

TUGAS BESAR

Pembacaan Dokumen Word Dan Pdf Melalui Proses *Stemming*, Pembobotan TF-IDF Dan *Similarity Measure* Dengan Algoritma *Euclidean Distance*

Diajukan untuk memenuhi tugas Data Mining dan Information Retrieval



Disusun oleh:

Maleakhi Ekklesia – 152019003

Nicolaus Buha Kristian Simarmata - 152019022

Agni Pangestu - 152019038

**PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL
BANDUNG**

2022

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Tuhan Yang Maha Esa atas rahmat dan berkat-Nya, penulis dapat menyelesaikan Tugas Besar mata kuliah Data Mining dan Information Retrieval tahun ajaran 2022/2023. Tujuan penyusunan laporan ini adalah guna untuk melengkapi Tugas Data Mining dan Information Retrieval ini berdasarkan materi-materi dan pembahasan saat perkuliahan yang telah dilalui dan dari sumber yang didapat.

Penulis menyadari masih banyak kekurangan dalam laporan ini. Hal ini dikarenakan terbatasnya pengalaman dan pengetahuan yang dimiliki dan didapatkan hingga saat ini. Oleh karena itu, saran dan masukan yang membangun selalu penulis harapkan untuk perbaikan tugas selanjutnya. Akhirnya, penulis berharap semoga Laporan Tugas Besar ini tentang “ Pembacaan Dokumen Word Dan Pdf Melalui Proses Stemming, Pembobotan TF-IDF Dan Similarity Measure Dengan Algoritma Euclidean Distance ” dapat bermanfaat bagi para pembaca.

Penulis

28 Desember 2022

DAFTAR ISI

DAFTAR ISI	ii
DAFTAR GAMBAR	iii
DAFTAR TABEL	iv
BAB I PENDAHULUAN	1
1.1 Latar Belakang	1
1.2 Rumusan Masalah	1
1.3 Tujuan	1
1.4 Ruang Lingkup	1
1.5 Manfaat	1
BAB II LANDASAN TEORI	2
2.1 Information Retrieval	2
2.2 Stemming	2
2.3 Sastrawi	2
2.4 Similarity Measure	3
2.5 Euclidean Distance	3
2.6 TF-IDF	3
BAB III PEMBAHASAN	4
3.1 Pembobotan nilai TF-IDF	4
3.2 <i>Euclidean Distance</i> dan pembacaan kata pada dokumen	11
BAB IV IMPLEMENTASI	13
BAB V PENUTUP	18
5.1 Kesimpulan	18
5.2 Saran	18
DAFTAR PUSTAKA	19

DAFTAR GAMBAR

Gambar 3. 1 Rumus normalisasi Term Frequency	6
Gambar 3. 2 Rumus Inverse Document Frequency (IDF)	8
Gambar 3. 3 Rumus TF-IDF	8
Gambar 3. 4 Rumus Euclidean Distance	11
Gambar 4. 1 Listing Code (1)	13
Gambar 4. 2 Listing Code (2)	13
Gambar 4. 3 Listing Code (3)	14
Gambar 4. 4 Listing Code(4)	14
Gambar 4. 5 Listing Code (5)	15
Gambar 4. 6 Listing Code (6)	15
Gambar 4. 7 Output Program (1)	16
Gambar 4. 8 Output Program (2)	16
Gambar 4. 9 Output Program (3)	17
Gambar 4. 10 Output Program (4)	17

DAFTAR TABEL

Tabel 3. 1 Teks kata dasar dari kumpulan 3 dokumen	4
Tabel 3. 2 Kata dasar dengan hasil TF awal	5
Tabel 3. 3 Kata dasar dengan hasil normalisasi TF	7
Tabel 3. 4 Kata dasar dengan hasil DF	7
Tabel 3. 5 Kata dasar dengan hasil IDF	8
Tabel 3. 6 Kata dasar dengan hasil TF-IDF	9
Tabel 3. 7 Hasil perhitungan TF-IDF keseluruhan	10
Tabel 3. 8 Nilai Pembobotan TF-IDF berdasarkan contoh kalimat	11

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pencarian informasi berupa dokumen teks atau yang biasa dikenal dengan istilah Information Retrieval (IR) merupakan proses pemisahan dokumen-dokumen yang dianggap relevan dari sekumpulan dokumen yang tersedia. Salah satu bagian penting dari Information Retrieval adalah proses stemming. Stemming adalah proses mereduksi kata berimbuhan menjadi kata dasar. Stemming sangat berguna untuk proses indexing maupun searching di dalam Information Retrieval. Ada pun pengecekan kemiripan teks dengan teks lainnya disebut juga similarity measure. Similarity measure adalah metode untuk menghitung kesamaan dua buah objek dan mengembalikan nilai kemiripan antara kedua objek tersebut berdasarkan pola atau karakteristik tertentu. Pengukuran jarak memegang peran penting dalam menentukan kemiripan atau keteraturan di antara data dan item. Euclidean distance merupakan salah satu metode untuk mengukur jarak dari dua objek tersebut. Kemudian menggunakan metode TF-IDF dimana metode ini digunakan untuk menentukan nilai frekuensi sebuah kata dalam sebuah dokumen atau artikel.

1.2 Rumusan Masalah

- Bagaimana proses pencarian informasi dari dokumen teks agar mendapatkan informasi yang sesuai dan relevan?
- Bagaimana cara mengetahui fungsi dari pembacaan dokumen teks pdf dan word?

1.3 Tujuan

- Mengetahui proses pencarian informasi dari dokumen teks agar mendapatkan informasi yang sesuai dan relevan.
- Mengetahui fungsi dari pembacaan dokumen teks pdf dan word.

1.4 Ruang Lingkup

Dalam pembacaan word dan pdf ini dilakukan dengan menggunakan proses *Stemming*, Pembobotan TF-IDF Dan *Similarity Measure* Dengan Algoritma *Euclidean Distance*.

1.5 Manfaat

Manfaatnya untuk menambah wawasan dalam proses pembacaan word dan pdf menggunakan proses yang ada pada Information Retrieval.

BAB II

LANDASAN TEORI

2.1 Information Retrieval

Information Retrieval adalah ilmu pencarian informasi dari sejumlah data yang sudah hilang karena terlalu banyaknya data yang ada. Ilmu ini dipopulerkan oleh Vannervar Bush pada tahun 1945 dan implementasinya mulai dikenalkan pada tahun 1950-an. Pada tahun 1990-an, sudah banyak teknik dan metode dari *information retrieval* yang dikembangkan dan digunakan. Tujuan dari sistem IR adalah untuk memenuhi kebutuhan informasi pengguna dengan me-*rerieve* semua dokumen yang mungkin relevan, pada waktu yang sama me-*retrieve* sesedikit mungkin dokumen yang tidak relevan.

