

PROJECT AKHIR MATAKULIAH PENELUSURAN INFORMASI

disusun untuk memenuhi
tugas akhir matakuliah Penelusuran Informasi

oleh :

Kelompok : 1

Anggota :

Ahmad Syah Ramadhan (2208107010033)

Naufal Aqil (2208107010043)

Ganang Setyo Hadi (2208107010052)



DEPARTEMEN INFORMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SYIAH KUALA

TAHUN 2024

Daftar Isi

RINGKASAN EKSEKUTIF.....	1
1. Masalah yang ditangani.....	1
BAB I.....	2
PENDAHULUAN.....	2
1.1 Latar Belakang Masalah.....	2
1.2 Tujuan.....	2
1.3 Manfaat.....	3
BAB II.....	4
METODOLOGI.....	4
2.1 Lingkungan Pengembangan (Environment).....	4
2.1.1 Perangkat Keras.....	4
2.1.2 Perangkat Lunak.....	4
2.2 Tahapan Pelaksanaan Penelitian.....	4
2.2.1 Persiapan Awal.....	4
2.2.2 Pengumpulan Data (Web Scraping).....	4
2.2.3 Preprocessing Data.....	5
2.2.4 Pengembangan Aplikasi Web.....	5
2.2.5 Pengujian dan Validasi.....	5
2.2.6 Deployment.....	5
BAB III.....	7
IMPLEMENTASI & HASIL.....	7
3.1 Pengumpulan data (Crawling dan Scrapping).....	7
3.2 Preprocessing data.....	9
3.2.1 Pemrosesan Teks.....	9
3.2.2 Filter Kata.....	9
3.2.3 Log Aktivitas dan Kesalahan.....	9
3.2.4 Hasil Akhir Preprocessing.....	9
3.3 Menghitung TF-IDF.....	10
3.3.1 Pembangunan Frekuensi Dokumen.....	10
3.3.2 Penghitungan IDF dan TF-IDF.....	10
3.3.3 Normalisasi.....	11
3.3.4 Optimasi Proses.....	11
3.3.5 Penyimpanan Hasil.....	11
3.4 Menghitung Cosine Similarity dan Jaccard Similarity untuk Query.....	14
3.4.1 Cosine Similarity.....	14
3.4.2 Jaccard Similarity.....	15
3.4.3 Proses Pencarian dan Peringkat Dokumen (Perankingan).....	15
3.4.4 Tampilan Hasil Pencarian.....	15
3.5 Tampilan Akhir dari website.....	16
3.5.1 Tampilan Halaman Pencarian.....	16

3.5.1 Tampilan Halaman Hasil Pencarian.....	17
KESIMPULAN.....	19
REFERENSI.....	20
LAMPIRAN.....	21
a. Settingan Spider (setting.py).....	21
b. Crawling dan Scraping (full_spider.py).....	23
c. Preprocessing dan Perhitungan TF-IDF.....	30
g. Integrasi Aplikasi dengan Flask (App.py).....	38
h. Pengembangan Halaman Web.....	47

RINGKASAN EKSEKUTIF

1. Masalah yang ditangani

Dalam memenuhi tugas pada perkuliahan Penelusuran Informasi. Namun, pencarian yang tidak efisien seringkali menghasilkan banyak hasil yang tidak relevan, yang menghambat produktivitas pengguna. Oleh karena itu, kami mengembangkan mesin pencari yang memenuhi konsep-konsep yang telah diajarkan.

2. Solusi yang disarankan

Proyek ini merancang dan membangun prototipe sistem penelusuran informasi yang meliputi beberapa tahap kunci: *web crawling*, pembuatan indeks, pembuatan ranking dokumen, serta tampilan hasil pencarian melalui GUI. Dua metode pembobotan yang digunakan untuk menentukan relevansi dokumen adalah **Cosine Similarity** dan **Jaccard Similarity**. Dengan menggunakan kedua metode ini, sistem dapat memberikan hasil pencarian yang lebih akurat berdasarkan relevansi dokumen terhadap kueri pencarian.

3. Nilai Solusi

Sistem yang dikembangkan menawarkan kemudahan bagi pengguna untuk mencari dokumen terkait dengan topik tertentu secara cepat dan efisien. Dengan menerapkan dua metode pembobotan yang berbeda, pengguna dapat membandingkan hasil pencarian berdasarkan keduanya dan memilih hasil yang paling relevan. GUI yang dikembangkan memudahkan pengguna untuk melihat 10 dokumen teratas yang relevan, memberikan pengalaman pencarian yang lebih baik dan intuitif. Proyek ini juga memberikan wawasan yang lebih dalam tentang penerapan teknik pembobotan dalam sistem penelusuran informasi.

4. Kesimpulan Pekerjaan

Proyek ini sangat penting dalam mengatasi tantangan dalam sistem penelusuran informasi di dunia digital yang terus berkembang. Dengan menyediakan solusi yang mengoptimalkan akurasi hasil pencarian menggunakan metode pembobotan yang terbukti, proyek ini memiliki potensi untuk diterapkan dalam berbagai aplikasi web yang memerlukan pencarian informasi berbasis topik. Kemampuan untuk membandingkan metode pembobotan juga memberikan pemahaman yang lebih luas tentang cara meningkatkan efektivitas sistem pencarian di masa depan.

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Dalam perkembangan teknologi informasi, jumlah data yang tersedia di internet terus bertambah dengan cepat. Hal ini menghadirkan tantangan bagi pengguna untuk menemukan informasi yang relevan secara efisien di antara banyaknya data yang ada. Sistem penelusuran informasi, seperti mesin pencari, menjadi alat penting untuk memfasilitasi kebutuhan ini. Namun, keberhasilan sistem tersebut sangat bergantung pada algoritma yang digunakan untuk mengukur relevansi dokumen terhadap permintaan pencarian.

Sebagai bagian dari pembelajaran dalam bidang informatika, penting bagi mahasiswa untuk memahami prinsip kerja dan implementasi sistem penelusuran informasi. Oleh karena itu, proyek ini dirancang untuk mengembangkan prototipe sistem penelusuran informasi yang mencakup tahapan penting seperti pengumpulan data menggunakan proses crawling halaman web, pembuatan indeks dokumen, perhitungan relevansi dokumen, dan penyajian hasil pencarian melalui antarmuka pengguna.

Untuk mengukur relevansi dokumen, digunakan dua metode pembobotan, yaitu Cosine Similarity dan Jaccard Similarity. Kedua metode ini akan diterapkan untuk membandingkan efektivitasnya dalam menentukan peringkat dokumen yang relevan dengan kueri pengguna. Pendekatan ini bertujuan untuk memberikan wawasan yang lebih mendalam tentang bagaimana metode pembobotan bekerja dan mempengaruhi hasil pencarian.

Melalui proyek ini, mahasiswa tidak hanya mendapatkan pengalaman praktis dalam pengembangan sistem penelusuran informasi, tetapi juga mampu mengeksplorasi teknologi yang berpotensi untuk dikembangkan lebih lanjut, baik dalam konteks akademik maupun aplikasi di dunia nyata.

1.2 Tujuan

1) Memahami dan Mengimplementasikan Konsep Sistem Penelusuran Informasi.

Mengembangkan prototipe sistem penelusuran informasi yang mencakup tahapan inti, seperti crawling halaman web, pembuatan indeks dokumen, penentuan peringkat dokumen, dan penyajian hasil pencarian.

2) Mengevaluasi Metode Pembobotan.

Menggunakan dua metode pembobotan, yaitu Cosine Similarity dan Jaccard Similarity, untuk menghitung relevansi antara dokumen dan kueri pencarian, serta membandingkan hasilnya untuk menilai efektivitas masing-masing metode.

3) Meningkatkan Kemampuan Teknis

Melatih keterampilan mahasiswa dalam menggunakan alat dan teknik yang relevan, seperti crawling, indeksasi, pengolahan data, dan pembuatan antarmuka pengguna.

4) Menyediakan Antarmuka Pengguna yang Intuitif

Membuat GUI yang mampu menampilkan hasil pencarian berupa 10 dokumen dengan ranking tertinggi berdasarkan metode pembobotan yang diterapkan, untuk memberikan pengalaman pengguna yang optimal.

1.3 Manfaat

Proyek ini memberikan berbagai manfaat baik dari segi akademik maupun praktis. Secara akademik, mahasiswa memperoleh pemahaman mendalam mengenai konsep dan teknik dasar sistem penelusuran informasi, seperti crawling halaman web, pembuatan indeks, perhitungan relevansi dokumen, serta penentuan ranking. Implementasi dua metode pembobotan, yaitu Cosine Similarity dan Jaccard Similarity, memberikan wawasan praktis mengenai efektivitas masing-masing metode dalam menghasilkan hasil pencarian yang relevan. Selain itu, melalui pengembangan antarmuka pengguna (GUI), mahasiswa dapat mempelajari cara menyajikan informasi yang ramah pengguna dan intuitif, meningkatkan keterampilan dalam desain dan implementasi sistem berbasis aplikasi.

Proyek ini juga membantu melatih penguasaan mahasiswa terhadap berbagai tools dan teknik yang relevan di dunia industri, sehingga meningkatkan daya saing mereka dalam menghadapi tantangan di dunia kerja. Secara praktis, hasil proyek dapat dijadikan referensi atau landasan untuk pengembangan sistem penelusuran informasi yang lebih efisien dan canggih di masa depan, baik untuk kebutuhan akademik maupun industri. Dengan mengevaluasi kinerja dua metode pembobotan yang digunakan, proyek ini juga memberikan kontribusi pada eksplorasi dan inovasi di bidang penelusuran informasi.

BAB II

METODOLOGI

2.1 Lingkungan Pengembangan (Environment)

2.1.1 Perangkat Keras

- Komputer/Laptop dengan spesifikasi:
 - Prosesor minimal Intel Core i5
 - RAM minimal 8 GB
 - Penyimpanan SSD minimal 256 GB

2.1.2 Perangkat Lunak

1. Sistem Operasi
 - Windows 11
 - Linux Ubuntu
 - Python versi 3.11.1
2. Lingkungan Pengembangan
 - Virtual Environment (venv/conda)
 - Integrated Development Environment (IDE): Visual Studio Code
3. Alat Pengembangan
 - Scrapy: Framework web scraping
 - Flask: Framework web development
 - Scikit-learn: Library machine learning untuk perhitungan TF-IDF
 - Pandas: Manipulasi dan analisis data
 - NLTK: Library untuk stemming
4. Alat Tambahan
 - Git: Kontrol versi
 - Postman: Pengujian API
 - Browser web untuk debugging

2.2 Tahapan Pelaksanaan Penelitian

2.2.1 Persiapan Awal

1. Instalasi Perangkat Lunak
 - Buat virtual environment
 - Install dependencies yang diperlukan
 - Konfigurasi environment pengembangan
2. Perancangan Arsitektur Sistem
 - Desain struktur direktori proyek
 - Tentukan alur kerja sistem
 - Identifikasi kebutuhan fungsional

2.2.2 Pengumpulan Data (Web Scraping)

1. Persiapan Crawler
 - Tentukan situs target
 - Buat spider Scrapy
 - Konfigurasi batasan crawling:
 - Depth limit
 - Download delay
 - User-agent
 - Implementasi mekanisme penghindaran blokir
2. Implementasi Scraping
 - Eksekusi spider
 - Simpan data mentah
 - Validasi hasil scraping

2.2.3 Preprocessing Data

1. Pembersihan Teks
 - Tokenisasi
 - Penghapusan stopwords
 - Stemming menggunakan Sastrawi
2. Perhitungan TF-IDF
 - Hitung document frequency
 - Lakukan filtering kata
 - Hitung nilai TF-IDF
 - Normalisasi data
 - Bangun inverted index

2.2.4 Pengembangan Aplikasi Web

1. Struktur Backend dengan Flask
 - Buat route dan endpoint
 - Integrasikan logic pencarian
 - Implementasi metode similarity
2. Pengembangan Frontend
 - Desain antarmuka menggunakan HTML/CSS
 - Tambahkan interaktivitas dengan JavaScript
 - Integrasikan frontend dengan backend Flask

2.2.5 Pengujian dan Validasi

1. Pengujian Fungsional
 - Uji setiap modul
 - Validasi hasil scraping
 - Tes akurasi pencarian
2. Pengujian Performa
 - Mittung waktu respon
 - Evaluasi konsumsi sumber daya
 - Optimasi jika diperlukan

2.2.6 Deployment

1. Konfigurasi Server
 - Pilih metode deployment
 - Konfiguasi server produksi
 - Setup web server (Gunicorn/uWSGI)
2. Manajemen Keamanan
 - Terapkan HTTPS
 - Konfigurasikan firewall
 - Lakukan update berkala

BAB III

HASIL

Bab ini menjelaskan secara rinci mengenai langkah-langkah atau alur yang diterapkan untuk menyelesaikan permasalahan yang telah dirumuskan sebelumnya. Penjabaran dalam bab ini mencakup tahapan implementasi solusi, mulai dari tahap awal persiapan hingga pelaksanaan dan evaluasi. Setiap proses yang dilakukan dijelaskan secara sistematis untuk memberikan gambaran yang jelas mengenai metode yang digunakan dalam penyelesaian masalah. Selain itu, hasil yang diperoleh dari implementasi tersebut juga disajikan dalam bentuk yang terukur dan terstruktur, sehingga memudahkan pembaca untuk memahami hubungan antara proses pelaksanaan dengan output yang dihasilkan. Bab ini diharapkan dapat memberikan pemahaman menyeluruh tentang bagaimana permasalahan diatasi melalui pendekatan yang telah dirancang.

3.1 Pengumpulan data (Crawling dan Scrapping)

Langkah pertama dalam proyek ini adalah **tahapan pengumpulan data**, yang dilakukan dengan menggunakan proses crawling menggunakan framework **Scrapy**. Pada tahap ini, kami merancang pola URL untuk menentukan halaman-halaman yang diizinkan dan dilarang diakses oleh crawler. Pola URL yang diizinkan mencakup artikel-artikel pada situs **halodoc.com**, yaitu halaman yang memiliki format:

["https://www.halodoc.com/artikel/"](https://www.halodoc.com/artikel/).

Sementara itu, untuk menghindari pengumpulan data yang tidak relevan atau yang berpotensi menimbulkan error, kami menetapkan pola URL yang dilarang. Beberapa pola URL yang dilarang antara lain:

- Halaman yang berisi informasi tentang obat atau suplemen, seperti `"/obatdansuplemen/"` dan `"/obat-konten/"`.
- Halaman yang mengembalikan error, misalnya `"/not-found"`, `"/server-error"`, dan `"/general-error"`.
- Halaman hasil validasi medis seperti `"/validasi/hasiltest/"`.
- Sitemap yang tidak relevan, seperti `"/sitemap_cari_doctor9.xml"` dan `"/sitemap_rumah_sakit1.xml"`.
- Halaman pencarian dokter atau rumah sakit terdekat, seperti `"/cari-dokter/terdekat/"` dan `"/rumah-sakit/terdekat/"`.

Dengan pendekatan ini, crawler hanya akan fokus pada artikel-artikel kesehatan yang relevan, mengurangi risiko pengambilan data yang tidak diperlukan dan meningkatkan efisiensi proses crawling.

