh lebih pendek meskipun jumlah katanya sama dengan D3?

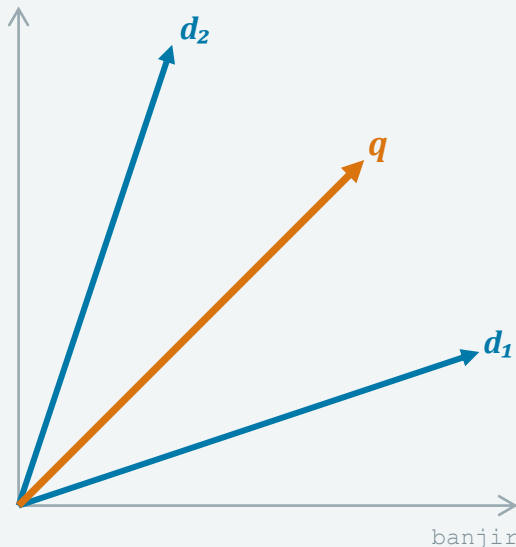
Nilai dibulatkan tiga angka di belakang koma. Sel bertanda “–” berarti term tersebut tidak muncul (bobot = nol).

Dokumen dan Query sebagai Vektor

Setiap term adalah satu sumbu; setiap dokumen adalah satu titik dalam ruang berdimensi $|V|$

surabaya

Ilustrasi dua dimensi



Dimensi sangat banyak

Kosakata puluhan ribu term berarti puluhan ribu sumbu. Kita tidak dapat membayangkannya, tetapi rumusnya tetap berlaku.

Vektor sangat jarang

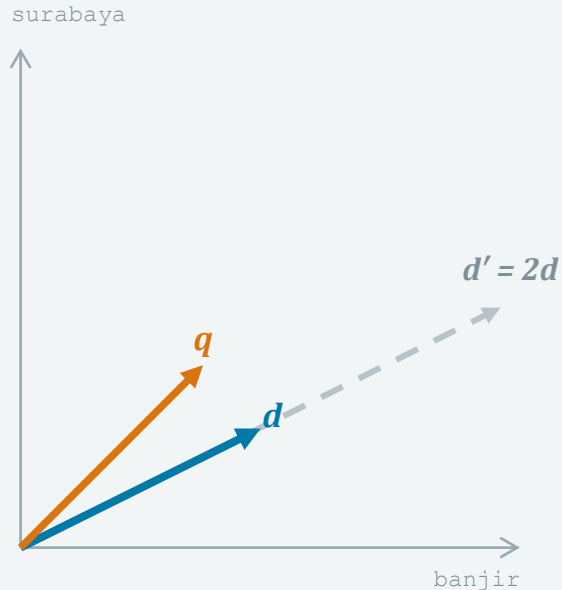
Hampir seluruh koordinat bernilai nol $\rightarrow$ sama dengan pengamatan tentang *incidence matrix* pada pertemuan 3.

Query diperlakukan sama

Query adalah dokumen pendek. Ia diproses dan dibobot dengan cara yang sama, dan diletakkan di ruang yang sama.

Mengapa Sudut, Bukan Jarak?

Eksperimen pikiran: salin isi dokumen d dua kali menjadi dokumen d'



Menurut jarak Euclid

d' jauh dari q , sedangkan d dekat. Padahal isinya identik $\rightarrow d'$ hanya lebih panjang.

Menurut sudut

d dan d' terletak pada garis yang sama, sehingga sudutnya terhadap q persis sama. Keduanya dinilai sama relevan.

Arah vektor menyatakan topik dokumen; panjangnya sebagian besar mencerminkan panjang teks. Yang kita bandingkan adalah topik.

Cosine Similarity

Mengukur kesamaan arah dua vector, yang nilainya antara 0 dan 1 untuk bobot non-negatif

$$\cos(q, d) = \frac{q \cdot d}{(\|q\| \times \|d\|)} = \frac{\sum w(t, q) \cdot w(t, d)}{(\|q\| \times \|d\|)}$$

1

Pembilang

Hasil kali titik. Hanya term yang ada di query dan di dokumen yang menyumbang, sedangkan term lain bernilai nol.

2

Penyebut

Perkalian panjang kedua vektor. Membagi dengan panjang sama dengan menormalkan vektor menjadi panjang satuan.