2.2 Stemming

Stemming merupakan suatu proses yang terdapat dalam sistem IR yang mentransformasi kata-kata yang terdapat dalam suatu dokumen ke kata – kata akarnya (*root word*) dengan menggunakan aturan-aturan tertentu. Sebagai contoh, kata bersama, kebersamaan, menyamai, akan distem ke root wordnya yaitu “sama”. Proses *stemming* dilakukan dengan menghilangkan semua imbuhan (*affixes*) baik yang terdiri dari awalan (*preffixes*) sisipan (*infixes*) maupun akhiran (*suffixes*), stemming dilakukan atas dasar asumsi bahwa kata-kata yang memiliki stem yang sama memiliki makna dasar yang sama. Teknik *stemming* dapat dikategorikan menjadi tiga, yaitu berdasarkan aturan dalam bahasa tertentu, berdasarkan kamus, dan berdasarkan kemunculan bersama. Salah satu tujuan utama dilakukan proses stemming adalah meningkatkan efisiensi dengan cara memilah isi dokumen menjadi unit-unit kecil yang akan menjadi penciri misalnya berupa kata, frase atau kalimat.

2.3 Sastrawi

Sastrawi merupakan *library* NLP yang dikhususkan untuk bahasa indonesia yang dibangun dengan algoritma NA (Nazief dan Adriani). Algoritma NA merupakan aturan yang mengikuti pada aturan bahasa indonesia, adapun aturannya adalah penentuan imbuhan yang diperbolehkan atau tidak. Awal mulanya sastrawi dikembangkan dan diperuntukkan untuk bahasa pemrograman PHP, akan tetapi karena popularitasnya selanjutnya library ini dikembangkan juga agar dapat mendukung bahasa pemrograman Python.

2.4 Similarity Measure

Similarity measure adalah metode yang digunakan untuk menghitung kesamaan dari dua buah objek berdasarkan pola atau karakteristik tertentu. Tahapan menentukan atau mendeskripsikan nilai kuantitatif dari tingkat kemiripan atau ketidakmiripan data memiliki peranan sangat penting. Ukuran kemiripan teks menggambarkan tingkat kemiripan antara satu teks dengan teks lainnya. Teks dapat terdiri dari beberapa kata, namun dapat pula terdiri dari milyaran kata yang tertulis dalam sebuah naskah.

2.5 Euclidean Distance

Euclidean distance merupakan salah satu metode perhitungan jarak yang digunakan untuk mengukur jarak dari 2 (dua) buah titik dalam *Euclidean space* (meliputi bidang *euclidean* dua dimensi, tiga dimensi, atau bahkan lebih). *Euclidean distance* mengidentifikasi seberapa jauh dua vektor terpisah satu sama lain. Artinya dia melihat jarak kedekatan antara dua teks.

2.6 TF-IDF

TF-IDF (Term Frequency Inverse Document Frequency) merupakan metode yang digunakan untuk menentukan nilai frekuensi sebuah kata di dalam sebuah dokumen atau artikel dan juga frekuensi di dalam banyak dokumen. Perhitungan ini menentukan seberapa relevan sebuah kata di dalam sebuah dokumen. Pada dasarnya, TF-IDF bekerja dalam menentukan frekuensi relatif suatu kata kemudian dibandingkan dengan proporsi kata tersebut pada seluruh dokumen. Algoritma TF-IDF melakukan pemberian bobot pada setiap kata kunci disetiap kategori untuk mencari kemiripan kata kunci dengan kategori yang tersedia.

BAB III

PEMBAHASAN

3.1 Pembobotan nilai TF-IDF

Data yang digunakan pada studi kasus yang diperlihatkan pada **Tabel 3.1** merupakan data contoh berupa teks kata dasar tentang sepeda yang diambil dari tiga dokumen dalam format word dan pdf. Dari tiga dokumen teks tersebut 58 kata dasar yang akan diproses untuk mendapatkan hasil akhir yaitu nilai TF-IDF.

Tabel 3. 1 Teks kata dasar dari kumpulan 3 dokumen

No	Text	No	Text	No	Text
1	alat	21	hindia	41	panjang
2	an	22	inggris	42	pedal
3	bagai	23	jengki	43	powerbike
4	bahasa	24	kasar	44	pria
5	baik	25	kemudian	45	punya
6	bantu	26	kenal	46	rancang
7	banyak	27	kendara	47	sama
8	beberapa	28	kerja	48	sepeda
9	beda	29	kompak	49	tahan
10	belanda	30	lebih	50	tahun
11	berat	31	listrik	51	tetapi
12	bike	32	maupun	52	tidak
13	daya	33	meda	53	tinggi
14	dengan	34	medan	54	tingkat
15	desain	35	milik	55	ukur
16	fitur	36	motor	56	umum
17	gerak	37	mtb	57	wanita
18	geser	38	mulai	58	zaman
19	guna	39	onthe		
20	gunung	40	pada		

Selanjutnya adalah mencari *Term Frequency* (TF) awal yaitu banyaknya atau frekuensi kemunculan kata pada suatu dokumen. Berikut ini hasil TF dari tiap kata dalam tiga dokumen yang akan diperlihatkan pada **Tabel 3.2**.

Tabel 3. 2 Kata dasar dengan hasil TF awal

No	Text	TF		
		R1	R2	R3
1	alat	0	1	0
2	an	0	0	1
3	bagai	0	1	0
4	bahasa	1	0	0
5	baik	0	0	1
6	bantu	0	1	0
7	banyak	0	0	1
8	beberapa	1	0	0
9	beda	0	1	1
10	belanda	0	0	1
11	berat	1	0	0
12	bike	0	1	0
13	daya	1	0	0
14	dengan	0	1	0
15	desain	0	0	1
16	fitur	1	0	0
17	gerak	0	1	0
18	geser	0	0	1
19	guna	1	0	0
20	gunung	2	0	0
21	hindia	0	0	1
22	inggris	1	0	0
23	jengki	0	0	1
24	kasar	1	0	0
25	kemudian	0	0	1
26	kenal	0	1	0
27	kendara	0	0	1
28	kerja	1	0	0
29	kompak	0	0	1
30	lebih	0	0	1
31	listrik	0	5	0
32	maupun	0	0	1
33	meda	1	0	0
34	medan	1	0	0
35	milik	1	0	0
36	motor	0	2	0
37	mtb	1	0	0
38	mulai	0	0	2
39	onthe	0	0	2
40	pada	0	0	1
41	panjang	0	0	1
42	pedal	0	1	0
43	powerbike	0	1	0
44	pria	0	0	1
45	punya	0	2	0
46	rancang	1	0	0
47	sama	1	0	0
48	sepeda	4	6	3
49	tahan	1	0	0
50	tahun	0	0	1
51	tetapi	1	0	0
52	tidak	0	0	1
53	tinggi	0	0	1
54	tingkat	1	0	0
55	ukur	0	0	2
56	umum	0	1	0
57	wanita	0	0	1
58	zaman	0	0	1
JUMLAH		24	26	31