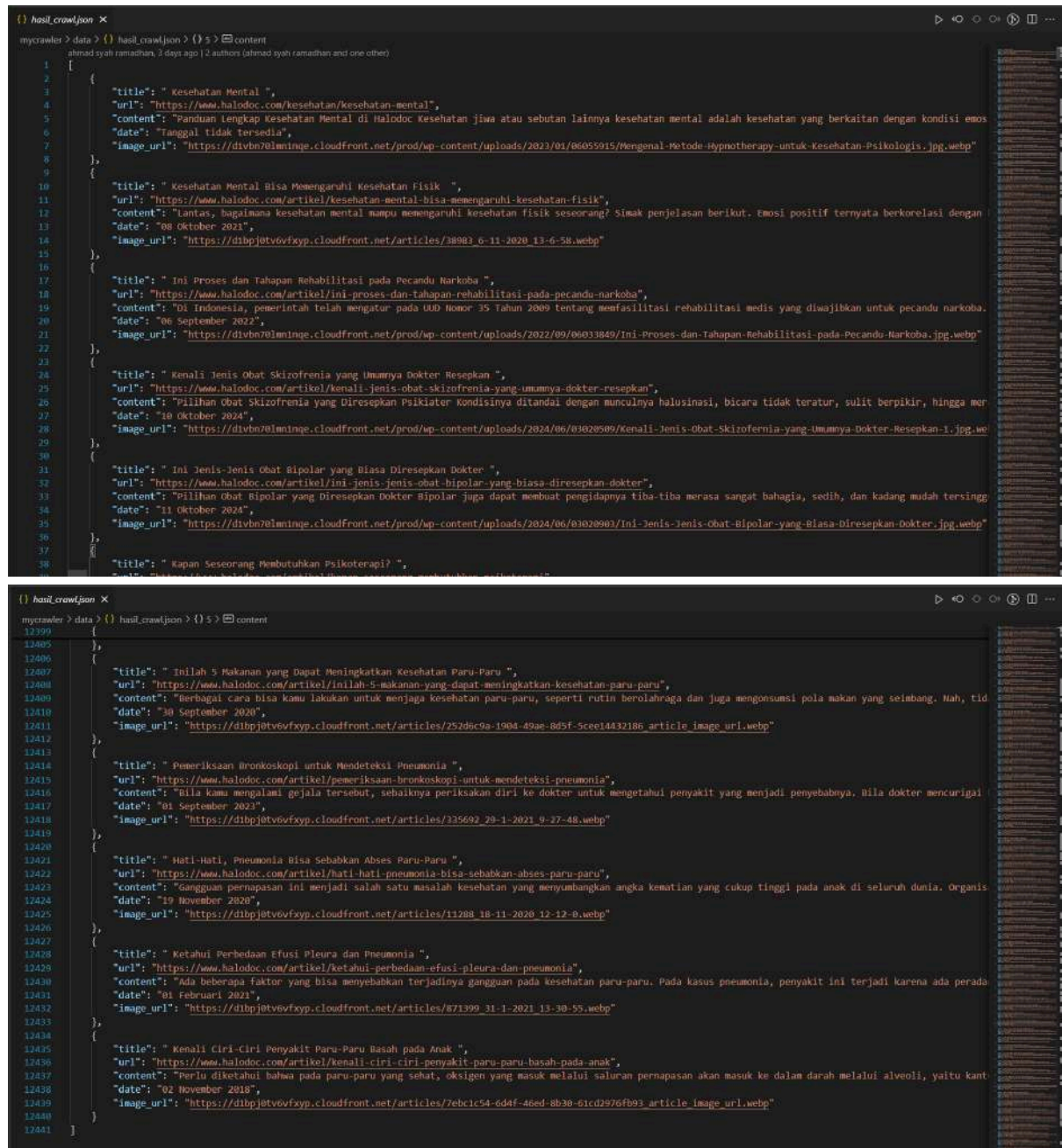
Setelah tautan-tautan yang sesuai diperoleh melalui proses crawling, langkah berikutnya adalah melakukan **scrapping** pada setiap tautan tersebut. Pada tahap ini, data-data yang relevan diekstraksi dari setiap halaman artikel. Informasi yang diambil meliputi:

- **Judul artikel:** Nama atau judul utama dari artikel yang mewakili isi artikel.
- **URL artikel:** Alamat tautan artikel sebagai sumber referensi.
- **Isi artikel:** Konten utama artikel berupa teks yang berisi informasi kesehatan.
- **Tanggal publikasi:** Waktu atau tanggal artikel diterbitkan untuk melacak relevansi informasi.
- **URL gambar:** Alamat tautan gambar yang terkait dengan artikel tersebut, jika tersedia.

Proses scraping dilakukan secara sistematis untuk memastikan bahwa data yang diambil memiliki struktur yang konsisten dan sesuai dengan kebutuhan proyek. Data-data ini kemudian disimpan dalam format yang terorganisir untuk digunakan pada tahap-tahap berikutnya, seperti preprocessing dan perhitungan relevansi.

Metode yang kami gunakan dalam tahapan ini memastikan bahwa pengumpulan data dilakukan secara efisien, terarah, dan etis, dengan tetap mematuhi aturan dan batasan yang ditetapkan oleh situs web yang bersangkutan.

Berikut adalah gambar untuk hasil crawling dan scrapping



```
{} hasil_crawling x
mycrawler > data > {} hasil_crawling > {} > content
ahmad syah ramadhan, 3 days ago | 2 authors (ahmad syah ramadhan and one other)
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2
```

3.2 Preprocessing data

Tahap preprocessing merupakan langkah krusial untuk memastikan bahwa data mentah hasil crawling dapat digunakan secara optimal dalam perhitungan *Term Frequency-Inverse Document Frequency* (TF-IDF). Berikut adalah penjelasan langkah-langkah yang dilakukan pada tahap ini:

3.2.1 Pemrosesan Teks

Proses awal melibatkan pembersihan teks untuk menghasilkan kata-kata yang relevan dengan kebutuhan analisis. Teks mentah diproses dengan langkah-langkah berikut:

- **Normalisasi Teks:** Seluruh teks diubah menjadi huruf kecil untuk menyamakan format dan menghindari penghitungan ganda akibat perbedaan kapitalisasi.
- **Tokenisasi:** Teks dipecah menjadi unit kata (token) untuk mempermudah analisis lanjutan.
- **Penghapusan Kata Berhenti (Stop Words):** Kata-kata umum dalam Bahasa Indonesia, seperti "yang", "dengan", "atau", dihapus karena tidak berkontribusi pada makna utama dokumen.
- **Stemming:** Mengembalikan kata ke bentuk dasar (stem) menggunakan pustaka Sastrawi. Misalnya, kata "berjalan" diubah menjadi "jalan".

3.2.2 Filter Kata

Kata-kata yang tidak relevan disaring berdasarkan:

- **Panjang Kata Minimum:** Kata yang memiliki panjang kurang dari tiga karakter diabaikan untuk mencegah penghitungan kata yang tidak bermakna, seperti "di" atau "ke".
- **Frekuensi Dokumen:** Kata-kata yang muncul terlalu jarang atau terlalu sering di seluruh dokumen diabaikan berdasarkan nilai ambang batas:
 - **Minimum Document Frequency (Min-DF):** Kata harus muncul di setidaknya dua dokumen agar dianggap relevan.
 - **Maximum Document Frequency (Max-DF):** Kata yang muncul di lebih dari 85% dokumen dianggap terlalu umum dan dihapus dari analisis.

3.2.3 Log Aktivitas dan Kesalahan

Setiap langkah preprocessing dicatat menggunakan mekanisme *logging*, termasuk waktu pelaksanaan, jumlah data yang diproses, serta peringatan atau kesalahan yang terjadi. Hal ini memastikan bahwa proses berjalan transparan dan mudah dilacak jika ada masalah.

3.2.4 Hasil Akhir Preprocessing

Hasil dari preprocessing adalah daftar kata yang telah dibersihkan dan disaring untuk setiap dokumen. Kata-kata ini kemudian siap digunakan dalam perhitungan TF-IDF untuk mengevaluasi relevansi dokumen berdasarkan kueri pengguna.

3.3 Menghitung TF-IDF

Tahap ini bertujuan untuk menghitung skor relevansi kata terhadap dokumen menggunakan pendekatan *Term Frequency-Inverse Document Frequency* (TF-IDF). Perhitungan ini dilakukan dalam dua langkah utama: **pembangunan frekuensi dokumen** dan **penghitungan nilai TF-IDF**, dengan detail sebagai berikut:

3.3.1 Pembangunan Frekuensi Dokumen

Langkah pertama adalah menghitung frekuensi kata di seluruh dokumen. Untuk setiap dokumen:

- **Term Frequency (TF)** dihitung berdasarkan jumlah kemunculan sebuah kata dalam dokumen tertentu dibandingkan dengan total jumlah kata dalam dokumen tersebut. TF memberikan bobot yang lebih tinggi pada kata-kata yang sering muncul di dokumen tertentu.
- **Document Frequency (DF)** dihitung sebagai jumlah dokumen yang mengandung sebuah kata tertentu. DF digunakan untuk mengevaluasi seberapa umum kata tersebut di seluruh koleksi dokumen.

Seluruh kata yang memenuhi ambang batas frekuensi (ditentukan pada tahap preprocessing) dimasukkan ke dalam *inverted index*, yaitu struktur data yang memetakan setiap kata ke daftar dokumen tempat kata tersebut muncul.

3.3.2 Penghitungan IDF dan TF-IDF

Setelah frekuensi kata dihitung, langkah berikutnya adalah menghitung *Inverse Document Frequency* (IDF) dan skor TF-IDF:

- **Inverse Document Frequency (IDF)** memberikan bobot lebih tinggi pada kata-kata yang jarang muncul di seluruh dokumen. Rumus IDF yang digunakan adalah:

$$IDF = \log \left(\frac{N + 1}{DF + 1} \right) + 1$$

di mana N adalah total jumlah dokumen, dan DF adalah frekuensi dokumen dari sebuah kata. Penambahan "1" pada pembilang dan penyebut dilakukan untuk mencegah nilai nol dan memastikan kestabilan perhitungan.

- **TF-IDF** dihitung dengan mengalikan nilai TF dan IDF untuk setiap kata dalam dokumen tertentu:

$$FT - IDF = TF \times IDF$$

3.3.3 Normalisasi

Agar skor TF-IDF dapat dibandingkan secara konsisten antar dokumen, setiap vektor TF-IDF dinormalisasi menggunakan norma Euclidean. Ini dilakukan dengan membagi setiap skor TF-IDF dengan panjang vektor (norma), sehingga skor akhir memiliki skala yang seragam.

3.3.4 Optimasi Proses

Perhitungan TF-IDF dilakukan dalam dua *pass* untuk memastikan efisiensi:

- *First pass* digunakan untuk menghitung DF dan membangun *vocabulary* (daftar kata yang relevan).
- *Second pass* digunakan untuk menghitung nilai TF-IDF setiap kata dalam dokumen berdasarkan *vocabulary* yang telah dibangun sebelumnya.

3.3.5 Penyimpanan Hasil

Hasil akhir perhitungan disimpan dalam beberapa berkas untuk keperluan analisis dan pencarian selanjutnya:

- **Inverted Index:** Berisi daftar kata dan dokumen tempat kata tersebut muncul.
- **TF-IDF Index:** Berisi skor TF-IDF setiap kata untuk masing-masing dokumen.
- **Vocabulary:** Berisi daftar kata yang digunakan dalam perhitungan TF-IDF beserta indeksnya.

Dengan pendekatan ini, sistem dapat memberikan skor relevansi yang akurat untuk mendukung pencarian informasi berbasis query pengguna. Seluruh proses juga dilengkapi dengan pelacakan kemajuan (*progress tracking*) dan log aktivitas untuk memastikan efisiensi dan transparansi.

Berikut adalah gambar untuk inverted index :

```
{ } inverted_index.json X
mycrawler > output_indices > { } inverted_index.json > [ ] beberapa
1 {
2   "beberapa": [
3     "https://www.halodoc.com/artikel/plak-pada-gigi-sebabkan-periodontitis-benarkah",
4     "https://www.halodoc.com/artikel/ini-tahapan-pengujian-dan-perkembangan-global-vaksin-corona",
5     "https://www.halodoc.com/artikel/ini-dokter-gizi-yang-bisa-beri-info-seputar-intermittent-fasting",
6     "https://www.halodoc.com/artikel/kata-dokter-berbagai-jenis-makanan-sehat-untuk-jantung",
7     "https://www.halodoc.com/artikel/benarkah-lidah-buaya-mampu-atasi-penuaan-dini? single=true",
8     "https://www.halodoc.com/artikel/efek-samping-vaksinasi-covid-19-ketahui-tentang-kipi",
9     "https://www.halodoc.com/artikel/tidak-semua-infeksi-memerlukan-pengobatan-antibiotik",
10    "https://www.halodoc.com/artikel/perhatikan-hal-ini-setelah-mendapatkan-vaksin-covid-19-1",
11    "https://www.halodoc.com/artikel/berapa-berat-badan-yang-dikategorikan-obesitas",
12    "https://www.halodoc.com/artikel/cara-mengetahui-kehamilan-melalui-denyut-nadi-seberapa-akuratkah",
13    "https://www.halodoc.com/artikel/ini-cara-mudah-membuat-jus-jeruk-segar-dan-kaya-nutrisi",
14    "https://www.halodoc.com/artikel/cara-virus-corona-menyerang-tubuh",
15    "https://www.halodoc.com/artikel/anatomi-sistem-perkemihan-fungsi-dan-peryakit-yang-mengintai",
16    "https://www.halodoc.com/artikel/carpal-tunnel-syndrome-berbahaya-atau-tidak",
17    "https://www.halodoc.com/artikel/ini-2-penyakit-gigi-kuning-dan-cara-mudah-mengatasinya",
18    "https://www.halodoc.com/artikel/ini-yang-terjadi-jika-tubuh-kurang-nutrisi",
19    "https://www.halodoc.com/artikel/ibu-hamil-harus-banyak-tanya-dan-banyak-bergerak-ini-alasannya?utm_tracker=19d12eb0-809a-4236-885e-df0acdfa01b0",
20    "https://www.halodoc.com/artikel/pola-asuh-anak-yang-harus-dijauhi-oleh-calon-ayah?utm_tracker=",
21    "https://www.halodoc.com/artikel/jangan-salah-ini-bedanya-penyakit-kronis-dan-akut",
22    "https://www.halodoc.com/artikel/ini-jenis-dan-rekomendasi-obat-hiv-yang-perlu-diketahui",
23    "https://www.halodoc.com/artikel/kebiasaan-menahan-buang-air-kecil-bisa-sebabkan-batu-ginjal",
24    "https://www.halodoc.com/artikel/kenal-variante-covid-19-b1617-yang-muncul-di-india",
25    "https://www.halodoc.com/artikel/ciri-ciri-cacar-air-yang-segera-membutuhkan-penanganan",
26    "https://www.halodoc.com/artikel/muncul-gejala-pneumonia-segera-hubungi-dokter-ini",
27    "https://www.halodoc.com/artikel/5-makanan-yang-sebaiknya-dikonsumsi-saat-asam-lambung-naik",
28    "https://www.halodoc.com/artikel/manfaat-mengonsumsi-pisang-saat-sahur",
29    "https://www.halodoc.com/artikel/waspada-ini-8-gejala-diabetes-melitus",
30    "https://www.halodoc.com/artikel/kenal-5-cara-tepat-memilih-facial-wash-untuk-remaja? single=true",
31    "https://www.halodoc.com/artikel/begitulah-penanganan-dan-pencegahan-splenomegali",
32    "https://www.halodoc.com/artikel/atasi-asam-lambung-dengan-memanfaatkan-lidah-buaya? single=true",
33    "https://www.halodoc.com/artikel/5-cara-mengatasi-batu-ginjal",
34    "https://www.halodoc.com/artikel/ini-5-rekomendasi-obat-wasir-herbal-yang-ampuh-atasi-ambeien",
35    "https://www.halodoc.com/artikel/cara-kerja-vaksin-virus-corona-pada-tubuh?utm_tracker=eeaf18ea-bd52-4d9f-a67b-6c397ccf1608",
36    "https://www.halodoc.com/artikel/5-makanan-yang-dapat-membantu-kesehatan-kulit? single=true",
37    "https://www.halodoc.com/artikel/terjangkit-virus-corona-kapan-gejalanya-akan-borakhir",
38    "https://www.halodoc.com/artikel/waspada-vivid-dream-rentan-terjadi-selama-pandemi-corona",
39    "https://www.halodoc.com/artikel/benarkah-minyak-zaitun-baik-untuk-kesehatan-jantung?utm_tracker=19d12eb0-809a-4236-885e-df0acdfa01b0",
40  ]
}
```

```
{ } inverted_index.json X
mycrawler > output_indices > { } inverted_index.json > [ ] beberapa
111638 {
111639   "aorta": [
111640     "https://www.halodoc.com/artikel/ini-yang-terjadi-kalau-terlalu-sering-terpapar-asap-rokok",
111641     "https://www.halodoc.com/artikel/2-tipe-penyakit-katup-jantung-yang-perlu-diketahui"
111642   ],
111643   "kubah": [
111644     "https://www.halodoc.com/artikel/kenali-pengaruh-otot-diafragma-terhadap-pernapasan-manusia",
111645     "https://www.halodoc.com/artikel/masalah-respirasi-apa-yang-harus-dilakukan"
111646   ],
111647   "pneumothorax": [
111648     "https://www.halodoc.com/artikel/waspada-ini-tanda-lansia-alami-gangguan-pernapasan",
111649     "https://www.halodoc.com/artikel/tension-pneumothorax-penyebab-gejala-dan-cara-tepat-mengatasinya"
111650   ],
111651   "tension": [
111652     "https://www.halodoc.com/artikel/waspada-ini-tanda-lansia-alami-gangguan-pernapasan",
111653     "https://www.halodoc.com/artikel/tension-pneumothorax-penyebab-gejala-dan-cara-tepat-mengatasinya"
111654   ],
111655   "bronkoskopi": [
111656     "https://www.halodoc.com/artikel/pemeriksaan-bronkoskopi-untuk-mendeteksi-pneumonia",
111657     "https://www.halodoc.com/artikel/masalah-respirasi-apa-yang-harus-dilakukan"
111658   ],
111659   "pdpi": [
111660     "https://www.halodoc.com/artikel/ini-6-obat-alami-paru-paru-yang-dapat-melegakan-pernapasan",
111661     "https://www.halodoc.com/artikel/muncul-gejala-pneumonia-segera-hubungi-dokter-ini"
111662   ],
111663   "wesnawa": [
111664     "https://www.halodoc.com/artikel/ini-6-obat-alami-paru-paru-yang-dapat-melegakan-pernapasan",
111665     "https://www.halodoc.com/artikel/muncul-gejala-pneumonia-segera-hubungi-dokter-ini"
111666   ],
111667   "agustya": [
111668     "https://www.halodoc.com/artikel/ini-6-obat-alami-paru-paru-yang-dapat-melegakan-pernapasan",
111669     "https://www.halodoc.com/artikel/muncul-gejala-pneumonia-segera-hubungi-dokter-ini"
111670   ],
111671   "sp p": [
111672     "https://www.halodoc.com/artikel/ini-6-obat-alami-paru-paru-yang-dapat-melegakan-pernapasan",
111673     "https://www.halodoc.com/artikel/muncul-gejala-pneumonia-segera-hubungi-dokter-ini"
111674   ]
111675 }
```

Berikut adalah gambar untuk hasil TF-IDF:

berdasarkan *TF-IDF* dari kata-kata yang ada. Setelah query di-*stem* dan diproses, sebuah *query vector* dibuat, yang berisi frekuensi dari setiap kata dalam query tersebut. Vektor dokumen sudah ada sebelumnya dalam indeks *TF-IDF* yang disimpan.