3

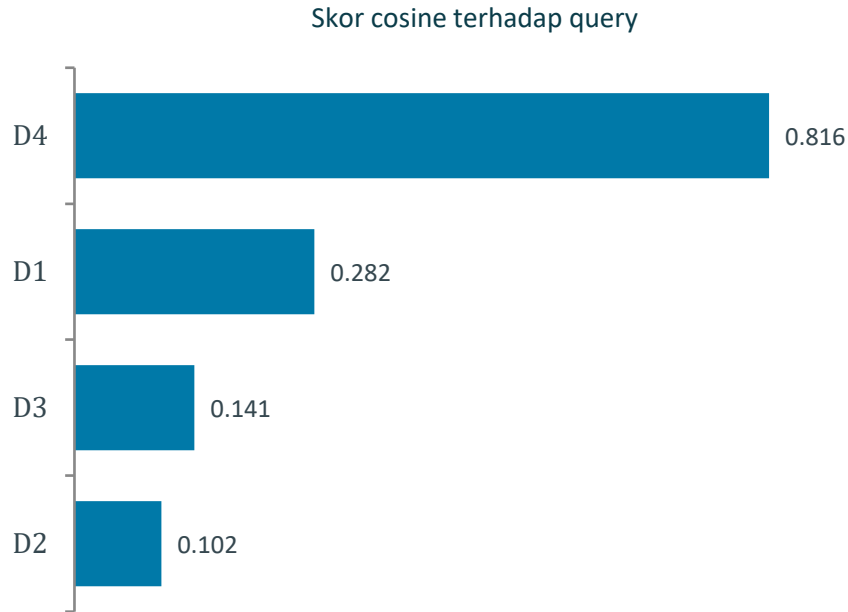
Akibatnya

Dokumen panjang tidak otomatis menang hanya karena memuat banyak kata. Yang dihitung adalah kemiripan proporsi.

Nilai 1 berarti arah identik; nilai 0 berarti tidak ada satu pun term yang sama.

Latihan: Merangking untuk “banjir surabaya”

Bobot query: 0,125 untuk kedua term, panjang query 0,177



Perhitungan

$$D4 \quad 0,125 \times 0,125 \times 2 \quad / \quad (0,177 \times 0,216)$$

$$D1 \quad 0,125 \times 0,125 \times 2 \quad / \quad (0,177 \times 0,627)$$

$$D3 \quad 0,125 \times 0,125 \quad / \quad (0,177 \times 0,627)$$

$$D2 \quad 0,125 \times 0,125 \quad / \quad (0,177 \times 0,870)$$

Bandingkan dengan pertemuan 3: Boolean AND hanya mengembalikan D1 dan D4 tanpa urutan. Kini keempat dokumen terurut, dan D4 jelas unggul.

Skema: $tf \text{ logaritmik} \times idf \text{ pada dokumen dan query, dinormalisasi cosine.}$

Apa yang Terjadi pada Peringkat Tadi?

Tiga pengamatan yang menjelaskan cara kerja VSM

1

D4 menang telak

D4 memuat kedua term query dan tidak memuat term lain yang berbobot besar. Arahnya paling searah dengan query.

2

D1 tertinggal jauh

D1 juga memuat kedua term, tetapi “landa” yang langka dan berbobot besar menarik arah vektornya menjauhi query.

3

D3 mengungguli D2

Keduanya memuat satu term query. D2 memuat dua term langka sehingga vektornya lebih panjang dan skornya lebih terencerkan.

Pertanyaan diskusi: apakah D1 layak berada jauh di bawah D4? Bagaimana jika D1 ternyata berita yang paling lengkap?

Normalisasi Panjang Dokumen

Cosine terlalu menguntungkan dokumen pendek, sedangkan pivoted normalization dapat mengoreksinya

Masalahnya

Singhal, Buckley & Mitra (1996) mengamati bahwa normalisasi cosine membuat dokumen pendek terambil lebih sering daripada peluangnya untuk relevan, sedangkan dokumen panjang, yang sering memang lebih lengkap, terambil terlalu jarang.

Gagasan koreksinya

Putar kurva normalisasi di sekitar sebuah titik poros: dokumen lebih pendek dari rata-rata dinormalkan sedikit lebih keras, dokumen lebih panjang sedikit lebih ringan.

$$\text{norm}' = (1 - s) \cdot \text{pivot} + s \cdot \text{norm}$$

s < 1 adalah kemiringan; pivot umumnya rata-rata norm di koleksi.

Jembatan ke pertemuan 5

Gagasan yang sama muncul lagi sebagai parameter b pada BM25. Di sana, derajat normalisasi panjang dokumen menjadi sebuah kenop yang dapat diatur untuk setiap koleksi.

Notasi SMART untuk Skema Pembobotan

Tiga huruf untuk dokumen, tiga huruf untuk query → ditulis ddd.qqq

Huruf 1: term frequency		Huruf 2: document frequency		Huruf 3: normalisasi	
n	tf apa adanya	n	tanpa idf	n	tanpa normalisasi
l	$1 + \log tf$	t	$\log (N / df)$	c	cosine
a	$0,5 + 0,5 \cdot tf / tf \text{ maks}$	p	idf probabilistik	u	pivoted unique
b	1 bila $tf > 0$			b	panjang byte

Inc.ltc

Dokumen: $\log tf$, tanpa idf, cosine.

Query: $\log tf$, idf, cosine.

Skema klasik yang dibahas Manning dkk. Bab 6.

Mengapa idf cukup di satu sisi?

Hasil kali titik mengalikan bobot query dan dokumen. idf yang dipasang di query sudah ikut terbawa ke setiap perkalian, dan bobot dokumen tidak perlu dihitung ulang ketika koleksi bertambah.