Terlihat dari **Tabel 3.2** setelah frekuensi kemunculan kata didapat dari masing-masing dokumen, maka akan mendapatkan jumlah total dari kemunculan kata keseluruhan dari ketiga dokumen tersebut, yaitu: dokumen 1 memiliki total 24 kata, dokumen 2 memiliki total 26 kata dan dokumen 3 memiliki total 31 kata. Tahapan selanjutnya adalah mencari nilai normalisasi dari *Term Frequency* yang telah didapatkan sebelumnya dengan menggunakan rumus sebagai berikut.

$$tf_{t,d} = \frac{n_{t,d}}{\text{Total number of terms in document}}$$

Gambar 3. 1 Rumus normalisasi Term Frequency

Dengan keterangan:

- tf = frekuensi kemunculan kata pada sebuah dokumen
- n = total kata pada suatu dokumen

Sehingga untuk perhitungan normalisasi nilai TF nya adalah:

1. Untuk TF1

- $tf(\text{alat}) = \frac{0}{24} = 0$
- $tf(\text{an}) = \frac{0}{24} = 0$
- $tf(\text{bagai}) = \frac{0}{24} = 0$
- $tf(\text{bahasa}) = \frac{1}{24} = 0.0416$
- $tf(\text{baik}) = \frac{0}{24} = 0$

2. Untuk TF2

- $tf(\text{alat}) = \frac{1}{26} = 0.0385$
- $tf(\text{an}) = \frac{0}{26} = 0$
- $tf(\text{bagai}) = \frac{1}{26} = 0.0385$
- $tf(\text{bahasa}) = \frac{0}{26} = 0$
- $tf(\text{baik}) = \frac{0}{26} = 0$

3. Untuk TF3

- $tf(\text{alat}) = \frac{0}{31} = 0$
- $tf(\text{an}) = \frac{1}{31} = 0.0323$

- $tf(\text{bagai}) = \frac{0}{31} = 0.0385$
- $tf(\text{bahasa}) = \frac{0}{31} = 0$
- $tf(\text{baik}) = \frac{1}{31} = 0.0323$

Maka hasil perhitungan normalisasi nilai *Term Frequency* (TF) dari tiga dokumen untuk lebih jelasnya dapat dilihat pada **Tabel 3.3** berikut.

Tabel 3. 3 Kata dasar dengan hasil normalisasi TF

No	Text	TF			TF - NORMALIZATION		
		R1	R2	R3	TF1	TF2	TF3
1	alat	0	1	0	0	0.0385	0
2	an	0	0	1	0	0	0.0323
3	bagai	0	1	0	0	0.0385	0
4	bahasa	1	0	0	0.0417	0	0
5	baik	0	0	1	0	0	0.0323

Tahapan selanjutnya adalah mencari nilai *Document Frequency* (DF), yaitu untuk mengetahui banyaknya dokumen yang mengandung kata ke-i. Seperti contohnya pada data ke 1 yaitu kata ‘alat’, kemunculan kata pada dokumen satu = 0, dokumen dua = 1, dan dokumen tiga = 0, maka nilai dari DF nya adalah 1 karena kemunculan kata ‘alat’ hanya ada pada dokumen dua. Untuk hasilnya akan diperlihatkan pada **Tabel 3.4** berikut.

Tabel 3. 4 Kata dasar dengan hasil DF

No	Text	TF			TF - NORMALIZATION			DF
		R1	R2	R3	TF1	TF2	TF3	
1	alat	0	1	0	0	0.0385	0	1
2	an	0	0	1	0	0	0.0323	1
3	bagai	0	1	0	0	0.0385	0	1
4	bahasa	1	0	0	0.0417	0	0	1
5	baik	0	0	1	0	0	0.0323	1

Setelah mendapatkan nilai DF, maka tahapan selanjutnya adalah menghitung nilai *Inverse Document Frequency* (IDF) yang merupakan nilai untuk mengukur seberapa penting sebuah kata. IDF akan menilai kata yang sering muncul sebagai kata yang kurang penting berdasarkan kemunculan kata tersebut pada seluruh dokumen. Sehingga semakin kecil nilai IDF maka akan dianggap semakin tidak penting kata tersebut, begitu pula sebaliknya. Rumus yang digunakan

untuk mencari nilai IDF adalah sebagai berikut.

$$idf(t) = \log \frac{n}{df(t)} + 1$$

Gambar 3. 2 Rumus Inverse Document Frequency (IDF)

Dengan keterangan:

- n = total dokumen
- df = banyaknya dokumen yang mengandung kata ke- i

Sehingga perhitungan untuk mendapatkan nilai IDF nya adalah:

- $idf(alat) = \log \frac{3}{1} + 1 = 1.4771$
- $idf(an) = \log \frac{3}{1} + 1 = 1.4771$
- $idf(bagai) = \log \frac{3}{1} + 1 = 1.4771$
- $idf(bahasa) = \log \frac{3}{1} + 1 = 1.4771$
- $idf(baik) = \log \frac{3}{1} + 1 = 1.4771$

Maka hasil perhitungan nilai *Inverse Document Frequency* (IDF) dari tiga dokumen untuk lebih jelasnya dapat dilihat pada Tabel 3.5 berikut.