Untuk menghitung kemiripan antara query dan dokumen, kedua vektor (query dan dokumen) diubah menjadi array numpy dan dihitung menggunakan fungsi **cosine_similarity** dari library **sklearn**. Fungsi ini mengukur kesamaan antara kedua vektor tersebut dengan menghitung *cosine of the angle* di antara mereka. Hasilnya adalah nilai antara 0 dan 1, di mana nilai yang lebih tinggi menunjukkan kemiripan yang lebih besar.

3.4.2 Jaccard Similarity

Jaccard Similarity adalah metrik yang mengukur kesamaan antara dua himpunan dengan membandingkan ukuran irisan (*intersection*) dan gabungan (*union*) dari dua himpunan tersebut. Pada implementasi ini, kedua himpunan diambil dari kata-kata dalam query dan dokumen, yang masing-masing disimpan dalam bentuk dictionary. Untuk menghitung kemiripan Jaccard, pertama-tama dihitung *intersection* dari kata-kata antara query dan dokumen, yaitu jumlah kata yang ada di kedua himpunan. Kemudian dihitung *union* dari kata-kata tersebut, yaitu jumlah kata unik yang ada di salah satu atau kedua himpunan. Kemiripan dihitung sebagai rasio antara *intersection* dan *union*. Jika *union* adalah 0 (yaitu, kedua himpunan kosong), nilai kemiripan diset menjadi 0 untuk menghindari pembagian dengan nol.

3.4.3 Proses Pencarian dan Peringkat Dokumen (Perankingan)

Ketika query pencarian dimasukkan, proses ini dimulai dengan stemming kata-kata dalam query dan menghitung vektor frekuensi kata-kata tersebut. Selanjutnya, baik **Cosine Similarity** maupun **Jaccard Similarity** digunakan untuk membandingkan vektor query dengan vektor dari setiap dokumen dalam indeks.

Hasil perhitungan kemiripan digunakan untuk menyusun daftar dokumen yang relevan dengan query. Dokumen yang memiliki nilai kemiripan lebih tinggi dengan query akan memiliki prioritas lebih tinggi. Daftar hasil pencarian ini kemudian diurutkan berdasarkan skor kemiripan dari tertinggi ke terendah, dan hanya 10 dokumen teratas yang ditampilkan sebagai hasil pencarian.

3.4.4 Tampilan Hasil Pencarian

Setelah skor kemiripan dihitung, hasil pencarian disajikan kepada pengguna dengan mencantumkan judul, cuplikan konten, tag, tanggal publikasi, dan gambar artikel (dengan gambar pengganti jika gambar tidak tersedia). Hasil ini kemudian ditampilkan dalam antarmuka pengguna yang dapat dinavigasi melalui halaman-halaman yang telah disediakan.

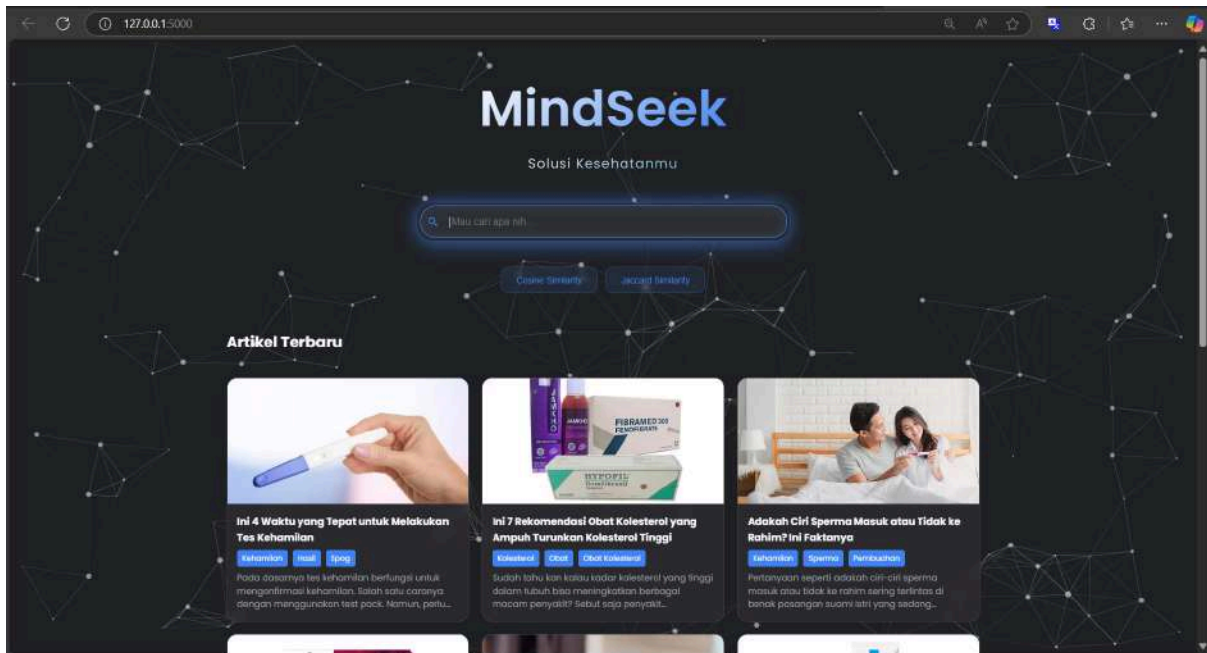
Dengan menggunakan dua metode ini, pengguna dapat memilih metode mana yang lebih sesuai untuk kebutuhan pencarian mereka, baik itu menggunakan **Cosine Similarity** untuk pengukuran berdasarkan vektor atau **Jaccard Similarity** untuk pengukuran berbasis himpunan.

Berikut adalah gambar json dari tag yang digunakan saat menampilkan hasil pencarian

```
{} tags_index.json X
mycrawler > output_indices > {} tags_index.json > ...
Ganang Setyo Hadi, yesterday | 1 author (Ganang Setyo Hadi)
Ganang Setyo Hadi, yesterday + FIXING MASALAH LAPA KALI HAS EXTRACT TAGS KEK P...
1  "https://www.halodoc.com/kesehatan/kesehatan-mental": {
2    "Kesehatan",
3    "Mental",
4    "Kesehatan Mental"
5  },
6  },
7  "https://www.halodoc.com/artikel/kesehatan-mental-bisa-memengaruhi-kesehatan-fisik": {
8    "Kesehatan",
9    "Mental",
10   "Fisik"
11  },
12  "https://www.halodoc.com/artikel/ini-proses-dan-tahapan-rehabilitasi-pada-pecandu-narkoba": {
13    "Pecandu",
14    "Narkoba",
15    "Pecandu Narkoba"
16  },
17  "https://www.halodoc.com/artikel/kenali-jenis-obat-skizofrenia-yang-umumnya-dokter-resepkan": {
18    "Skizofrenia",
19    "Obat",
20    "Psikiater"
21  },
22  "https://www.halodoc.com/artikel/ini-jenis-jenis-obat-bipolar-yang-biasa-diresepkan-dokter": {
23    "Obat",
24    "Bipolar",
25    "Obat Bipolar"
26  },
27  "https://www.halodoc.com/artikel/apan-seseorang-membutuhkan-psikoterapi": {
28    "Psikoterapi",
29    "Mengatasi",
30    "Mental"
31  },
32  "https://www.halodoc.com/artikel/waspada-ini-bahaya-self-diagnosis-kondisi-kesehatan-mental": {
33    "Mental",
34    "Diagnosis",
35    "Kesehatan"
36  },
37  "https://www.halodoc.com/artikel/9-cara-sederhana-menjaga-kesehatan-mental": {
38    "Membantu",
39    "Kesehatan"
40  },
41  },
42  },
43  },
44  },
45  },
46  },
47  },
48  },
49  },
50  },
51  },
52  },
53  },
54  },
55  },
56  },
57  },
58  },
59  },
60  },
61  },
62  },
63  },
64  },
65  },
66  },
67  },
68  },
69  },
70  },
71  },
72  },
73  },
74  },
75  },
76  },
77  },
78  },
79  },
80  },
81  },
82  },
83  },
84  },
85  },
86  },
87  },
88  },
89  },
90  },
91  },
92  },
93  },
94  },
95  },
96  },
97  },
98  },
99  },
100 },
101 },
102 },
103 },
104 },
105 },
106 },
107 },
108 },
109 },
110 },
111 },
112 },
113 },
114 },
115 },
116 },
117 },
118 },
119 },
120 },
121 },
122 },
123 },
124 },
125 },
126 },
127 },
128 },
129 },
130 },
131 },
132 },
133 },
134 },
135 },
136 },
137 },
138 },
139 },
140 },
141 },
142 },
143 },
144 },
145 },
146 },
147 },
148 },
149 },
150 },
151 },
152 },
153 },
154 },
155 },
156 },
157 },
158 },
159 },
160 },
161 },
162 },
163 },
164 },
165 },
166 },
167 },
168 },
169 },
170 },
171 },
172 },
173 },
174 },
175 },
176 },
177 },
178 },
179 },
180 },
181 },
182 },
183 },
184 },
185 },
186 },
187 },
188 },
189 },
190 },
191 },
192 },
193 },
194 },
195 },
196 },
197 },
198 },
199 },
200 },
201 },
202 },
203 },
204 },
205 },
206 },
207 },
208 },
209 },
210 },
211 },
212 },
213 },
214 },
215 },
216 },
217 },
218 },
219 },
220 },
221 },
222 },
223 },
224 },
225 },
226 },
227 },
228 },
229 },
230 },
231 },
232 },
233 },
234 },
235 },
236 },
237 },
238 },
239 },
240 },
241 },
242 },
243 },
244 },
245 },
246 },
247 },
248 },
249 },
250 },
251 },
252 },
253 },
254 },
255 },
256 },
257 },
258 },
259 },
260 },
261 },
262 },
263 },
264 },
265 },
266 },
267 },
268 },
269 },
270 },
271 },
272 },
273 },
274 },
275 },
276 },
277 },
278 },
279 },
280 },
281 },
282 },
283 },
284 },
285 },
286 },
287 },
288 },
289 },
290 },
291 },
292 },
293 },
294 },
295 },
296 },
297 },
298 },
299 },
300 },
301 },
302 },
303 },
304 },
305 },
306 },
307 },
308 },
309 },
310 },
311 },
312 },
313 },
314 },
315 },
316 },
317 },
318 },
319 },
320 },
321 },
322 },
323 },
324 },
325 },
326 },
327 },
328 },
329 },
330 },
331 },
332 },
333 },
334 },
335 },
336 },
337 },
338 },
339 },
340 },
341 },
342 },
343 },
344 },
345 },
346 },
347 },
348 },
349 },
350 },
351 },
352 },
353 },
354 },
355 },
356 },
357 },
358 },
359 },
360 },
361 },
362 },
363 },
364 },
365 },
366 },
367 },
368 },
369 },
370 },
371 },
372 },
373 },
374 },
375 },
376 },
377 },
378 },
379 },
380 },
381 },
382 },
383 },
384 },
385 },
386 },
387 },
388 },
389 },
390 },
391 },
392 },
393 },
394 },
395 },
396 },
397 },
398 },
399 },
400 },
401 },
402 },
403 },
404 },
405 },
406 },
407 },
408 },
409 },
410 },
411 },
412 },
413 },
414 },
415 },
416 },
417 },
418 },
419 },
420 },
421 },
422 },
423 },
424 },
425 },
426 },
427 },
428 },
429 },
430 },
431 },
432 },
433 },
434 },
435 },
436 },
437 },
438 },
439 },
440 },
441 },
442 },
443 },
444 },
445 },
446 },
447 },
448 },
449 },
450 },
451 },
452 },
453 },
454 },
455 },
456 },
457 },
458 },
459 },
460 },
461 },
462 },
463 },
464 },
465 },
466 },
467 },
468 },
469 },
470 },
471 },
472 },
473 },
474 },
475 },
476 },
477 },
478 },
479 },
480 },
481 },
482 },
483 },
484 },
485 },
486 },
487 },
488 },
489 },
490 },
491 },
492 },
493 },
494 },
495 },
496 },
497 },
498 },
499 },
500 },
501 },
502 },
503 },
504 },
505 },
506 },
507 },
508 },
509 },
510 },
511 },
512 },
513 },
514 },
515 },
516 },
517 },
518 },
519 },
520 },
521 },
522 },
523 },
524 },
525 },
526 },
527 },
528 },
529 },
530 },
531 },
532 },
533 },
534 },
535 },
536 },
537 },
538 },
539 },
540 },
541 },
542 },
543 },
544 },
545 },
546 },
547 },
548 },
549 },
550 },
551 },
552 },
553 },
554 },
555 },
556 },
557 },
558 },
559 },
560 },
561 },
562 },
563 },
564 },
565 },
566 },
567 },
568 },
569 },
570 },
571 },
572 },
573 },
574 },
575 },
576 },
577 },
578 },
579 },
580 },
581 },
582 },
583 },
584 },
585 },
586 },
587 },
588 },
589 },
590 },
591 },
592 },
593 },
594 },
595 },
596 },
597 },
598 },
599 },
600 },
601 },
602 },
603 },
604 },
605 },
606 },
607 },
608 },
609 },
610 },
611 },
612 },
613 },
614 },
615 },
616 },
617 },
618 },
619 },
620 },
621 },
622 },
623 },
624 },
625 },
626 },
627 },
628 },
629 },
630 },
631 },
632 },
633 },
634 },
635 },
636 },
637 },
638 },
639 },
640 },
641 },
642 },
643 },
644 },
645 },
646 },
647 },
648 },
649 },
650 },
651 },
652 },
653 },
654 },
655 },
656 },
657 },
658 },
659 },
660 },
661 },
662 },
663 },
664 },
665 },
666 },
667 },
668 },
669 },
670 },
671 },
672 },
673 },
674 },
675 },
676 },
677 },
678 },
679 },
680 },
681 },
682 },
683 },
684 },
685 },
686 },
687 },
688 },
689 },
690 },
691 },
692 },
693 },
694 },
695 },
696 },
697 },
698 },
699 },
700 },
701 },
702 },
703 },
704 },
705 },
706 },
707 },
708 },
709 },
710 },
711 },
712 },
713 },
714 },
715 },
716 },
717 },
718 },
719 },
720 },
721 },
722 },
723 },
724 },
725 },
726 },
727 },
728 },
729 },
730 },
731 },
732 },
733 },
734 },
735 },
736 },
737 },
738 },
739 },
740 },
741 },
742 },
743 },
744 },
745 },
746 },
747 },
748 },
749 },
750 },
751 },
752 },
753 },
754 },
755 },
756 },
757 },
758 },
759 },
760 },
761 },
762 },
763 },
764 },
765 },
766 },
767 },
768 },
769 },
770 },
771 },
772 },
773 },
774 },
775 },
776 },
777 },
778 },
779 },
780 },
781 },
782 },
783 },
784 },
785 },
786 },
787 },
788 },
789 },
790 },
791 },
792 },
793 },
794 },
795 },
796 },
797 },
798 },
799 },
800 },
801 },
802 },
803 },
804 },
805 },
806 },
807 },
808 },
809 },
810 },
811 },
812 },
813 },
814 },
815 },
816 },
817 },
818 },
819 },
820 },
821 },
822 },
823 },
824 },
825 },
826 },
827 },
828 },
829 },
830 },
831 },
832 },
833 },
834 },
835 },
836 },
837 },
838 },
839 },
840 },
841 },
842 },
843 },
844 },
845 },
846 },
847 },
848 },
849 },
850 },
851 },
852 },
853 },
854 },
855 },
856 },
857 },
858 },
859 },
860 },
861 },
862 },
863 },
864 },
865 },
866 },
867 },
868 },
869 },
870 },
871 },
872 },
873 },
874 },
875 },
876 },
877 },
878 },
879 },
880 },
881 },
882 },
883 },
884 },
885 },
886 },
887 },
888 },
889 },
890 },
891 },
892 },
893 },
894 },
895 },
896 },
897 },
898 },
899 },
900 },
901 },
902 },
903 },
904 },
905 },
906 },
907 },
908 },
909 },
910 },
911 },
912 },
913 },
914 },
915 },
916 },
917 },
918 },
919 },
920 },
921 },
922 },
923 },
924 },
925 },
926 },
927 },
928 },
929 },
930 },
931 },
932 },
933 },
934 },
935 },
936 },
937 },
938 },
939 },
940 },
941 },
942 },
943 },
944 },
945 },
946 },
947 },
948 },
949 },
950 },
951 },
952 },
953 },
954 },
955 },
956 },
957 },
958 },
959 },
960 },
961 },
962 },
963 },
964 },
965 },
966 },
967 },
968 },
969 },
970 },
971 },
972 },
973 },
974 },
975 },
976 },
977 },
978 },
979 },
980 },
981 },
982 },
983 },
984 },
985 },
986 },
987 },
988 },
989 },
990 },
991 },
992 },
993 },
994 },
995 },
996 },
997 },
998 },
999 },
1000 }
```