Contoh pada slide 12 memakai ltc.ltc, dengan terdapat idf di kedua sisi. Urutan hasil dapat berbeda antarskema; itulah alasan skema harus selalu dilaporkan.

Menghitung Skor Secara Efisien

Jangan hitung cosine terhadap seluruh koleksi, melainkan telusuri postings term query saja

```
skor = defaultdict(float)
for t in query:
    w_tq = bobot_query(t)
    for d, w_td in postings[t]:
        skor[d] += w_tq * w_td
for d in skor:
    skor[d] /= panjang[d]
return heapq.nlargest(k, skor.items(), key=lambda x: x[1])
```

Akumulator

Hanya dokumen yang memuat minimal satu term query yang mendapat skor.

Panjang disimpan

panjang[d] dihitung sekali saat indexing, bukan setiap query.

Heap untuk top-k

Tidak perlu mengurutkan seluruh dokumen, cukup menjaga k terbaik.

Pola ini disebut *term-at-a-time*. Alternatifnya, *document-at-a-time*, menelusuri semua postings secara serentak per docID, seperti algoritma intersect pertemuan 3.

Lebih Cepat Lagi: Memangkas Kandidat

Pengguna hanya melihat sepuluh hasil teratas, sedangkan skor persis untuk dokumen lain tidak diperlukan

1

Eliminasi index

Abaikan term query dengan idf rendah, atau utamakan dokumen yang memuat banyak term query sekaligus.

2

Champion list

Untuk setiap term, simpan terpisah r dokumen dengan bobot tertinggi. Query hanya dihitung pada gabungan daftar ini.

3

WAND (Weak AND)

Simpan batas atas skor setiap term. Dokumen yang skor maksimalnya tidak mungkin melampaui peringkat ke- k dilewati tanpa dihitung (Broder dkk., 2003).

4

Block-max WAND

Batas atas disimpan per blok postings sehingga lompatan jauh lebih agresif. Dipakai Lucene sejak versi 8, yang merupakan dasar Elasticsearch dan OpenSearch.

Dua teknik pertama tidak menjamin hasil persis: top- k bisa sedikit berbeda. WAND menjamin hasil top- k yang persis sama dengan perhitungan penuh.

Batas Kemampuan Vector Space Model

Kelemahan yang menjadi motivasi hampir seluruh materi paruh kedua semester

Sinonim

“banjir” dan “air bah” adalah sumbu yang berbeda; kemiripannya nol.

Mg 10–11

Polisemi

“bisa” sebagai kata kerja dan “bisa” ular berbagi satu sumbu.

Mg 12

Term dianggap independen

Setiap sumbu tegak lurus; hubungan antar-term diabaikan.

Mg 10

Urutan kata hilang

Bag of words tidak membedakan siapa mengalahkan siapa.

Mg 12

Bobot bersifat heuristik

Rumus TF-IDF lahir dari intuisi, bukan dari model peluang relevansi.

Mg 5

Meski demikian, TF-IDF tetap baseline yang wajib dilaporkan di hampir setiap penelitian IR hingga hari ini.

Praktikum: Mesin Pencari VSM Sederhana

Lanjutkan inverted index pertemuan 3; dikumpulkan melalui Ilmu2

1

Perluas postings

Setiap posting menyimpan (docID, tf); simpan df dan idf di dictionary

2

Hitung bobot & panjang

Bobot tf-idf logaritmik per posting, panjang vektor per dokumen

3

Fungsi cari(query, k)

Akumulator term-at-a-time, normalisasi cosine, top-k dengan heapq

4

Bandingkan dengan scikit-learn

TfidfVectorizer + cosine_similarity pada korpus yang sama

5

Jelaskan perbedaan angka

Periksa rumus idf dan tf bawaan TfidfVectorizer, lalu samakan konfigurasinya

6

Uji lima query

Bandingkan peringkat VSM dengan hasil Boolean pertemuan 3

```
from sklearn.feature_extraction.text import TfidfVectorizer      import heapq
```

Rangkuman & Persiapan Pertemuan Berikutnya

- TF mengukur seberapa penting term di satu dokumen; IDF mengukur seberapa langka term di seluruh koleksi.
- Dokumen dan query adalah vektor di ruang yang sama; kemiripannya diukur dengan sudut, bukan jarak.
- Normalisasi panjang menentukan siapa yang menang. Cosine cenderung menguntungkan dokumen pendek.
- Skor dihitung hanya lewat postings term query, dan top-k dapat dipangkas tanpa menghitung seluruh koleksi.

Persiapan Pertemuan 5

Baca Manning dkk. Bab 11. Tugas Mandiri 1 (bobot 10%) diberikan minggu depan: membandingkan TF-IDF dengan BM25. *Search engine* hari ini adalah baseline-nya.

Pustaka pertemuan ini

- [1] Manning, Raghavan & Schütze (2008), Bab 6, 7
- [2] Croft, Metzler & Strohman (2015), Bab 7
- Singhal, Buckley & Mitra (1996), SIGIR