Tabel 3. 5 Kata dasar dengan hasil IDF

No	Text	TF			TF - NORMALIZATION			DF	IDF
		R1	R2	R3	TF1	TF2	TF3		
1	alat	0	1	0	0	0.0385	0	1	1.47712
2	an	0	0	1	0	0	0.0323	1	1.47712
3	bagai	0	1	0	0	0.0385	0	1	1.47712
4	bahasa	1	0	0	0.0417	0	0	1	1.47712
5	baik	0	0	1	0	0	0.0323	1	1.47712

Setelah mendapatkan nilai TF dan IDF dari perhitungan sebelumnya, maka tahap terakhir untuk mendapatkan nilai pembobotan adalah dengan mencari nilai TF-IDF yang merupakan hasil dari perkalian TF dan IDF dengan rumus yang digunakan adalah sebagai berikut.

$$tfidf_{t,d} = tf_{t,d} \times idf_d$$

Gambar 3. 3 Rumus TF-IDF

Sehingga perhitungan untuk mendapatkan nilai TF-IDF nya adalah:

1. Untuk TF-IDF1

- $tfidf(alat) = 0 \times 1.4771 = 0$
- $tfidf(an) = 0 \times 1.4771 = 0$
- $tfidf(bagai) = 0 \times 1.4771 = 0$
- $tfidf(bahasa) = 0.0417 \times 1.4771 = 0.0615$
- $tfidf(baik) = 0 \times 1.4771 = 0$

2. Untuk TF-IDF2

- $tfidf(alat) = 0.0385 \times 1.4771 = 0.0568$
- $tfidf(an) = 0 \times 1.4771 = 0$
- $tfidf(bagai) = 0.0385 \times 1.4771 = 0.0568$
- $tfidf(bahasa) = 0 \times 1.4771 = 0$
- $tfidf(baik) = 0 \times 1.4771 = 0$

3. Untuk TF-IDF3

- $tfidf(alat) = 0 \times 1.4771 = 0$
- $tfidf(an) = 0.0323 \times 1.4771 = 0.0476$
- $tfidf(bagai) = 0 \times 1.4771 = 0$
- $tfidf(bahasa) = 0 \times 1.4771 = 0$
- $tfidf(baik) = 0.0323 \times 1.4771 = 0.0476$

Maka hasil perhitungan nilai TF-IDF dari tiga dokumen untuk lebih singkat dan jelasnya dapat dilihat pada **Tabel 3.6** berikut.

Tabel 3. 6 Kata dasar dengan hasil TF-IDF

No	Text	TF			TF - NORMALIZATION			DF	IDF	TF-IDF1	TF-IDF2	TF-IDF3
		R1	R2	R3	TF1	TF2	TF3					
1	alat	0	1	0	0	0.0385	0	1	1.47712	0	0.05681	0
2	an	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
3	bagai	0	1	0	0	0.0385	0	1	1.47712	0	0.05681	0
4	bahasa	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
5	baik	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765

Sehingga hasil perhitungan TF-IDF dari total keseluruhan 58 kata dasar yang didapat dari tiga dokumen dapat dilihat pada **Tabel 3.7** berikut.

Tabel 3. 7 Hasil perhitungan TF-IDF keseluruhan

No	Text	TF			TF - NORMALIZATION			DF	IDF	TF-IDF1	TF-IDF2	TF-IDF3
		R1	R2	R3	TF1	TF2	TF3					
1	alat	0	1	0	0	0.0385	0	1	1.47712	0	0.05681	0
2	an	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
3	bagai	0	1	0	0	0.0385	0	1	1.47712	0	0.05681	0
4	bahasa	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
5	baik	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
6	bantu	0	1	0	0	0.0385	0	1	1.47712	0	0.05681	0
7	banyak	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
8	beberapa	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
9	beda	0	1	1	0	0.0385	0.0323	2	1.17609	0	0.04523	0.03794
10	belanda	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
11	berat	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
12	bike	0	1	0	0	0.0385	0	1	1.47712	0	0.05681	0
13	daya	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
14	dengan	0	1	0	0	0.0385	0	1	1.47712	0	0.05681	0
15	desain	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
16	fitur	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
17	gerak	0	1	0	0	0.0385	0	1	1.47712	0	0.05681	0
18	geser	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
19	guna	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
20	gunung	2	0	0	0.0833	0	0	1	1.47712	0.12309	0	0
21	hindia	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
22	inggris	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
23	jengki	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
24	kasar	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
25	kemudian	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
26	kenal	0	1	0	0	0.0385	0	1	1.47712	0	0.05681	0
27	kendara	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
28	kerja	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
29	kompak	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
30	lebih	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
31	listrik	0	5	0	0	0.1923	0	1	1.47712	0	0.28406	0
32	maupun	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
33	meda	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
34	medan	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
35	milik	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
36	motor	0	2	0	0	0.0769	0	1	1.47712	0	0.11362	0
37	mtb	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
38	mulai	0	0	2	0	0	0.0645	1	1.47712	0	0	0.0953
39	onthel	0	0	2	0	0	0.0645	1	1.47712	0	0	0.0953
40	pada	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
41	panjang	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
42	pedal	0	1	0	0	0.0385	0	1	1.47712	0	0.05681	0
43	powerbike	0	1	0	0	0.0385	0	1	1.47712	0	0.05681	0
44	pria	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
45	punya	0	2	0	0	0.0769	0	1	1.47712	0	0.11362	0
46	rancang	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
47	sama	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
48	sepeda	4	6	3	0.1667	0.2308	0.0968	3	1	0.16667	0.23077	0.09677
49	tahan	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
50	tahun	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
51	tetapi	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
52	tidak	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
53	tinggi	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
54	tingkat	1	0	0	0.0417	0	0	1	1.47712	0.06155	0	0
55	ukur	0	0	2	0	0	0.0645	1	1.47712	0	0	0.0953
56	umum	0	1	0	0	0.0385	0	1	1.47712	0	0.05681	0
57	wanita	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
58	zaman	0	0	1	0	0	0.0323	1	1.47712	0	0	0.04765
JUMLAH		24	26	31								

3.2 Euclidean Distance dan pembacaan kata pada dokumen

Hasil dari pembobotan nilai TF-IDF sebelumnya akan digunakan untuk perhitungan mencari *Euclidean Distance* dari kata yang di *input* terhadap tiga dokumen yang digunakan. Sebagai contoh kata yang akan diproses adalah kata dari kalimat '**saya punya sepeda yang bergerak dengan pedal**'. Dari kalimat tersebut, kata dasar yang dapat diambil untuk diproses adalah kata: 'punya', 'sepeda', 'gerak', 'pedal'. Untuk nilai TF-IDF dari kata-kata tersebut dapat dilihat pada **Tabel 3.7** berikut ini.