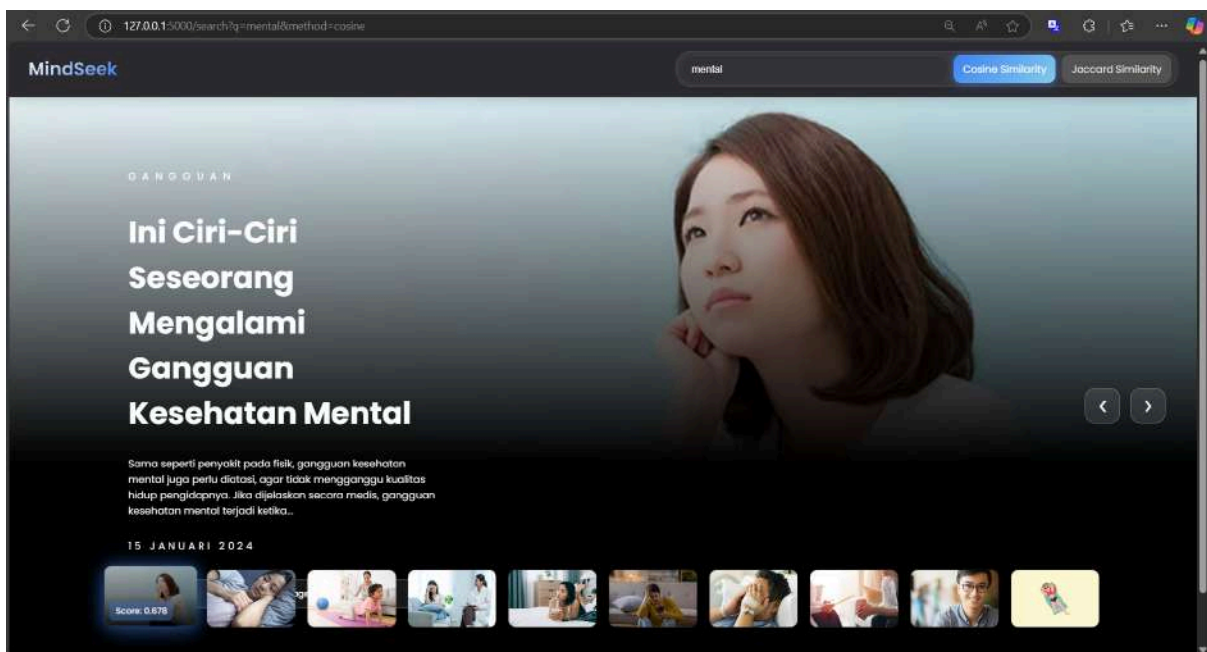
3.5 Tampilan Akhir dari website

3.5.1 Tampilan Halaman Pencarian



3.5.1 Tampilan Halaman Hasil Pencarian

Hasil cosine similarity pada query “mental”



Hasil jaccard similarity pada query “mental”

127.0.0.1:5000/search?q=mental&method=jaccard

MindSeek

mental

Cosine Similarity

Jaccard Similarity

G A N G U A N

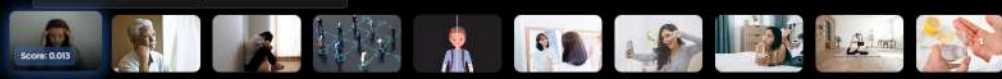
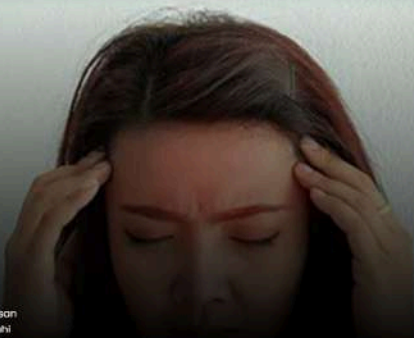
15 Gejala yang Timbul dari Gangguan Kecemasan

Gejala yang muncul pada pengidap gangguan kecemasan sangat beragam. Gejala ini umumnya akan memengaruhi kondisi kesehatan pengidapnya, baik secara fisik maupun psikis. Gejala yang muncul dapat meliputi...

13 FEBRUARI 2021

Go to Page

Score: 0.013



KESIMPULAN

Proyek ini bertujuan untuk menerapkan keseluruhan konsep dalam perkuliahan penelusuran informasi untuk pembuatan mesin pencari yang cepat, akurat, dan relevan di era digital, di mana hasil pencarian yang tidak efisien. Sistem penelusuran informasi yang dirancang mencakup web crawling, pembuatan indeks, pembobotan dokumen, dan tampilan hasil pencarian melalui antarmuka pengguna grafis (GUI). Dua metode pembobotan, Cosine Similarity dan Jaccard Similarity, digunakan untuk menentukan relevansi dokumen, memungkinkan pengguna untuk membandingkan hasil pencarian berdasarkan keduanya dan memilih yang paling relevan.

Dalam pelaksanaannya, proyek ini melibatkan berbagai tahapan, mulai dari pengumpulan data menggunakan teknik web scraping hingga pengolahan data melalui tokenisasi, stemming, dan perhitungan TF-IDF. Data yang telah diproses kemudian digunakan untuk membangun sistem backend berbasis Flask yang diintegrasikan dengan frontend interaktif.

Hasilnya, sistem ini memungkinkan pengguna untuk dengan mudah mencari dan menemukan informasi yang relevan dengan cepat melalui tampilan 10 dokumen teratas yang intuitif. Proyek ini memberikan solusi inovatif untuk meningkatkan pengalaman pencarian informasi berbasis topik di web. Selain itu, penerapan kedua metode pembobotan juga menawarkan wawasan berharga dalam pengembangan sistem pencarian yang lebih efektif.

REFERENSI

Scrapy (2024) *Scrapy Documentation*. Available at: <https://docs.scrapy.org/en/latest/> (Accessed: 24 November 2024).

Yusuf, M. (2020) *TF-IDF (Term Frequency-Inverse Document Frequency): Representasi Vector Data Text*. Available at: <https://yunusmuhammad007.medium.com/tf-idf-term-frequency-inverse-document-frequency-representasi-vector-data-text-2a4eff56cda> (Accessed: 26 November 2024).

Python Software Foundation (2024) *Python Documentation*. Available at: <https://docs.python.org/3/> (Accessed: 23 November 2024).

Pallets (2010) *Flask Documentation*. Available at: <https://flask.palletsprojects.com/en/stable/> (Accessed: 25 November 2024).

LAMPIRAN

1. Source Code

a. Setingan Spider (setting.py)

```
1 # Scrapy settings for mycrawler project
2 #
3 # For simplicity, this file contains only settings considered important or
4 # commonly used. You can find more settings consulting the documentation:
5 #
6 #     https://docs.scrapy.org/en/latest/topics/settings.html
7 #     https://docs.scrapy.org/en/latest/topics/downloader-middleware.html
8 #     https://docs.scrapy.org/en/latest/topics/spider-middleware.html
9
10 BOT_NAME = "mycrawler"
11
12 SPIDER_MODULES = ["mycrawler.spiders"]
13 NEWSPIDER_MODULE = "mycrawler.spiders"
14
15
16 # Crawl responsibly by identifying yourself (and your website) on the user-agent
17 #USER_AGENT = "mycrawler (+http://www.yourdomain.com)"
18
19 # Obey robots.txt rules
20 ROBOTSTXT_OBEY = True
21
22 # Configure maximum concurrent requests performed by Scrapy (default: 16)
23 #CONCURRENT_REQUESTS = 32
24
25 # Configure a delay for requests for the same website (default: 0)
26 # See https://docs.scrapy.org/en/latest/topics/settings.html#download-delay
27 # See also autothrottle settings and docs
28 #DOWNLOAD_DELAY = 3
29 # The download delay setting will honor only one of:
30 #CONCURRENT_REQUESTS_PER_DOMAIN = 16
31 #CONCURRENT_REQUESTS_PER_IP = 16
32
33 # Disable cookies (enabled by default)
34 #COOKIES_ENABLED = False
35
36 # Disable Telnet Console (enabled by default)
37 #TELNETCONSOLE_ENABLED = False
38
39 # Override the default request headers:
40 #DEFAULT_REQUEST_HEADERS = {
41 #     "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8",
42 #     "Accept-Language": "en",
43 # }
44
45 # Enable or disable spider middlewares
46 # See https://docs.scrapy.org/en/latest/topics/spider-middleware.html
47 #SPIDER_MIDDLEWARES = {
48 #     "mycrawler.middlewares.MycrawlerSpiderMiddleware": 543,
49 # }
50
51 # Enable or disable downloader middlewares
52 # See https://docs.scrapy.org/en/latest/topics/downloader-middleware.html
53 #DOWNLOADER_MIDDLEWARES = {
54 #     "mycrawler.middlewares.MycrawlerDownloaderMiddleware": 543,
55 # }
56
57 # Enable or disable extensions
58 # See https://docs.scrapy.org/en/latest/topics/extensions.html
59 #EXTENSIONS = {
60 #     "scrapy.extensions.telnet.TelnetConsole": None,
61 # }
62
63 # Configure item pipelines
64 # See https://docs.scrapy.org/en/latest/topics/item-pipeline.html
65 #ITEM_PIPELINES = {
66 #     "mycrawler.pipelines.MycrawlerPipeline": 300,
67 # }
68
69 # Enable and configure the Autothrottle extension (disabled by default)
70 # See https://docs.scrapy.org/en/latest/topics/autothrottle.html
71 #AUTOTHROTTLE_ENABLED = True
72 # The initial download delay
73 #AUTOTHROTTLE_START_DELAY = 5
74 # The maximum download delay to be set in case of high latencies
75 #AUTOTHROTTLE_MAX_DELAY = 60
76 # The average number of requests Scrapy should be sending in parallel to
77 # each remote server
78 #AUTOTHROTTLE_TARGET_CONCURRENCY = 1.0
79 # Enable showing throttling stats for every response received:
80 #AUTOTHROTTLE_DEBUG = False
81
82 # Enable and configure HTTP caching (disabled by default)
83 # See https://docs.scrapy.org/en/latest/topics/downloader-middleware.html#httpcache-middleware-settings
84 #HTTPCACHE_ENABLED = True
85 #HTTPCACHE_EXPIRATION_SECS = 0
86 #HTTPCACHE_DIR = "httpcache"
87 #HTTPCACHE_IGNORE_HTTP_CODES = []
88 #HTTPCACHE_STORAGE = "scrapy.extensions.httpcache.FilesystemCacheStorage"
89
90 # Set settings whose default value is deprecated to a future-proof value
91 REQUEST_FINGERPRINTER_IMPLEMENTATION = "2.7"
92 TWISTED_REACTOR = "twisted.internet.asyncioreactor.AsyncioSelectorReactor"
93 FEED_EXPORT_ENCODING = "utf-8"
94 CLOSESPIDER_PAGECOUNT = 2000
95 # settings.py
96
97 # Batas waktu antar request
98 DOWNLOAD_DELAY = 0.2 # 0.2 detik (setara dengan 5 request per detik)
99
100 # Batas maksimum permintaan secara bersamaan
101 CONCURRENT_REQUESTS = 5 # Sesuaikan dengan kebutuhan Anda
102
103 # Pastikan spider mematuhi aturan robots.txt (opsional)
104 ROBOTSTXT_OBEY = True
```


File `settings.py` pada proyek **Scrapy** ini berfungsi untuk mengonfigurasi berbagai pengaturan penting terkait cara kerja crawler dalam mengambil data dari situs web. Di awal, **BOT_NAME** diatur sebagai "mycrawler", yang menjadi identitas bot saat melakukan crawling, meskipun pengaturan `USER_AGENT` yang lebih spesifik bisa diaktifkan jika diperlukan. **SPIDER_MODULES** dan **NEWSPIDER_MODULE** menentukan lokasi modul spider yang digunakan untuk mendefinisikan logika crawling dalam proyek ini.

Pengaturan **ROBOTSTXT_OBEY** diatur ke `True` untuk memastikan bahwa crawler mematuhi aturan yang terdapat pada file `robots.txt` situs web yang dituju, yang memberi tahu crawler tentang bagian-bagian situs yang boleh atau tidak boleh diakses. **DOWNLOAD_DELAY** ditetapkan menjadi 0.2 detik, yang memberikan jeda antar permintaan untuk menghindari pengiriman permintaan secara berlebihan yang bisa membebani server. **CONCURRENT_REQUESTS** dibatasi hingga 5, memastikan bahwa hanya lima permintaan yang dapat diproses secara bersamaan, guna mencegah pemblokiran IP oleh server yang mungkin sensitif terhadap jumlah permintaan yang besar.

Selain itu, pengaturan **CLOSESPIDER_PAGECOUNT** mengatur batas maksimal jumlah halaman yang dapat di-crawl, dalam hal ini 2000 halaman, untuk menghentikan crawling setelah mencapai jumlah halaman yang ditentukan. **REQUEST_FINGERPRINTER_IMPLEMENTATION** diatur ke versi terbaru untuk memastikan bahwa sidik jari (fingerprint) untuk setiap permintaan dapat dibangkitkan secara unik, sementara **TWISTED_REACTOR** diatur ke `AsyncioSelectorReactor`, yang kompatibel dengan pustaka *asyncio*, memungkinkan pengelolaan permintaan asinkron secara lebih efisien.

Beberapa pengaturan lainnya yang dikomentari, seperti **AUTOTHROTTLE_ENABLED**, **COOKIES_ENABLED**, dan pengaturan untuk **downloader middlewares**, memungkinkan untuk diaktifkan sesuai kebutuhan. Sebagai contoh, **AUTOTHROTTLE** bisa digunakan untuk menyesuaikan kecepatan crawling berdasarkan latensi server, sementara **COOKIES_ENABLED** bisa mempengaruhi bagaimana crawler menangani sesi atau status login. Pengaturan **FEED_EXPORT_ENCODING** memastikan bahwa data yang diekspor menggunakan encoding UTF-8, yang mendukung karakter non-ASCII dan memastikan data dapat dibaca dengan baik di berbagai sistem.