Tabel 3. 8 Nilai Pembobotan TF-IDF berdasarkan contoh kalimat

No	Text	TF			TF - NORMALIZATION			DF	IDF	TF-IDF1	TF-IDF2	TF-IDF3
		R1	R2	R3	TF1	TF2	TF3					
1	punya	0	2	0	0	0.0769	0	1	1.47712	0	0.1136	0
2	sepeda	4	6	3	0.1667	0.2308	0.0968	3	1	0.1667	0.2308	0.0968
3	gerak	0	1	0	0	0.0385	0	1	1.47712	0	0.0568	0
4	pedal	0	1	0	0	0.0385	0	1	1.47712	0	0.0568	0

Terlihat pada tabel 3.8 terdapat 4 kata yang memiliki bobot nilai TF-IDF dari dokumen 1, 2 dan 3 adalah berikut:

1. Dokumen 1 = (0, 0.1667, 0, 0)
2. Dokumen 2 = (0.1136, 0.2308, 0.0568, 0.0568)
3. Dokumen 3 = (0, 0.0968, 0, 0)

Sehingga tahapan selanjutnya adalah mencari nilai *Euclidean Distance* dari ketiga dokumen tersebut dengan rumus sebagai berikut.

$$d_{Euclidean}(x, y) = \sqrt{\sum_i^n (x_i - y_i)^2}$$

Gambar 3. 4 Rumus Euclidean Distance

Dengan keterangan:

- d = jarak antara x dan y
- x = data pusat kluster
- y = data pada atribut
- i = setiap data

- n = jumlah data
- x_i = data pada pusat kluster ke i
- y_i = data pada setiap data ke i

Perhitungan nilai *Euclidean Distance* dari ketiga dokumen tersebut adalah:

1. Dokumen 1 = (0, 0.1667, 0, 0) dengan *query* : punya=1, sepeda=1, gerak=1, pedal=1.

$$\begin{aligned}
 d_{dokumen1} &= ((1 - 0)^2 + (1 - 0.1667)^2 + (1 - 0)^2 + (1 - 0)^2)^{\frac{1}{2}} \\
 &= (1 + (0.8333)^2 + 1 + 1)^{\frac{1}{2}} \\
 &= (1 + 0.6943 + 1 + 1)^{\frac{1}{2}} = \mathbf{1.922}
 \end{aligned}$$

2. Dokumen 2 = (0.1136, 0.2308, 0.0568, 0.0568) dengan *query* : punya=1, sepeda=1, gerak=1, pedal=1.

$$\begin{aligned}
 d_{dokumen2} &= ((1 - 0.1136)^2 + (1 - 0.2308)^2 + (1 - 0.0568)^2 + (1 - 0.0568)^2)^{\frac{1}{2}} \\
 &= ((0.8864)^2 + (0.7692)^2 + (0.9432)^2 + (0.9432)^2)^{\frac{1}{2}} \\
 &= (0.7857 + 0.5917 + 0.8896 + 0.8896)^{\frac{1}{2}} = \mathbf{1.777}
 \end{aligned}$$

3. Dokumen 3 = (0, 0.0968, 0, 0) dengan *query* : punya=1, sepeda=1, gerak=1, pedal=1.

$$\begin{aligned}
 d_{dokumen3} &= ((1 - 0)^2 + (1 - 0.0968)^2 + (1 - 0)^2 + (1 - 0)^2)^{\frac{1}{2}} \\
 &= (1 + (0.9032)^2 + 1 + 1)^{\frac{1}{2}} \\
 &= (1 + 0.8158 + 1 + 1)^{\frac{1}{2}} = \mathbf{1.953}
 \end{aligned}$$

Dari hasil perhitungan diatas didapatkan hasil nilai *Euclidean Distance* dari tiga dokumen adalah sebagai berikut:

- Nilai *euclidean distance* pada dokumen 1 adalah 1.922
- Nilai *euclidean distance* pada dokumen 2 adalah 1.777
- Nilai *euclidean distance* pada dokumen 3 adalah 1.953

Karena nilai $1.922 < \mathbf{1.777} < 1.953$, maka dapat disimpulkan bahwa kalimat '**saya punya sepeda yang bergerak dengan pedal**' terdapat pada **dokumen 2**.

BAB IV

IMPLEMENTASI

Pada pembangunan sistem yang dapat membaca sebuah *document* dan juga *file* dengan ekstensi PDF didalam *path* atau folder yang sama ini, dibangun menggunakan bahasa pemrograman Python. Implementasi sistem yang dilakukan ini menggunakan beberapa *library* didalamnya seperti *Streamlit*, *Pandas*, *docx2txt*, *numpy*, *pdfminer*, dan *Sastrawi*. Perancangan sistem ini adalah sebagai berikut :

```
def uploadfiles():
    uploaded_file = st.text_input(label="_"_"Please enter the path to the file")
    try:
        global inputuser_path_query
        path = uploaded_file
        os.chdir(uploaded_file)
        files = os.listdir()
        st.write(files)
        inputuser = st.text_input(label="Input Query")
        query = inputuser
        for file_name in files:
            if uploaded_file is not None:
                global datauji
                try:
                    text = extract_text(file_name)
                    text = re.sub("[a-zA-Z]", " ", text)
                except:
                    text = docx2txt.process(file_name)
                    text = re.sub("[a-zA-Z]", " ", text)
                datauji.append(text)
        del datauji[0]
        st.write("==Hasil Ekstraksi Data Uji==")
        st.write(datauji)
    except:
```

Gambar 4. 1 Listing Code (1)

Telah diperlihatkan pada **Gambar 4.1** dimana implementasi sistem ini dilakukan dengan penggunaan *function*, hal ini yang dilakukan adalah untuk memberikan dan menerima *path* yang diinput oleh *user* untuk memberikan informasi pada *file – file* yang ada pada *path* tersebut dengan menggunakan *for*.

```
proses stemming text
def stemming(datauji):
    for i in datauji:
        text = "".join(i)
        text = text.lower()
        # Menghilangkan komponen selain angka dan huruf
        text = re.sub("[a-zA-Z]", " ", text)
        text = text.translate(str.maketrans("", "", string.punctuation))
        text = text.strip()
        text = re.sub('[\s+]', ' ', text)

        # Tokenizing dan lemmatizing
        text = nltk.word_tokenize(text)
        # proses unarray data
        text = " ".join(text)

        # Proses Stemming Nazief dan Adriani
        text = stemmer.stem(text)

        # Melakukan filtering (menghapus stopwords)
        text = stopwords.remove(text)
        text = nltk.tokenize.word_tokenize(text)

    global hasilstem
    global hasilstemsemua
    # output data
    hasilstemsemua = hasilstemsemua+text
    hasilstem.append(text)
    del hasilstem[0]
    st.write("==Hasil Stemming (Nazief dan Adriani) Semua File==")
    st.write(hasilstem)
```