Dengan pengaturan ini, **Scrapy** dapat melakukan crawling dengan cara yang lebih terkontrol dan efisien, mematuhi aturan situs, serta menghindari masalah seperti pemblokiran IP atau pengiriman

permintaan yang berlebihan. Pengaturan-pengaturan ini memberikan fleksibilitas dalam menyesuaikan kinerja crawling sesuai dengan kebutuhan dan kondisi yang ada.


b. Crawling dan Scraping (full_spider.py)

i. import library



Dalam implementasi proyek ini, beberapa pustaka Python digunakan untuk mempermudah proses pengolahan data dan pengambilan informasi dari sumber web. **Scrapy** merupakan framework yang digunakan untuk melakukan web crawling, memungkinkan pengambilan data dari situs web secara otomatis dan efisien. Pustaka **re** digunakan untuk manipulasi teks dengan ekspresi reguler (regex), yang sangat berguna untuk mengekstraksi pola tertentu dalam teks, seperti URL atau elemen-elemen lain yang relevan. Selain itu, pustaka **json** digunakan untuk membaca dan menulis file JSON, yang merupakan format data yang umum digunakan untuk menyimpan informasi dalam struktur yang mudah dibaca dan diproses. Pustaka **os** memfasilitasi operasi file dan direktori, seperti pembuatan folder atau pengelolaan file yang diperlukan dalam proses crawling. Terakhir, **urljoin** digunakan untuk menggabungkan URL relatif dengan URL dasar, sehingga memastikan bahwa setiap URL yang ditemukan selama crawling dapat diakses dengan benar meskipun tidak seluruhnya berada pada jalur absolut. Semua pustaka ini bekerja bersama-sama untuk mendukung kelancaran proses pengambilan dan pengolahan data dari situs web.

ii. Kelas Spider




```

1 class HealthArticleSpider(scrapy.Spider):
2     name = "full_health_article_spider"
3     start_urls = ["https://www.halodoc.com/kesehatan/kesehatan-mental"]
4     link_count = 0
5     max_links = 2000

```

Kelas **spider** mendefinisikan *HealthArticleSpider*, yang berfungsi untuk melakukan proses web crawling pada situs yang dituju. Spider ini disusun dengan beberapa komponen penting yang mendukung fungsinya. **name** adalah atribut yang menentukan nama unik untuk spider, yang memudahkan dalam mengidentifikasi dan menjalankan spider tersebut dalam proyek Scrapy. **start_urls** adalah daftar URL awal yang menjadi titik awal untuk proses crawling, di mana spider akan mulai menjelajahi situs web yang telah ditentukan. Di dalam spider ini, terdapat pula dua atribut penting lainnya, yaitu **link_count** dan **max_links**. **link_count** berfungsi untuk menghitung jumlah tautan yang telah dicrawl, sementara **max_links** digunakan untuk membatasi jumlah maksimum tautan yang akan diproses oleh spider. Dengan adanya pengaturan ini, proses crawling dapat dikendalikan secara efisien, menghindari eksplorasi berlebihan yang tidak diperlukan. Semua komponen ini bekerja sama untuk memastikan bahwa spider melakukan crawling dengan tujuan yang terstruktur dan terarah.

iii. Konfigurasi Crawler



```

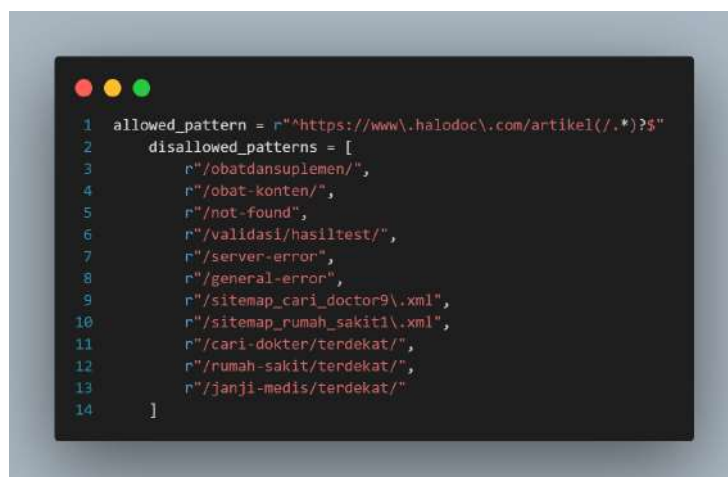
1 custom_settings = {
2     'DOWNLOAD_DELAY': 0.2,
3     'CONCURRENT_REQUESTS': 5,
4     'ROBOTSTXT_OBEY': True,
5     'LOG_LEVEL': 'INFO'
6 }

```

Konfigurasi **Crawler** di Scrapy memainkan peran penting dalam mengatur cara spider berinteraksi dengan situs web yang sedang dicrawl. Salah satu pengaturan utama adalah **DOWNLOAD_DELAY**, yang menunda setiap permintaan yang dikirimkan oleh spider selama 0.2 detik. Hal ini bertujuan untuk mengurangi beban pada server yang

sedang diakses dan menghindari pengiriman permintaan yang terlalu cepat, yang dapat dianggap sebagai serangan oleh server. Selanjutnya, **CONCURRENT_REQUESTS** membatasi jumlah permintaan yang dapat diproses secara bersamaan, yang dalam hal ini diatur maksimal sebanyak 5 permintaan. Pengaturan ini membantu mengelola trafik dan memastikan server tidak terbebani oleh permintaan yang datang dalam jumlah besar dalam waktu singkat. **ROBOTSTXT_OBEY** adalah pengaturan yang memastikan spider mematuhi aturan yang tertera di file **robots.txt** situs web, yang menginformasikan halaman-halaman yang boleh dan tidak boleh di-crawl oleh bot. Terakhir, **LOG_LEVEL** mengatur level log yang ditampilkan oleh Scrapy, di mana pengaturan ini diatur pada **INFO** atau lebih tinggi, sehingga hanya pesan log yang relevan dan penting yang akan ditampilkan, membantu pengembang dalam memantau dan menganalisis proses crawling yang sedang berjalan. Semua konfigurasi ini bekerja sama untuk memastikan bahwa crawling dilakukan secara efisien, etis, dan terkontrol.

iv. Pola URL



```
1 allowed_pattern = r"^https://www\.halodoc\.com/artikel/(.*)?$"
2 disallowed_patterns = [
3     r"/obatdansuplemen/",
4     r"/obat-konten/",
5     r"/not-found",
6     r"/validasi/hasilttest/",
7     r"/server-error",
8     r"/general-error",
9     r"/sitemap_cari_dokter9\.xml",
10    r"/sitemap_rumah_sakit1\.xml",
11    r"/cari-dokter/terdekat/",
12    r"/rumah-sakit/terdekat/",
13    r"/janji-medis/terdekat/"
14 ]
```

Pola URL merupakan bagian penting dalam mengatur bagaimana spider berinteraksi dengan situs web, terutama dalam menentukan halaman-halaman yang akan diakses dan yang harus dihindari. Pengaturan **allowed_pattern** digunakan untuk menentukan pola URL yang diizinkan untuk diakses oleh spider. Hanya URL yang sesuai dengan pola ini yang akan dijelajahi lebih lanjut, memungkinkan kontrol yang lebih spesifik terhadap halaman-halaman yang relevan dengan tujuan crawling. Sebaliknya, **disallowed_patterns** mengatur pola URL yang harus dihindari oleh spider, biasanya karena mengarah ke halaman yang tidak relevan atau tidak diinginkan, seperti halaman login, halaman dengan konten dinamis yang tidak diperlukan, atau halaman yang sudah diproses sebelumnya. Dengan adanya pengaturan ini, proses crawling dapat difokuskan pada halaman-halaman yang penting dan menghindari tautan-tautan yang tidak berguna, sehingga meningkatkan efisiensi dan relevansi data yang diambil.

v. Inisial Spider

```
1 def __init__(self, *args, **kwargs):
2     super(HealthArticleSpider, self).__init__(*args, **kwargs)
3     os.makedirs(os.path.dirname(self.output_path), exist_ok=True)
4     if not os.path.exists(self.output_path):
5         with open(self.output_path, 'w', encoding='utf-8') as f:
6             json.dump([], f, ensure_ascii=False)
```

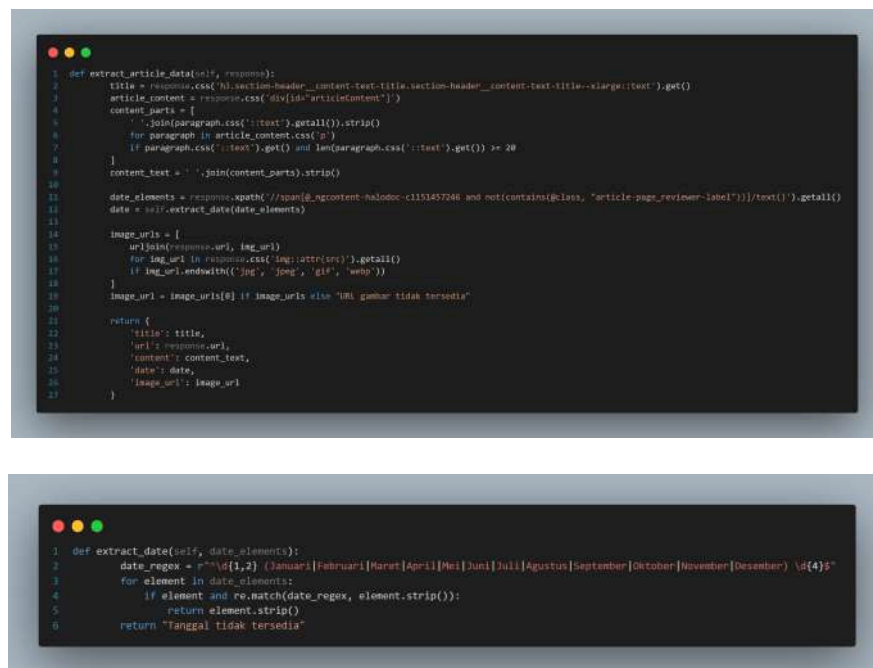
Inisial Spider mengacu pada langkah-langkah awal yang dilakukan untuk mempersiapkan lingkungan kerja spider sebelum melakukan crawling. Salah satu langkah pertama adalah **membuat direktori** untuk menyimpan data hasil crawling jika direktori tersebut belum ada. Direktori ini akan berfungsi sebagai tempat untuk menyimpan berbagai file yang dihasilkan oleh spider, seperti file artikel yang diambil dari situs web. Dengan memastikan bahwa direktori sudah tersedia, data yang diambil dapat tersimpan dengan rapi dan terorganisir. Selain itu, proses **inisialisasi file JSON** juga dilakukan untuk menyimpan artikel-artikel yang berhasil dicrawl. Format JSON dipilih karena kemudahan dalam membaca dan memproses data dalam struktur key-value, yang sangat sesuai untuk menyimpan artikel yang berisi berbagai informasi seperti judul, tanggal, penulis, dan konten artikel. Inisialisasi ini memastikan bahwa data hasil crawling dapat tersimpan dengan baik dan siap untuk diproses lebih lanjut, baik untuk analisis atau pemrosesan lainnya.

vi. Parse Halaman

```
1 def parse(self, response):
2     if self.link_count >= self.max_links:
3         return
4
5     if response.url in self.visited_links:
6         return
7
8     self.visited_links.add(response.url)
9     article_data = self.extract_article_data(response)
10
11     if article_data['title'] and article_data['content']:
12         word_count = len(article_data['content'].split())
13
14         if word_count >= 200 and not self.is_duplicate_title(article_data['title']):
15             self.save_article_data(article_data)
16             self.link_count += 1
17             self.logger.info(f"Berhasil mengekstrak artikel {self.link_count}: {article_data['title']}")
18         else:
19             self.logger.info(f"Konten tidak valid atau duplikat: {article_data['title']}")
20
21     for link in response.css('a::attr(href)').getall():
22         absolute_link = response.urljoin(link)
23         if self.is_valid_link(absolute_link):
24             yield scrapy.Request(
25                 absolute_link,
26                 callback=self.parse,
27                 errback=self.handle_error
28             )
```

Parse Halaman adalah tahap di mana spider mengekstrak informasi atau data yang relevan dari halaman yang telah dicrawl. Proses ini melibatkan pengambilan elemen-elemen tertentu dari halaman web, seperti judul artikel, konten, penulis, atau metadata lainnya, yang kemudian disimpan dalam format yang sesuai, seperti file JSON. Selama proses ini, spider juga perlu **memfilter tautan baru** untuk memastikan bahwa hanya tautan yang valid dan relevan yang akan diproses lebih lanjut. Fungsi **is_valid_link** digunakan untuk memeriksa apakah tautan yang ditemukan memenuhi kriteria yang telah ditentukan, seperti berada dalam domain yang diizinkan atau mengarah ke halaman yang relevan dengan topik yang diinginkan. Setelah itu, spider akan **mengirim permintaan baru** untuk setiap tautan yang dianggap valid. Ini memastikan bahwa spider dapat menjelajahi halaman-halaman berikutnya dan terus mengumpulkan data yang diperlukan, sambil menjaga efisiensi dan relevansi dalam proses crawling. Dengan demikian, tahap ini memungkinkan spider untuk secara dinamis menavigasi situs dan mengumpulkan informasi dari berbagai halaman yang terkait.

vii. Ekstraksi Data Artikel



```

1 def extract_article_data(self, response):
2     title = response.css('h1.section-header::content-text::title, h2.section-header::content-text::title::xlarge::text').get()
3     article_content = response.css('div[id="articleContent"]')
4     content_parts = []
5     for paragraph in article_content.css('p'):
6         paragraph_text = paragraph.css('::text').get().strip()
7         if paragraph.css('::text').get() and len(paragraph.css('::text').get()) >= 20:
8             content_parts.append(paragraph_text)
9     content_text = ' '.join(content_parts).strip()
10
11 date_elements = response.xpath('//span[@ngcontent=halodoc-c153457246 and not(contains(@class, "article-page-reviewer-label"))]/text()').getall()
12 date = self.extract_date(date_elements)
13
14 image_urls = []
15 url = response.url
16 for img_url in response.css('img::attr(src)').getall():
17     if img_url.endswith(('.jpg', '.png', '.gif', '.webp')):
18         image_urls.append(url + img_url)
19 image_url = image_urls[0] if image_urls else "URL gambar tidak tersedia"
20
21 return {
22     'title': title,
23     'url': response.url,
24     'content': content_text,
25     'date': date,
26     'image_url': image_url
27 }

```

```

1 def extract_date(self, date_elements):
2     date_regex = r"^(?!(1,2)) (Januari|Februari|Marat|April|Mei|Juni|Juli|Agustus|September|Oktober|November|Desember) \d{4}$"
3     for element in date_elements:
4         if element and re.match(date_regex, element.strip()):
5             return element.strip()
6     return "Tanggal tidak tersedia"


```

Ekstraksi Data Artikel adalah tahap penting dalam proses crawling di mana spider mengumpulkan informasi yang relevan dari halaman artikel yang dicrawl. Pada tahap ini, spider **mengekstrak elemen-elemen penting** dari artikel, seperti **judul**, **konten**, **tanggal publikasi**, dan **gambar**. Data ini sangat penting untuk memberikan gambaran lengkap tentang artikel yang diambil.

Untuk **membersihkan teks konten**, digunakan teknik seperti **CSS selector**, yang memungkinkan spider untuk memilih dan mengekstrak hanya bagian teks yang relevan dari halaman, seperti paragraf atau artikel utama, dan mengabaikan elemen-elemen lain yang tidak diperlukan, seperti iklan atau menu navigasi. Dengan menggunakan selector ini, spider dapat dengan efisien mengidentifikasi dan mengumpulkan konten yang diinginkan dari struktur HTML halaman.

Selain itu, dalam hal gambar, spider perlu **menggabungkan URL gambar relatif** dengan **URL dasar** untuk memastikan bahwa gambar dapat diakses dengan benar, bahkan jika URL gambar yang ditemukan dalam halaman tersebut hanya relatif. Proses penggabungan URL ini memungkinkan spider untuk membentuk URL lengkap yang dapat digunakan untuk mengunduh atau menampilkan gambar dengan tepat. Semua data yang diekstraksi, termasuk judul, konten, tanggal, dan gambar, kemudian disimpan dalam format yang sesuai, seperti JSON, untuk proses analisis atau penggunaan lebih lanjut.

viii. Validasi URL



```
1 def is_valid_link(self, url):
2     if not re.match(self.allowed_pattern, url):
3         return False
4     if url in self.visited_links:
5         return False
6     if any(re.search(pattern, url) for pattern in self.disallowed_patterns):
7         return False
8     return True
9
```

Validasi URL adalah langkah penting dalam proses crawling untuk memastikan bahwa spider hanya mengunjungi halaman-halaman yang relevan dan sesuai dengan kebijakan yang telah ditetapkan. Pada tahap ini, **memastikan URL sesuai dengan pola yang diizinkan** berarti mengecek apakah URL yang ditemukan selama crawling memenuhi kriteria yang telah ditentukan sebelumnya, seperti domain yang diizinkan atau pola path tertentu. Dengan melakukan validasi ini, spider dapat menghindari mengunjungi halaman yang tidak relevan atau tidak sesuai dengan tujuan crawling.

Selain itu, dalam validasi URL, penting untuk **menghindari URL yang sudah dikunjungi**. Untuk itu, spider perlu memeriksa apakah URL yang ditemukan sebelumnya sudah pernah diproses. Hal ini mencegah spider mengunjungi halaman yang sama lebih dari sekali, yang bisa mengakibatkan pengambilan data yang duplikat atau pemborosan sumber daya.

Selain itu, URL juga perlu dibandingkan dengan **pola yang dilarang**. Pola URL yang dilarang biasanya mencakup halaman yang tidak

relevan, seperti halaman login, halaman error, atau halaman yang sudah diproses sebelumnya. Dengan menerapkan validasi ini, spider dapat memastikan bahwa hanya URL yang sesuai dengan kriteria yang diperbolehkan untuk diproses lebih lanjut, menjaga proses crawling tetap efisien dan terarah.

ix. Deteksi Judul



```
1 def is_duplicate_title(self, title):
2     """Cek apakah judul sudah ada di file JSON"""
3     try:
4         with open(self.output_path, 'r', encoding='utf-8') as f:
5             articles = json.load(f)
6             return any(article['title'] == title for article in articles)
7     except Exception as e:
8         self.logger.error(f"Error membaca file JSON: {e}")
9         return False
```

Deteksi Duplikasi Judul berfungsi untuk memastikan bahwa artikel dengan **judul yang sama** tidak disimpan lebih dari sekali dalam file JSON. Proses ini dilakukan dengan memeriksa apakah judul artikel yang baru ditemukan sudah ada dalam data yang telah diambil sebelumnya. Jika artikel dengan judul tersebut sudah ada, maka artikel tersebut tidak akan disimpan lagi, sehingga menghindari duplikasi data.

x. Menyimpan Data



```
1 def save_article_data(self, data):
2     try:
3         with open(self.output_path, 'r+', encoding='utf-8') as f:
4             articles = json.load(f)
5             articles.append(data)
6             f.seek(0)
7             json.dump(articles, f, indent=4, ensure_ascii=False)
8             f.truncate()
9     except Exception as e:
10        self.logger.error(f"Error saat menyimpan data: {e}")
```

Menyimpan Data dilakukan dengan **menambahkan data artikel baru** ke dalam file JSON tanpa menimpa data yang sudah ada. Proses ini memastikan bahwa setiap artikel yang berhasil dicrawl dan diekstraksi disimpan secara terpisah, menjaga data yang sudah ada tetap utuh. Dengan cara ini, file JSON akan terus bertambah seiring dengan

penambahan artikel baru, memungkinkan pengumpulan data yang lebih besar dan lengkap.

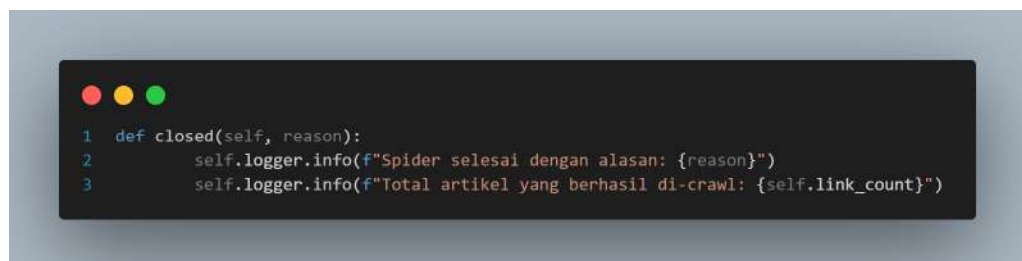
xi. Mengangani Error



```
1 def handle_error(self, failure):
2     self.logger.error(f"Request gagal: {failure.request.url}")
3     self.logger.error(f"Error: {str(failure.value)}")
```

Menangani Error berfungsi untuk **mengelola kesalahan** yang terjadi saat spider mencoba mengakses halaman tertentu. Jika terjadi masalah, seperti halaman tidak dapat diakses atau terjadi kesalahan koneksi, spider akan menangani error tersebut dengan cara yang terstruktur. Kesalahan yang terjadi akan dicatat dalam log, yang memungkinkan pengembang untuk memantau masalah yang muncul dan memperbaikinya jika diperlukan, tanpa menghentikan seluruh proses crawling.

xii. Menutup Spider



```
1 def closed(self, reason):
2     self.logger.info(f"Spider selesai dengan alasan: {reason}")
3     self.logger.info(f"Total artikel yang berhasil di-crawl: {self.link_count}")
```

Menutup Spider dilakukan dengan menampilkan **log** yang memberikan informasi tentang **alasan berhenti** spider dan **jumlah artikel** yang telah berhasil dicrawl. Setelah proses crawling selesai, log ini memberikan gambaran tentang kinerja spider, apakah berhenti karena mencapai batas yang ditentukan, tidak ada lagi tautan yang perlu diproses, atau alasan lainnya. Selain itu, log ini juga mencatat jumlah artikel yang berhasil diambil, memberikan feedback yang berguna bagi pengembang untuk mengevaluasi hasil dan mengidentifikasi potensi perbaikan di masa mendatang.

c. Preprocessing dan Perhitungan TF-IDF

a. Import Library dan Konfigurasi Logging

```

1 import json
2 import math
3 import logging
4 import gc
5 from pathlib import Path
6 from typing import Dict, List, Set, Union, Optional
7 from collections import defaultdict, Counter
8 from dataclasses import dataclass
9 from Sastrawi.Stemmer.StemmerFactory import StemmerFactory
10 import numpy as np
11 from datetime import datetime
12 from tqdm import tqdm # Import tqdm for progress bars
13
14 # Configure logging dengan level DEBUG
15 logging.basicConfig(
16     level=logging.DEBUG,
17     format='%(asctime)s - %(levelname)s - %(message)s',
18     handlers=[
19         logging.FileHandler(f'tfidf_processing_{datetime.now().strftime("%Y%m%d_%H%M%S")}.log'),
20         logging.StreamHandler()
21     ]
22 )
23 logger = logging.getLogger(__name__)

```

Program dimulai dengan mengimpor berbagai pustaka yang diperlukan, seperti math untuk perhitungan matematis, json untuk membaca dan menyimpan data dalam format JSON, serta logging untuk mencatat aktivitas dan kesalahan yang terjadi selama eksekusi program. Konfigurasi logging diatur untuk menyimpan log dengan informasi yang mencakup timestamp, level log, dan pesan kesalahan, yang akan disimpan dalam file log dan ditampilkan di terminal, sehingga memudahkan pemantauan jalannya program.

b. Kelas TF - IDF Config

```

1 class TFIDFConfig:
2     """Configuration class for TF-IDF processing parameters"""
3     min_df: int = 2 # Minimum document frequency
4     max_df: float = 0.85 # Maximum document frequency (as fraction)
5     min_word_length: int = 3 # Minimum word length
6     normalize: bool = True # Normalize vectors
7     use_idf: bool = True # Use IDF weighting
8     smooth_idf: bool = True # Smooth IDF weights
9     batch_size: int = 100 # Batch size for processing

```

Konfigurasi untuk parameter TF-IDF disimpan dalam sebuah kelas yang mengatur beberapa parameter penting, seperti batas minimum dan maksimum frekuensi kata di dokumen (min_df dan max_df), panjang minimum kata untuk diproses (min_word_length), serta apakah vektor hasil perhitungan perlu dinormalisasi (normalize). Selain itu, parameter ukuran batch (batch_size) juga diatur untuk mengontrol berapa banyak dokumen yang diproses dalam satu batch,

memungkinkan pemrosesan lebih efisien dan terstruktur saat menangani data dalam jumlah besar.

c. Kelas DocumentPreprocessor

```
1 class DocumentPreprocessor:
2     def __init__(self, config: TFIDFConfig):
3         self.config = config
4         self.stemmer = StemmerFactory().create_stemmer()
5         self._load_stop_words()
6         logger.debug("DocumentPreprocessor initialized")
7
8     def _load_stop_words(self) -> None:
9         """Load Indonesian stop words"""
10        self.stop_words = {
11            "yang", "di", "ke", "dari", "dan", "ini", "itu", "dengan", "untuk",
12            "atau", "pada", "adalah", "dalam", "oleh", "akan", "sebagai", "sangat",
13            "juga", "tidak", "karena", "bagi", "hanya", "kita", "saya", "kamu",
14            "kami", "kalian", "mereka", "beliau", "ada", "akan", "bisa", "bukan",
15            "harus", "apakah", "apa", "agar", "tersebut", "demikian", "tetapi",
16            "supaya", "sedangkan", "serta", "namun", "antara", "itu", "sebuah",
17            "setelah", "sekitar", "selalu", "sejak", "sedang", "masih", "pun",
18            "pernah", "maka", "seperti", "sampai", "tanpa", "yaitu", "selain",
19            "belum", "perlu", "kapan", "dimana", "bagaimana", "siapa", "dalam",
20            "demi", "jika", "jikalau", "manakala", "seusai", "sebelum", "sehabis",
21            "ketika", "dimana", "terus", "saja", "atau", "bahwa", "sebab", "lalu", "baik"
22        }
23        logger.debug("Stop words loaded")
24
25    def preprocess_text(self, text: str) -> List[str]:
26        try:
27            if not isinstance(text, str):
28                logger.warning(f"Expected string input, got {type(text)}")
29                return []
30
31            tokens = text.lower().split()
32            processed_tokens = [
33                self.stemmer.stem(word)
34                for word in tokens
35                if (word not in self.stop_words and
36                    len(word) >= self.config.min_word_length)
37            ]
38            return processed_tokens
39        except Exception as e:
40            logger.error(f"Error in text preprocessing: {str(e)}")
41            return []
```

Pada tahap praproses teks, pertama-tama dilakukan penghilangan **stop words**, yaitu kata-kata umum dalam Bahasa Indonesia yang tidak memberikan kontribusi signifikan terhadap makna, seperti "dan", "yang", dan "dengan". Selanjutnya, setiap kata yang tersisa di-*stem* menggunakan **Sastrawi Stemmer**, yang mengubah kata ke bentuk dasarnya, misalnya "berlari" menjadi "lari". Selain itu, kata-kata dengan panjang lebih pendek dari batas minimum yang ditentukan juga diabaikan, untuk menghindari pengolahan kata yang tidak relevan atau terlalu umum, seperti "di" atau "ke". Proses ini bertujuan untuk memurnikan teks dan meningkatkan kualitas analisis data lebih lanjut.

d. Kelas TFIDFProcessor

```

1 class TFIDFProcessor:
2     def __init__(self, config: TFIDFConfig):
3         self.config = config
4         self.preprocessor = DocumentPreprocessor(config)
5         self.reset_indices()
6         logger.debug("TFIDFProcessor initialized")
7
8     def reset_indices(self) -> None:
9         self.inverted_index = defaultdict(set)
10        self.tfidf_index = defaultdict(dict)
11        self.df = Counter()
12        self.vocabulary = {}
13        self.document_count = 0
14        logger.debug("Indices reset")
15
16    def _check_df_threshold(self, term: str, total_docs: int) -> bool:
17        """Check if term meets document frequency thresholds"""
18        if self.df[term] < self.config.min_df:
19            return False
20        if (self.df[term] / total_docs) > self.config.max_df:
21            return False
22        return True
23
24    def _compute_idf(self, term: str) -> float:
25        """Compute IDF for a term"""
26        if not self.config.use_idf:
27            return 1.0
28
29        n_samples = self.document_count
30        df = self.df[term]
31
32        if self.config.smooth_idf:
33            n_samples += 1
34            df += 1
35
36        return math.log(n_samples / df) + 1 if df else 0
37
38    def _process_batch(self, batch: List[Dict], is_first_pass: bool) -> None:
39        """Process a batch of documents"""
40        for doc in batch:
41            try:
42                doc_id = doc['url']
43                content = doc.get('content', '')
44
45                if not content:
46                    logger.warning(f"Empty content for document {doc_id}")
47                    continue
48
49                terms = self.preprocessor.preprocess_text(content)
50
51                if is_first_pass:
52                    # First pass: build document frequencies
53                    for term in set(terms):
54                        self.df[term] += 1
55                        self.inverted_index[term].add(doc_id)
56                else:
57                    # Second pass: compute TF-IDF
58                    term_counts = Counter(terms)
59                    doc_length = len(terms)
60
61                    if doc_length == 0:
62                        logger.warning(f"No valid terms found in document {doc_id}")
63                        continue
64
65                    doc_tfidf = {}
66                    for term, count in term_counts.items():
67                        if term in self.vocabulary:
68                            tf = count / doc_length
69                            idf = self._compute_idf(term)
70                            doc_tfidf[term] = tf * idf
71
72                    if self.config.normalize and doc_tfidf:
73                        norm = math.sqrt(sum(score * score for score in doc_tfidf.values()))
74                        if norm > 0:
75                            doc_tfidf = {term: score/norm for term, score in doc_tfidf.items()}
76
77                    self.tfidf_index[doc_id] = doc_tfidf
78
79            except Exception as e:
80                logger.error(f"Error processing document {doc.get('url', 'unknown')}: {str(e)}")
81                continue

```

```

1  def process_documents(self, documents: List[Dict]) -> None:
2      try:
3          total_docs = len(documents)
4          logger.info(f"Starting document processing for {total_docs} documents")
5          self.reset_indices()
6          self.document_count = total_docs
7
8          # Process in batches
9          batch_size = self.config.batch_size
10
11         # First pass: Build document frequencies
12         logger.info("Starting first pass - building document frequencies")
13         for i in tqdm(range(0, total_docs, batch_size), desc="First pass"):
14             batch = documents[i:i + batch_size]
15             self._process_batch(batch, is_first_pass=True)
16             gc.collect() # Force garbage collection
17
18         # Build vocabulary
19         logger.info("Building vocabulary")
20         self.vocabulary = {
21             term: idx for idx, term in enumerate(
22                 sorted(term for term in self.df
23                     if self._check_df_threshold(term, total_docs))
24             )
25         }
26         logger.info(f"Vocabulary size: {len(self.vocabulary)}")
27
28         # Second pass: Compute TF-IDF scores
29         logger.info("Starting second pass - computing TF-IDF scores")
30         for i in tqdm(range(0, total_docs, batch_size), desc="Second pass"):
31             batch = documents[i:i + batch_size]
32             self._process_batch(batch, is_first_pass=False)
33             gc.collect()
34
35         logger.info("Document processing completed")
36
37     except Exception as e:
38         logger.error(f"Error in document processing: {str(e)}")
39         raise
40
41 def save_indices(self, output_dir: Union[str, Path]) -> None:
42     """
43     Save indices to JSON files
44
45     Args:
46         output_dir: Directory to save the index files
47     """
48     try:
49         output_dir = Path(output_dir)
50         output_dir.mkdir(parents=True, exist_ok=True)
51
52         # Prepare inverted index for serialization
53         inverted_dict = {
54             term: list(docs)
55             for term, docs in self.inverted_index.items()
56             if term in self.vocabulary
57         }
58
59         # Save files
60         with open(output_dir / 'inverted_index.json', 'w', encoding='utf-8') as f:
61             json.dump(inverted_dict, f, ensure_ascii=False, indent=4)
62
63         with open(output_dir / 'tfidf_index.json', 'w', encoding='utf-8') as f:
64             json.dump(dict(self.tfidf_index), f, ensure_ascii=False, indent=4)
65
66         with open(output_dir / 'vocabulary.json', 'w', encoding='utf-8') as f:
67             json.dump(self.vocabulary, f, ensure_ascii=False, indent=4)
68
69         logger.info(f"Successfully saved indices to {output_dir}")
70
71     except Exception as e:
72         logger.error(f"Error saving indices: {str(e)}")
73         raise