Gambar 4. 2 Listing Code (2)

Lalu pada **Gambar 4.2** setelah dilakukan proses *input path* dan juga melakukan ekstrasi *file* yang dilakukan, maka dilakukan proses *stemming* pada seluruh *file* yang sudah diekstrasi untuk menghilangkan *stopword* menggunakan Algoritma Nazief dan Adriani. Lalu selanjutnya adalah melakukan proses dari menghitung jumlah kemunculan kata – kata yang telah di *stemming* dengan melakukan perulangan *for*, dan juga *set()* agar tidak terdapat redundansi kata yang muncul. Proses tersebut ditampilkan pada **Gambar 4.3**.

```
def countstring(hasilstem_path):
    datasama = {}
    kata = []
    jumlah=[]
    global hasilhitung
    try:
        os.chdir(path)
        files = os.listdir()
        b = -1
        for a in hasilstem_:
            b = b+1
            for i in set(a):
                datasama[i] = a.count(i)
                kata.append(i)
                jumlah.append(datasama[i])

            hasil_data = {
                'Jumlah': jumlah,
                'Kata': kata
            }
            write = pd.DataFrame(hasil_data)
            kata.clear()
            jumlah.clear()
            datasama.clear()
            tab1, tab2 = st.tabs(["Data", "Grafik"])
            with tab1:
                col1, col2, col3, col4, col5 = st.columns(5)
                with col1:
                    st.write('')
                with col2:
                    st.write('')
                with col3:
                    st.write('Hasil Stemming File - **' + files[b] + '**')
                    st.write(write)
                with col4:
                    st.write('')
                with col5:
                    st.write('')
            with tab2:
                st.bar_chart(write_x="Kata", y="Jumlah")
    except:
        st.text('')
```

Gambar 4. 3 Listing Code (3)

Setelah mendapatkan jumlah kemunculan kata – kata yang muncul pada hasil *stemming* ini, langkah selanjutnya adalah melakukan proses pembobotan TFIDF dengan melakukan pada hasil *array* atau kemunculan kata – kata tersebut yang disimpan pada *array*, dan hasil proses ini dilakukan pada **Gambar 4.4**.

```
def tfidf(hasilstem2, hasilstemuser, jumlahuser):
    global hasiltfidf
    dump=[]
    try:
        tfidfvectorizer = TfidfVectorizer()
        tfidf_wm = tfidfvectorizer.fit_transform(hasilstem2)
        tfidf_tokens = tfidfvectorizer.get_feature_names_out()
        df_tfidfvect = pd.DataFrame(data=tfidf_wm.toarray(), columns=tfidf_tokens)
        st.write('Pembobotan TF - IDF')
        st.write(df_tfidfvect)
        try:
            for a in range(len(df_tfidfvect.index)):
                for i in hasilstemuser:
                    dump.append(df_tfidfvect.iloc[a][i])
                for i in range(0, len(dump), len(jumlahuser)):
                    hasiltfidf.append(dump[i:i + len(jumlahuser)])
        except:
            hasiltfidf = []
    except:
        hasiltfidf = []
```

Gambar 4. 4 Listing Code(4)

Setelah melakukan proses dari pembobotan *term* melalui TFIDF, yang perlu dilakukan adalah menghitung jarak *Euclidean distance* untuk melihat dan mencari *file* mana yang merujuk pada *query* yang diberikan. Proses tersebut dilakukan pada **Gambar 5.5**.

```
def euclidean(hasiltfidf, jumlahuser, path):
    try:
        count = 0
        for i in os.listdir(path):
            # check if current path is a file
            if os.path.isfile(os.path.join(path, i)):
                count += 1
        global jarak
        try:
            for i in hasiltfidf:
                hasil = math.dist(jumlahuser, i)
                jarak.append(hasil)
        except:
            st.text('')
            jarak = []
    except:
        st.text('')
```

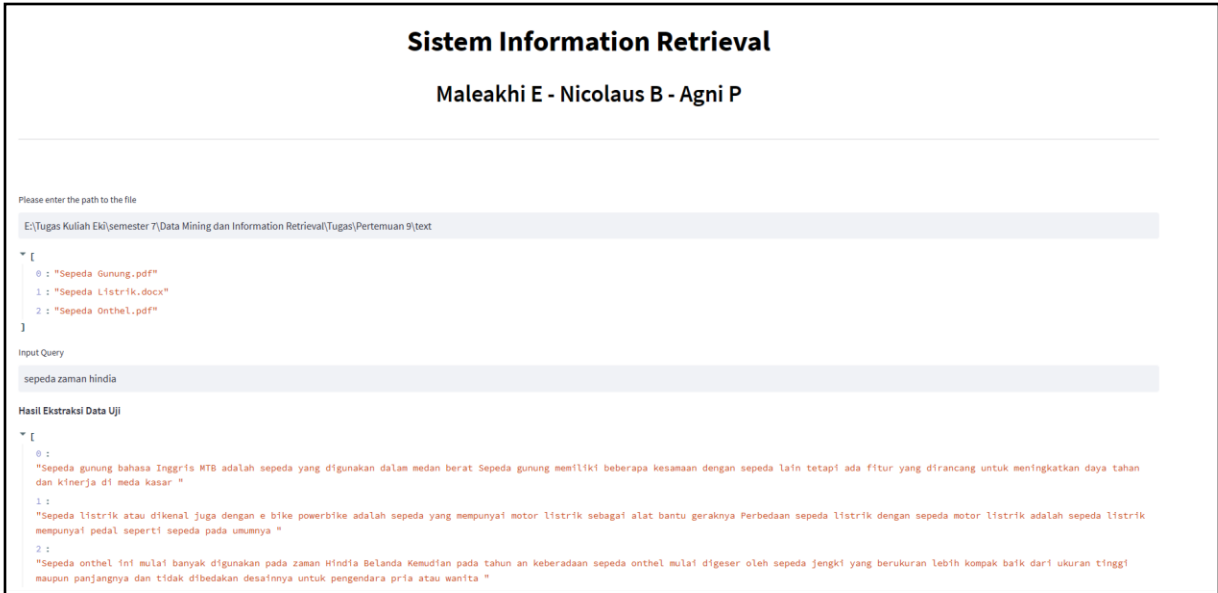
Gambar 4. 5 Listing Code (5)

Jika telah mendapatkan nilai dari *Euclidean distance* yang dicari melalui nilai pembobotan tersebut maka yang perlu dilakukan terakhir adalah mencari nilai jarak terkecil yang dijadikan *index* pada suatu *file* pada *path* yang diberikan.

```
def cekfile(jarak, path, query):
    try:
        count = 0
        for i in os.listdir(path):
            # check if current path is a file
            if os.path.isfile(os.path.join(path, i)):
                count += 1
        if not jarak:
            with st.expander("**Rujukan File Dari Hasil Euclidean Distance**"):
                st.write("**Tidak Ada File Yang Menyerupai Query**")
        else:
            terkecil = jarak.index(min(jarak))
            os.chdir(path)
            files = os.listdir()
            with st.expander("**Rujukan File Dari Hasil Euclidean Distance**"):
                st.write("Mungkin dengan query yang dimasukkan "
                        "adalah **"+str(query)+"** , File yang dimaksud "
                        "adalah **"+str(files[terkecil])+"** + " " Dengan Jumlah Euclidean Distance : **"+str(min(jarak))+"**")
                st.write("**(Total Perhitungan Euclidean Distance Lainnya : "
                        "+ str(jarak) + ")**")
    except:
        st.text('')
```

Gambar 4. 6 Listing Code (6)

Sehingga dari implementasi yang dilakukan dan dirancang pada pemograman Python ini, berikut merupakan hasil dari *output* sistem yang telah dibangun,



```
Sistem Information Retrieval
Maleakhi E - Nicolaus B - Agni P

Please enter the path to the file
E:\Tugas Kuliah Eki\semester 7\Data Mining dan Information Retrieval\Tugas\Pertemuan 9\text

[
  0 : "Sepeda Gunung.pdf"
  1 : "Sepeda Listrik.docx"
  2 : "Sepeda Onthel.pdf"
]

Input Query
sepeda zaman hindia

Hasil Ekstraksi Data Uji
[
  0 :
    "Sepeda gunung bahasa Inggris MTB adalah sepeda yang digunakan dalam medan berat Sepeda gunung memiliki beberapa kesamaan dengan sepeda lain tetapi ada fitur yang dirancang untuk meningkatkan daya tahan dan kinerja di meda kasar "
  1 :
    "Sepeda listrik atau dikenal juga dengan e bike powerbike adalah sepeda yang mempunyai motor listrik sebagai alat bantu geraknya Perbedaan sepeda listrik dengan sepeda motor listrik adalah sepeda listrik mempunyai pedal seperti sepeda pada umumnya "
  2 :
    "Sepeda onthel ini mulai banyak digunakan pada zaman Hindia Belanda Kemudian pada tahun an keberadaan sepeda onthel mulai digeser oleh sepeda jengki yang berukuran lebih kompak baik dari ukuran ttinggi maupun panjangnya dan tidak dbedakan desainnya untuk pengendara pria atau wanita "
```

Gambar 4. 7 Output Program (1)

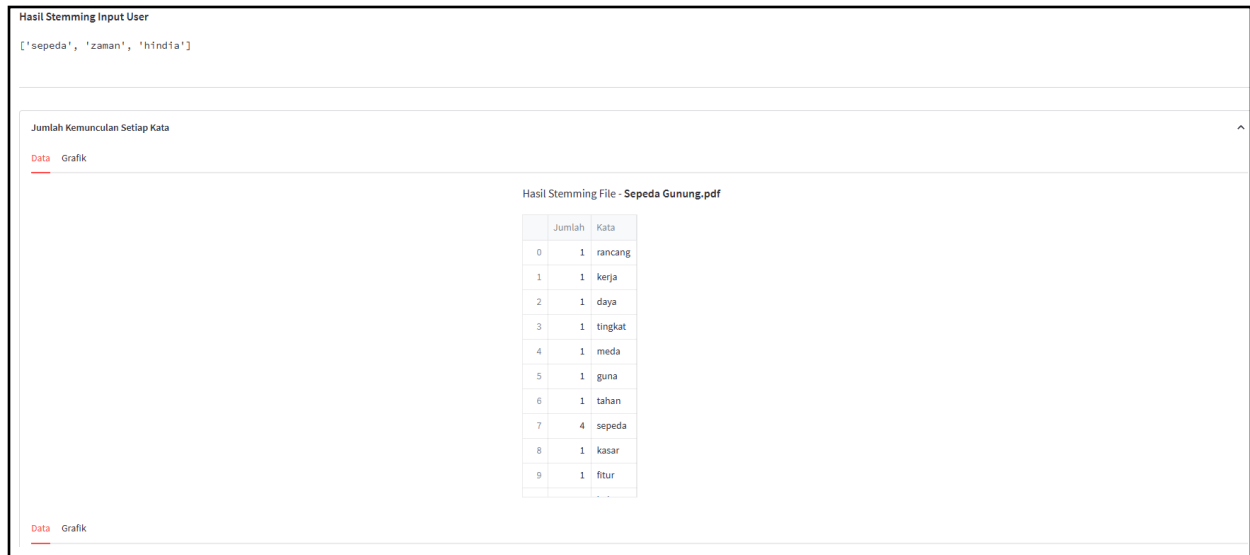


```
maupun panjangnya dan tidak dibedakan desainnya untuk pengendara pria atau wanita "
]

Stemming Text
Hasil Stemming (Nazief dan Adriani) Semua File

[
  0 : [
    0 : "sepeda"
    1 : "gunung"
    2 : "bahasa"
    3 : "inggris"
    4 : "mtb"
    5 : "sepeda"
    6 : "guna"
    7 : "medan"
    8 : "berat"
    9 : "sepeda"
    10 : "gunung"
    11 : "milik"
    12 : "beberapa"
    13 : "sama"
    14 : "sepeda"
    15 : "tetapi"
    16 : "fitur"
    17 : "rancang"
    18 : "tingkat"
    19 : "daya"
    20 : "tahan"
    21 : "kerja"
    22 : "meda"
    23 : "kasar"
  ]
]
```

Gambar 4. 8 Output Program (2)



Gambar 4. 9 Output Program (3)