```


Proses perhitungan nilai TF-IDF dimulai dengan **reset_indices**, yang menginisialisasi kembali struktur data untuk menyimpan hasil perhitungan. Langkah berikutnya adalah **_check_df_threshold**, yang memastikan bahwa setiap kata yang diproses memenuhi batas frekuensi dokumen yang ditentukan, baik untuk minimum maupun maksimum frekuensi. Selanjutnya, **_compute_idf** menghitung nilai *Inverse Document Frequency* (IDF) untuk setiap kata berdasarkan distribusinya di seluruh dokumen. **_process_batch** digunakan untuk memproses dokumen dalam batch, baik untuk membangun frekuensi kata pada *pass pertama* atau menghitung skor TF-IDF pada *pass kedua*. Pada tahap **process_documents**, dua *pass* utama dilakukan: pertama untuk membangun frekuensi kata (document frequency) dan kedua untuk menghitung skor TF-IDF untuk setiap kata dalam dokumen. Akhirnya, **save_indices** menyimpan hasil perhitungan dalam format JSON, yang mencakup *inverted index*, skor TF-IDF untuk setiap dokumen, dan kosakata yang digunakan. Langkah-langkah ini memastikan bahwa data yang dihasilkan terstruktur dengan baik dan siap digunakan dalam pencarian atau analisis lebih lanjut.

e. Fungsi main

```

1  def main():
2      try:
3          logger.info("Starting TF-IDF processing")
4
5          # Load data
6          file_path = r'data/hasil_crawl.json'
7          logger.info(f"Loading data from {file_path}")
8
9          with open(file_path, 'r', encoding='utf-8') as file:
10             data = json.load(file)
11
12             logger.info(f"Loaded {len(data)} documents")
13
14             # Initialize processor with config
15             config = TFIDFConfig(
16                 min_df=2,
17                 max_df=0.85,
18                 min_word_length=3,
19                 normalize=True,
20                 use_idf=True,
21                 smooth_idf=True,
22                 batch_size=100 # Process 100 documents at a time
23             )
24
25             processor = TFIDFProcessor(config)
26
27             # Process documents
28             processor.process_documents(data)
29
30             # Save indices
31             output_dir = 'output_indices'
32             logger.info(f"Saving indices to {output_dir}")
33             processor.save_indices(output_dir)
34
35             logger.info("Processing completed successfully")
36
37     except FileNotFoundError:
38         logger.error(f"Could not find file: {file_path}")
39     except json.JSONDecodeError:
40         logger.error("Error decoding JSON file")
41     except Exception as e:
42         logger.error(f"Unexpected error: {str(e)}")
43         raise

```

Alur utama eksekusi program dimulai dengan **membaca data dokumen** yang tersimpan dalam file JSON, yang berisi informasi artikel yang telah di-crawl sebelumnya. Setelah itu, **processor**

- f. Output dari advance_tfidf.py
 - i. inverted_index.json

ii. tfidf index.json

- iii. `vocabulary.json`




a. Import dan setup

Pada proyek ini, **Flask** digunakan untuk membuat server web yang memungkinkan interaksi dengan pengguna melalui antarmuka berbasis web. **json** digunakan untuk memproses dan menyimpan data dalam format JSON, seperti dokumen yang telah di-crawl dan hasil perhitungan TF-IDF. Untuk perhitungan kemiripan antar dokumen, **sklearn** digunakan untuk menghitung **Cosine Similarity**, yang mengukur sejauh mana dua vektor teks (dokumen dan query) saling mendekati. **numpy** mendukung operasi numerik, seperti manipulasi array dan perhitungan matematis yang diperlukan untuk pengolahan vektor dalam penghitungan Cosine Similarity. **nltk** berfungsi untuk pengolahan teks, seperti tokenisasi kata, stemming, dan penghapusan **stopwords**, yang membantu dalam mempersiapkan data teks untuk analisis lebih lanjut. Terakhir, **re** digunakan untuk pengolahan teks berbasis ekspresi reguler, seperti membersihkan teks dari karakter

khusus atau angka yang tidak relevan, guna meningkatkan kualitas data sebelum diproses lebih lanjut. Kombinasi pustaka ini memungkinkan pemrosesan teks yang efisien dan akurat untuk sistem pencarian berbasis konten.

b. Setup NLTK Resource



```
1  try:
2      nltk.download('punkt')
3      nltk.download('stopwords')
4      nltk.download('averaged_perceptron_tagger')
5  except Exception as e:
6      print(f"Error downloading NLTK data: {e}")
7
8  app = Flask(__name__)
```

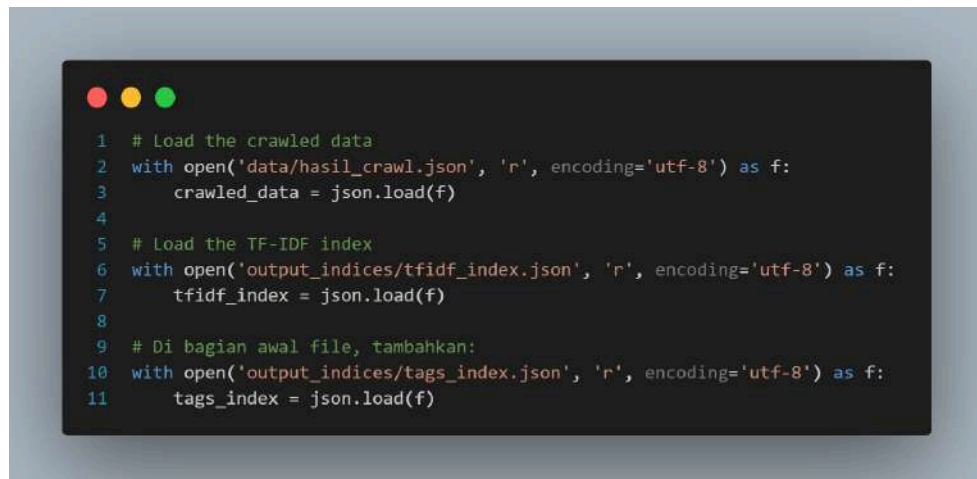
Pada tahap ini, **NLTK** (Natural Language Toolkit) digunakan untuk mengunduh beberapa sumber daya yang diperlukan untuk pengolahan teks, seperti **tokenizer**, **stopwords**, dan **tagger**. **Tokenizer** berfungsi untuk memecah teks menjadi kata-kata atau token, yang merupakan langkah penting dalam mempersiapkan teks untuk analisis lebih lanjut. **Stopwords** adalah daftar kata umum yang tidak memberikan kontribusi signifikan terhadap makna teks, seperti "dan", "atau", dan "yang", yang akan dihapus selama preprocessing. **Tagger** digunakan untuk menandai atau memberi label pada kata-kata dalam teks berdasarkan kategori gramatikalnya (seperti kata benda, kata kerja, dll.), yang bisa berguna untuk analisis sintaksis dan semantik. Dengan mengunduh dan mempersiapkan ketiga sumber daya ini, NLTK memungkinkan pemrosesan teks yang lebih efektif dalam tugas-tugas seperti stemming, tokenisasi, dan penghilangan stopwords.

c. Flask Application Initialization




Membuat aplikasi Flask.

d. Load data



Langkah pertama dalam aplikasi adalah **memuat data hasil crawling** dan **indeks TF-IDF**. File `hasil_crawl.json` berisi data yang diambil dari situs web menggunakan proses crawling, yang mencakup informasi seperti judul artikel, konten, URL, tanggal, dan gambar. File ini dimuat menggunakan pustaka **json**, yang memungkinkan data dibaca dan diubah menjadi struktur data Python (seperti list atau dictionary) untuk diproses lebih lanjut. Selain itu, file `tfidf_index.json` memuat hasil perhitungan TF-IDF, yang menyimpan vektor TF-IDF untuk setiap dokumen yang di-crawl. Indeks ini digunakan untuk menghitung relevansi antara query pencarian pengguna dengan dokumen yang ada. Kedua file ini dimuat saat aplikasi dijalankan, sehingga data yang diperlukan untuk pencarian dan analisis dapat digunakan langsung dalam proses pencarian dan perhitungan kesamaan. Dengan memuat kedua sumber data ini, aplikasi dapat dengan cepat memproses permintaan pencarian dan mengembalikan hasil yang relevan berdasarkan perhitungan TF-IDF.


- e. Fungsi Pendukung
 - i. Parse Tanggal



```
1 def parse_date(date_str):
2     if date_str == "Tanggal tidak tersedia":
3         return datetime.min
4     try:
5         return datetime.strptime(date_str, "%d %B %Y")
6     except:
7         return datetime.min
```

Pada tahap ini, **fungsi konversi tanggal** digunakan untuk mengubah string yang berisi tanggal menjadi objek **datetime** yang dapat diproses lebih lanjut. Fungsi ini berusaha untuk mengonversi tanggal yang diterima (dalam format string) ke dalam format datetime menggunakan fungsi `datetime.strptime()`, yang memerlukan format yang sesuai dengan input tanggal, seperti `"%d %B %Y"` untuk format `"25 Desember 2024"`. Jika konversi berhasil, fungsi akan mengembalikan objek `datetime` yang mewakili tanggal tersebut. Namun, jika terjadi kesalahan dalam konversi, seperti format yang tidak sesuai atau tanggal yang tidak valid, fungsi akan menangani kesalahan tersebut dan mengembalikan **tanggal minimum** (yang direpresentasikan sebagai `datetime.min`). Penggunaan tanggal minimum ini berguna untuk memastikan bahwa data tetap dapat diproses dengan lancar, meskipun ada tanggal yang tidak valid atau hilang, tanpa menyebabkan error dalam alur program.

- ii. Bersihkan Teks



```
1 def clean_text(text):
2     # Lowercase the text
3     text = text.lower()
4
5     # Remove special characters and digits
6     text = re.sub(r'^\w\s', '', text)
7     text = re.sub(r'\d+', '', text)
8
9     return text
```

Pada tahap ini, teks yang diproses diubah menjadi huruf kecil untuk memastikan konsistensi dan menghindari perbedaan hasil analisis antara huruf besar dan kecil. Proses ini dilakukan dengan menggunakan metode **.lower()** pada string, yang mengubah semua karakter dalam teks menjadi huruf kecil. Setelah itu, karakter khusus (seperti tanda baca, simbol, dan karakter non-alfabet) serta angka dihapus dari teks menggunakan ekspresi reguler yang diterapkan dengan pustaka **re**. Ekspresi reguler **re.sub(r'^\w\s', '', text)** digunakan untuk menghapus semua karakter yang bukan huruf atau spasi, sementara **re.sub(r'\d+', '', text)** menghilangkan angka. Dengan demikian, hanya kata-kata yang relevan dan dapat dianalisis yang tersisa dalam teks, meningkatkan kualitas dan efektivitas proses berikutnya dalam pengolahan data teks, seperti tokenisasi, stemming, dan perhitungan TF-IDF.

iii. Siapkan Data Artikel

```


1  # Modifikasi fungsi prepare_article_data:
2  def prepare_article_data(articles, page=1, per_page=9):
3      prepared_articles = []
4      for article in articles:
5          if article['title'] and article['content']:
6              image_url = article['image_url']
7              if image_url == "URL gambar tidak tersedia":
8                  image_url = "https://via.placeholder.com/400x200"
9
10             # Get tags from index instead of extracting
11             tags = tags_index.get(article['url'], ['Artikel'])
12
13             prepared_articles.append({
14                 'title': article['title'].strip(),
15                 'url': article['url'],
16                 'content': article['content'].strip(),
17                 'tags': tags,
18                 'image_url': image_url,
19                 'date': article['date']
20             })
21
22     # Sort articles by date (newest first)
23     prepared_articles.sort(key=lambda x: parse_date(x['date']), reverse=True)
24
25     # Calculate total pages
26     total_articles = len(prepared_articles)
27     total_pages = (total_articles + per_page - 1) // per_page
28
29     # Get articles for current page
30     start_idx = (page - 1) * per_page
31     end_idx = start_idx + per_page
32
33     # Add pagination range
34     if total_pages <= 10:
35         page_range = range(1, total_pages + 1)
36     else:
37         if page <= 5:
38             page_range = range(1, 11)
39         elif page > total_pages - 5:
40             page_range = range(total_pages - 9, total_pages + 1)
41         else:
42             page_range = range(page - 4, page + 6)
43
44     return {
45         'articles': prepared_articles[start_idx:end_idx],
46         'total_pages': total_pages,
47         'current_page': page,
48         'has_next': page < total_pages,
49         'total_articles': total_articles,
50         'page_range': list(page_range) # Add this
51     }

```

Pada tahap ini, **data artikel** diformat untuk memastikan informasi yang relevan disusun dengan baik dan siap untuk ditampilkan di antarmuka pengguna. **Tag** dan **URL gambar** disisipkan ke dalam setiap artikel untuk memberikan informasi tambahan yang dapat meningkatkan keterbacaan dan kegunaan artikel. Jika gambar tidak tersedia, URL gambar pengganti akan digunakan. Selanjutnya, artikel-artikel tersebut **diurutkan berdasarkan tanggal** publikasi, memastikan bahwa artikel yang

lebih baru ditampilkan lebih dahulu. Pengurutan ini dilakukan dengan menggunakan fungsi **parse_date**, yang mengonversi string tanggal menjadi objek **datetime** dan membandingkannya untuk urutan yang benar. Terakhir, artikel dibagi ke dalam **halaman-halaman** menggunakan parameter **page** dan **per_page** untuk mendukung **pagination**. Pembagian ini memastikan bahwa hanya sejumlah artikel tertentu yang ditampilkan per halaman, mengurangi beban halaman dan membuatnya lebih mudah dinavigasi oleh pengguna. Dengan cara ini, aplikasi dapat menampilkan artikel secara efisien dan terstruktur, memungkinkan pengguna untuk melihat dan menavigasi artikel sesuai kebutuhan.

iv. Similarity Calculation



```
1 def calculate_cosine_similarity(query_vector, document_vector):
2     all_terms = set(query_vector.keys()) | set(document_vector.keys())
3     query_array = np.array([query_vector.get(term, 0) for term in all_terms])
4     doc_array = np.array([document_vector.get(term, 0) for term in all_terms])
5     query_array = query_array.reshape(1, -1)
6     doc_array = doc_array.reshape(1, -1)
7     return cosine_similarity(query_array, doc_array)[0][0]
8
9 def calculate_jaccard_similarity(query_terms, document_terms):
10     query_set = set(query_terms.keys())
11     doc_set = set(document_terms.keys())
12     intersection = len(query_set.intersection(doc_set))
13     union = len(query_set.union(doc_set))
14     return intersection / union if union != 0 else 0
```

Cosine similarity mengukur kemiripan antara dua vektor berdasarkan sudut atau arah di dalam ruang vektor. Dalam konteks ini, vektor yang dimaksud adalah representasi TF-IDF dari dokumen dan query pencarian. Semakin kecil sudut antara kedua vektor, semakin besar nilai kemiripan mereka, yang dihitung menggunakan rumus cosine, menghasilkan nilai antara 0 (tidak mirip) hingga 1 (identik). Ini memungkinkan pencocokan dokumen yang relevan dengan query berdasarkan kesamaan dalam distribusi kata-kata penting.