Pembobotan TF - IDF

	alat	an	bagai	bahasa	baik	bantu	banyak	beberapa	beda	belanda	berat	bike	daya	dengan	desain	fitur	gerak	geser	guna	gunung	hindia	inggris	jengki	kasar	kemudian	kenal	kendara	kerja	kompak	lebih
0	0.0000	0.0000	0.0000	1.6931	0.0000	0.0000	0.0000	1.6931	0.0000	0.0000	1.6931	0.0000	1.6931	0.0000	0.0000	1.6931	0.0000	0.0000	1.6931	3.3863	0.0000	1.6931	0.0000	1.6931	0.0000	0.0000	0.0000	1.6931	0.0000	0.0000
1	1.6931	0.0000	1.6931	0.0000	0.0000	1.6931	0.0000	0.0000	1.2877	0.0000	0.0000	1.6931	0.0000	1.6931	0.0000	0.0000	1.6931	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.6931	0.0000	0.0000	0.0000	0.0000
2	0.0000	1.6931	0.0000	0.0000	1.6931	0.0000	1.6931	0.0000	1.2877	1.6931	0.0000	0.0000	0.0000	0.0000	1.6931	0.0000	0.0000	1.6931	0.0000	0.0000	1.6931	0.0000	1.6931	0.0000	1.6931	0.0000	1.6931	0.0000	1.6931	1.6931

Rujukan File Dari Hasil Euclidean Distance

Mungkin dengan query yang dimasukkan adalah **sepeda zaman hindia**, File yang dimaksud adalah **Sepeda Onthel.pdf** Dengan Jumlah Euclidean Distance : **2.227309145097825**

(Total Perhitungan Euclidean Distance Lainnya : [3.3186247903554003, 5.196152422706632, 2.227309145097825])

Gambar 4. 10 Output Program (4)

Telah diperlihatkan pada **Gambar 4.7** hingga **Gambar 4.10** dimana *output* program yang dilakukan menggunakan bahasa pemrograman Python. Implementasi sistem ini dilakukan dengan pertama – tama *user* menginput *path* folder, lalu setelah itu sistem akan melakukan ekstraksi *text* dan proses *stemming*, lalu melakukan proses perhitungan kemunculan kata – kata yang sama pada sejumlah dokumen yang dibaca atau diekstraksi, setelah itu dilakukan pembobotan dengan menggunakan proses TFIDF yang akan dijadikan sebagai bobot dalam menentukan *Euclidean distance*.

BAB V

PENUTUP

5.1 Kesimpulan

Berikut merupakan kesimpulan yang dapat diperoleh dari adanya *Data Mining* dan *Information Retrieval* yang digunakan untuk mengumpulkan sekumpulan data dengan jumlah yang besar untuk menghasilkan sebuah informasi yang dapat digunakan dengan metode temu balik informasi ini, dimana dapat disimpulkan sebagai berikut :

1. Algoritma Nazief dan Adriani merupakan suatu metode yang digunakan dalam proses *stemming* yang digunakan dalam Bahasa Indonesia, algoritma ini mengasumsikan kata sebagai *root word* yang menentukan kata awalan dan akhiran dalam proses *stemming*.
2. Teknik pembobotan term menggunakan TF-IDF adalah untuk menentukan seberapa relevan suatu kata pada dokumen tertentu dengan proses pencarian nilai *term frequency* dan *inverse document frequency* yang kemudian kedua nilai dari TF dan IDF dikalikan untuk mendapatkan nilai TF-IDF.
3. *Similarity Measure* dengan menggunakan algoritma *Euclidean Distance* digunakan untuk melakukan perhitungan dari *query* terhadap nilai pembobotan TF-IDF yang telah didapatkan, dimana hasil akhirnya akan menentukan letak dari kata pada *query* terhadap kata-kata pada dokumen tertentu dengan mengambil nilai terkecil yang didapatkan.
4. Proses perhitungan menggunakan teknik pembobotan term TF-IDF dengan cara manual dan menggunakan program menghasilkan hasil yang berbeda tetapi tidak begitu jauh karena implementasi *sklearn* terhadap TF-IDF menggunakan formula atau rumus yang sedikit berbeda dengan perhitungan manualnya.

5.2 Saran

Terdapat beberapa metode yang digunakan pada proses pembacaan dokumen dan penentuan relevansi kata terhadap dokumen tersebut, dimana setiap metode yang digunakan pastinya memiliki kelebihan dan kelemahannya masing-masing, sehingga masih perlu adanya pembelajaran lebih lanjut terkait metode yang digunakan untuk lebih mengembangkan formula pembacaan dokumen yang telah dibuat.

DAFTAR PUSTAKA

- Agusta, L. (2009). Perbandingan Algoritma Stemming Porter Dengan Algoritma Nazief dan Adriani Untuk Stemming Dokumen Teks Bahasa Indonesia. 6.
- Fahrizain, A. (2021, November 1). *Bag of Words vs TF-IDF — Penjelasan dan Perbedaannya*. Retrieved from medium.com: <https://medium.com/data-folks-indonesia/bag-of-words-vs-tf-idf-penjelasan-dan-perbedaannya-3739f32cdc72>
- Harishamzah. (2020, April 13). *Perbandingan Perhitungan Bobot TF-IDF secara Manual dan Menggunakan Python*. Retrieved from bisa-ai: <https://medium.com/bisa-ai/perbandingan-perhitungan-bobot-tf-idf-secara-manual-dan-menggunakan-python-377392a165c6>
- Jazari, I. (2019, November 30). *Perbedaan Stemming Bahasa Indonesia dengan Algoritma Nazief dan Andriani*. Retrieved from informasi-anakutm.blogspot.com: <https://informasi-anakutm.blogspot.com/2016/11/perbedaan-stemming-bahasa-indonesia.html>
- liyantanto. (2011, Juni 28). *Stemming Bahasa Indonesia dengan Algoritma Nazief dan Andriani*. Retrieved from liyantanto.wordpress.com: <https://liyantanto.wordpress.com/2011/06/>
- Nishom, M. (2019, Januari). Perbandingan Akurasi Euclidean Distance, Minkowski Distance, dan Manhattan Distance pada Algoritma K Means Clustering berbasis Chi-Square. *Jurnal Informatika: Jurnal Pengembangan IT (JPIT)*, 6. Retrieved from www.researchgate.net.
- Sabrina, S. (2022). *Pengenalan Kemiripan Teks (Text Similarity) Di Python*. Retrieved from algoritmaonline.com: <https://algoritmaonline.com/kemiripan-teks/>
- Trivusi. (2022, September 17). *Pengertian dan Jenis-jenis Distance Metric pada Machine Learning*. Retrieved from www.trivusi.web.id: <https://www.trivusi.web.id/2022/06/jenis-distance-metric.html>