Jaccard similarity, di sisi lain, mengukur kesamaan antara dua himpunan (set) dengan membandingkan ukuran irisan (intersection) dan gabungan (union) elemen-elemen dalam kedua himpunan tersebut. Dalam hal ini, elemen yang dibandingkan adalah kata-kata dalam query dan dokumen. Kemiripan Jaccard dihitung dengan rumus:

$$JACCARD = \frac{|A \cap B|}{|A \cup B|}$$

di mana A dan B adalah himpunan kata dari query dan dokumen. Nilai similarity ini juga berkisar antara 0 hingga 1, dengan nilai 1 menunjukkan bahwa dua himpunan tersebut sepenuhnya identik (memiliki elemen yang sama) dan nilai 0 menunjukkan tidak ada kesamaan sama sekali.

v. Search

```

1 def search(query, method='cosine'):
2     # Stem query
3     query_words = query.lower().split()
4     stemmed_query_words = [stemmer.stem(word) for word in query_words]
5
6     # Create query vector from stemmed words
7     query_vector = defaultdict(float)
8     for term in stemmed_query_words:
9         query_vector[term] += 1
10
11 # Use existing TF-IDF index
12 results = []
13 for url, doc_vector in tfidf_index.items():
14     similarity = calculate_cosine_similarity(query_vector, doc_vector) if method == 'cosine' else calculate_jaccard_similarity(query_vector, doc_vector)
15     doc_data = next((doc for doc in crawled_data if doc['url'] == url), None)
16     if doc_data and similarity > 0:
17         results.append({
18             'url': url,
19             'title': doc_data.get('title', ''),
20             'content': doc_data.get('content', '')[:200] + '...',
21             'score': similarity,
22             'date': doc_data.get('date', 'N/A'),
23             'image_url': doc_data.get('image_url', ''),
24             'tags': tags_index.get(url, ['Artikel'])
25         })
26
27 results.sort(key=lambda x: x['score'], reverse=True)
28 return results[:10]

```

Pencarian artikel berdasarkan query dimulai dengan memproses query yang dimasukkan oleh pengguna. Query tersebut terlebih dahulu di-stem untuk mengubah kata-kata menjadi bentuk dasarnya, kemudian vektor frekuensi kata dari query dibuat, yang mencatat jumlah kemunculan kata-kata relevan dalam query. Selanjutnya, skor kemiripan dihitung antara vektor query dan vektor setiap dokumen menggunakan metode yang dipilih, yaitu **Cosine Similarity** atau **Jaccard Similarity**. Dalam **Cosine Similarity**, skor dihitung berdasarkan kedekatan antara kedua vektor dalam ruang vektor, sementara **Jaccard Similarity** mengukur kesamaan berdasarkan ukuran irisan dan gabungan kata antara query dan dokumen. Setelah menghitung skor kemiripan untuk semua dokumen, hasil pencarian diurutkan berdasarkan skor tersebut, dengan dokumen yang memiliki skor tertinggi ditempatkan di urutan teratas. Hanya sejumlah dokumen teratas yang akan dikembalikan sebagai hasil pencarian, memberikan artikel-artikel yang paling relevan dengan query pengguna.

f. Routes Flask

i. Homepage


```

1 @app.route('/')
2 def home():
3     page = request.args.get('page', 1, type=int)
4     articles_data = prepare_article_data(crawled_data, page=page)
5     return render_template('index.html', **articles_data)

```

Menampilkan halaman utama dimulai dengan mengambil data artikel yang telah diformat dan disiapkan sebelumnya. Artikel-artikel ini kemudian diurutkan berdasarkan tanggal publikasi dan dibagi dalam beberapa halaman menggunakan pagination. Setiap artikel yang ditampilkan akan mencakup informasi seperti judul, cuplikan konten, tanggal publikasi, tag, dan gambar. Jika gambar tidak tersedia, gambar pengganti akan ditampilkan. Artikel-artikel ini disajikan dalam format grid atau daftar yang mudah dibaca, memberikan pengguna antarmuka yang rapi dan terstruktur. Halaman utama ini juga menyertakan navigasi untuk berpindah antar halaman, memastikan bahwa artikel-artikel yang lebih banyak dapat dijelajahi tanpa membebani tampilan satu halaman dengan terlalu banyak data. Dengan cara ini, halaman utama memastikan bahwa pengguna dapat dengan mudah menemukan dan mengakses artikel-artikel yang relevan sesuai kebutuhan mereka.

ii. Search Page

```

1 @app.route('/search')
2 def search_results():
3     query = request.args.get('q', '')
4     method = request.args.get('method', 'cosine')
5     if not query:
6         return render_template('index.html')
7
8     results = search(query, method)
9
10    # Format scores as percentages and ensure all required fields exist
11    for result in results:
12        result['score'] = result['score']
13        # Ensure image_url exists
14        if not result.get('image_url') or result['image_url'] == "URL gambar tidak tersedia":
15            result['image_url'] = "https://via.placeholder.com/400x200"
16        # Ensure tags exist
17        if not result.get('tags'):
18            result['tags'] = ['Artikel']
19
20    return render_template('result.html', query=query, results=results, method=method)

```

Menampilkan hasil pencarian dimulai dengan memproses query yang dimasukkan oleh pengguna dan menghitung skor kemiripan antara query dan dokumen yang ada. Setelah itu, artikel-artikel yang relevan berdasarkan skor kemiripan tertinggi

akan ditampilkan pada halaman hasil pencarian. Setiap artikel yang ditampilkan mencakup informasi seperti judul, cuplikan konten, tanggal publikasi, tag, dan gambar artikel (dengan gambar pengganti jika gambar asli tidak tersedia). Hasil pencarian akan diurutkan berdasarkan skor kemiripan, dengan artikel yang paling relevan berada di urutan teratas. Selain itu, halaman ini juga menyediakan informasi terkait tentang metode pencarian yang digunakan (Cosine atau Jaccard Similarity), memungkinkan pengguna untuk memahami cara pencarian dilakukan. Navigasi untuk berpindah antar halaman hasil pencarian juga disediakan untuk memudahkan pengguna menelusuri lebih banyak artikel sesuai kebutuhan.

g. Jalankan Server



Menjalankan aplikasi Flask dalam mode debug.

h. Pengembangan Halaman Web

a. index.html

i. Header (Tag `<head>` dan pengaturannya)



Pada bagian awal halaman web, encoding halaman diatur ke **UTF-8** untuk memastikan bahwa teks dalam halaman dapat mendukung berbagai karakter internasional, termasuk karakter non-ASCII, yang penting untuk menjaga konsistensi tampilan teks di berbagai perangkat

dan bahasa. Selanjutnya, tag **viewport** ditambahkan untuk memastikan halaman dapat merespons dengan baik pada perangkat mobile, sehingga tampilan halaman akan menyesuaikan ukuran layar perangkat yang digunakan. **Judul halaman** diatur sebagai "MindSeek", yang akan tampil di tab browser atau judul halaman saat pengguna mengaksesnya. Terakhir, file **CSS eksternal** (home.css) dihubungkan melalui **url_for**, yang memungkinkan Flask untuk mengelola file statis dan memastikan gaya yang diterapkan konsisten di seluruh aplikasi web. Pengaturan ini memungkinkan halaman untuk tampil dengan baik dan responsif di berbagai platform, serta memastikan pengelolaan file statis yang efisien.

i. Body Awal dan Particles.js Background



Elemen `<div>` dengan ID **particles-js** digunakan untuk menampilkan efek partikel interaktif di latar belakang halaman. Efek ini diatur dan dikendalikan menggunakan JavaScript, dengan konfigurasi yang disesuaikan melalui file eksternal untuk menciptakan tampilan dinamis dan menarik yang dapat meningkatkan pengalaman pengguna di halaman tersebut. Sementara itu, elemen `<div>` dengan kelas **container** berfungsi sebagai pembungkus utama untuk seluruh isi halaman, mengatur tata letak dan memastikan elemen-elemen di dalamnya, seperti logo, form pencarian, dan artikel, tersusun dengan rapi dan terstruktur. Kelas **container** ini juga membantu menjaga konsistensi desain dan memastikan tampilan halaman tetap responsif pada berbagai ukuran layar.

ii. Logo dan Tagline

```
1 <div class="logo-container">
2     <div class="logo">MindSeek</div>
3     <div class="tagline">Solusi Kesehatanmu</div>
4 </div>
```

Bagian logo (MindSeek) dan tagline (Solusi Kesehatanmu) untuk memperkuat identitas aplikasi.

iii. Formulir Pencarian

```
2 <form action="/search" method="GET" class="search-container" onsubmit="return validateSearch();"
3 <div class="search-box-wrapper">
4 <div class="search-box">
5 <input type="text" class="search-input" value="" placeholder="Masukkan kata kunci..." autofocus />
6 </div>
7 <div class="search-button">
8 <button type="button" value="Cari" class="search-button">Cari</button>
9 </div>
10 </div>
11 <div class="search-results">
12 <div class="search-results">
13 <div class="search-results">
14 <div class="search-results">
15 <div class="search-results">
16 <div class="search-results">
17 <div class="search-results">
18 <div class="search-results">
19 <div class="search-results">
20 <div class="search-results">
21 <div class="search-results">
22 </div>
23 </form>
```

Formulir pencarian ini memungkinkan pengguna untuk memasukkan query pencarian (**q**) dan memilih metode pencarian yang diinginkan, yaitu **Cosine Similarity** atau **Jaccard Similarity**, yang dikirimkan melalui metode **GET**. Ini berarti data pencarian akan dikirimkan sebagai parameter URL saat pengguna mengirimkan formulir. Sebelum pengiriman formulir, **onsubmit="return validateSearch()"** memastikan bahwa input dari pengguna divalidasi terlebih dahulu, seperti memastikan bahwa query tidak kosong atau memenuhi kriteria tertentu, untuk mencegah pengiriman data yang tidak valid. Di dalam formulir, terdapat dua tombol yang memungkinkan pengguna untuk memilih metode pencarian yang diinginkan: satu untuk **Cosine Similarity** dan satu lagi untuk **Jaccard Similarity**. Tombol-tombol ini memudahkan pengguna dalam memilih metode pencarian yang sesuai dengan preferensi mereka, memberikan kontrol penuh atas cara pencarian dilakukan.

iv. Daftar Artikel

```

1 <div class="articles-container">
2   <h2>Artikel Terbaru</h2>
3   <div class="articles-grid">
4     {% for article in articles %}
5       <div class="article-card" onclick="window.location.href='{{ article.url }}'">
6         <div class="article-image" style="background-image: url('{{ article.image_url }}')"></div>
7         <div class="article-content">
8           <div class="article-title">{{ article.title }}</div>
9           <div class="article-tags">
10            {% for tag in article.tags %}
11              <span class="tag">{{ tag }}</span>
12            {% endfor %}
13          </div>
14          <div class="article-description">
15            {{ article.content[:200] + '...' if article.content else '' }}
16          </div>
17        </div>
18      </div>
19    {% endfor %}
20  </div>

```

Daftar artikel ditampilkan dalam format **grid**, di mana setiap artikel disajikan dalam bentuk kartu yang terdiri dari beberapa elemen penting. Setiap kartu artikel menampilkan **gambar** yang diambil dari properti `article.image_url`, **judul** dari artikel yang diambil dari `article.title`, **tag** yang terkait dengan artikel, yang diambil dari `article.tags`, serta **cuplikan konten** yang merupakan potongan awal dari isi artikel, diambil dari `article.content`. Cuplikan ini memberikan gambaran singkat mengenai artikel tersebut untuk menarik perhatian pembaca. Setiap kartu artikel dilengkapi dengan fungsionalitas interaktif, di mana **klik pada kartu artikel** akan membuka URL yang relevan, yang diambil dari `article.url`, untuk mengarahkan pengguna ke halaman artikel lengkap. Dengan cara ini, pengguna dapat dengan mudah menelusuri artikel-artikel yang tersedia dan mengakses konten yang lebih mendalam.

v. Pagination

```

1 {% if has_next %}
2   <div class="pagination-container">
3     {% if current_page > 1 %}
4       <a href="/?page={{ current_page - 1 }}" class="next-button">Sebelumnya</a>
5     {% endif %}
6
7     {% for p in page_range %}
8       <a href="/?page={{ p }}" id="page-{{ p }}"
9         class="pagination-button {% if p == current_page %}active{% endif %}">
10        {{ p }}
11      </a>
12    {% endfor %}
13
14    {% if current_page < total_pages %} <a href="/?page={{ current_page + 1 }}" class="next-button">
15      Berikutnya</a>
16    {% endif %}
17  </div>
18 {% endif %}

```

Sistem **pagination** digunakan untuk membagi daftar artikel menjadi beberapa halaman, memudahkan pengguna untuk menavigasi artikel-artikel yang lebih banyak tanpa membebani satu halaman

dengan terlalu banyak konten. Tombol **"Sebelumnya"** dan **"Berikutnya"** ditampilkan berdasarkan kondisi saat ini, yaitu jika pengguna berada di halaman pertama, tombol **"Sebelumnya"** tidak akan tampil, dan jika pengguna berada di halaman terakhir, tombol **"Berikutnya"** akan disembunyikan. Halaman saat ini diberi gaya **active**, yang menandakan halaman mana yang sedang dibuka oleh pengguna, dengan memberikan tampilan visual yang berbeda (misalnya, warna yang lebih terang atau background yang berbeda). Ini memungkinkan pengguna untuk dengan mudah melacak posisi mereka di antara halaman-halaman yang ada dan berpindah antar halaman artikel secara intuitif.

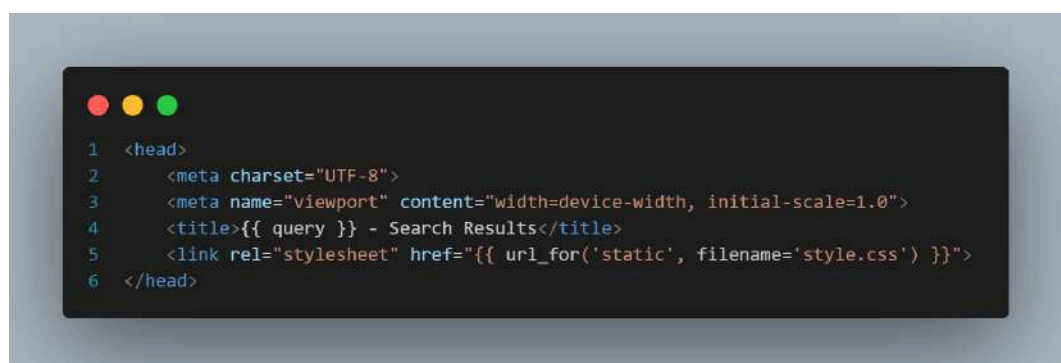
vi. Skrip latar belakang web



Skrip untuk mengaktifkan efek partikel latar belakang menggunakan particles.js. particle-config.js dan home.js adalah file JavaScript untuk konfigurasi partikel dan interaktivitas halaman.

b. result.html

i. Header (Tag **<head>** dan pengaturannya)



Encoding halaman diset ke **UTF-8** untuk memastikan bahwa halaman dapat mendukung berbagai karakter internasional, termasuk karakter non-ASCII, yang penting untuk menjaga tampilan teks yang konsisten di berbagai bahasa dan perangkat. Tag **viewport** ditambahkan untuk memastikan halaman web responsif, artinya tampilan halaman akan menyesuaikan ukuran layar perangkat yang digunakan, seperti smartphone atau tablet. **Judul halaman** diatur untuk menampilkan

query pencarian yang dimasukkan pengguna, memberikan konteks yang jelas mengenai apa yang sedang dicari. Terakhir, file **CSS eksternal** (style.css) dihubungkan menggunakan **url_for**, yang memungkinkan Flask untuk mengelola file statis dan memastikan gaya halaman dapat diterapkan secara konsisten, serta mempermudah pemeliharaan dan pengelolaan file dalam aplikasi web.

ii. Header dan Formulir Pencarian



```
1 <header>
2   <div class="logo"><a href="/">MindSeek</a></div>
3   <div class="search">
4     <form action="{{ url_for('search_results') }}" method="GET" class="search-container">
5       <input type="text" name="q" value="{{ query }}" required placeholder="Search..." class="search-box">
6       <div class="buttons-container">
7         <button type="submit" name="method" value="cosine" class="search-button {% if method == 'cosine' %}active{% endif %}>
8           Cosine Similarity
9         </button>
10        <button type="submit" name="method" value="jaccard" class="search-button {% if method == 'jaccard' %}active{% endif %}>
11          Jaccard Similarity
12        </button>
13      </div>
14    </form>
15  </div>
16 </header>
```

Logo **MindSeek** ditampilkan di halaman dengan sebuah tautan yang mengarah ke halaman utama, memberikan akses cepat bagi pengguna untuk kembali ke halaman utama situs. Formulir pencarian menampilkan **query** yang sedang dicari dengan menyertakan nilai `value="{{ query }}"`, sehingga pengguna dapat melihat dan mengedit query pencarian yang mereka masukkan sebelumnya. Formulir ini juga memiliki dua tombol untuk memilih metode pencarian: **Cosine Similarity** dan **Jaccard Similarity**. Tombol yang aktif, sesuai dengan pilihan metode pencarian yang dipilih pengguna, diberi gaya **active**, sehingga pengguna dapat dengan mudah mengetahui metode pencarian yang sedang diterapkan. Pengaturan ini meningkatkan kenyamanan pengguna dalam menavigasi dan menyesuaikan pencarian mereka di aplikasi.

iii. Bagian Hasil Pencarian (Jika Ada)


```

1 <main>
2   {% if results %}
3   <!-- Slider for search results -->
4   <div class="slider">
5     <div class="list">
6       {% for result in results %}
7       <div class="item {% if loop.first %}active{% endif %}">
8         <!-- Display image -->
9         
10        <div class="content">
11          <!-- Display category or tags as a label -->
12          <p id="tag">{{ result.tags[0] if result.tags else 'Artikel' }}</p>
13          <!-- Display title -->
14          <h2 class="animate">{{ result.title }}</h2>
15          <!-- Display content preview -->
16          <p class="animate">{{ result.content }}</p>
17          <!-- Display article date -->
18          <small class="animate">{{ result.date }}</small>
19          <!-- Button for navigating to the URL -->
20          <a href="{{ result.url }}" class="btn animate" target="_blank">Go to Page</a>
21        </div>
22      </div>
23    {% endfor %}
24  </div>
25  <div class="arrows">
26    <button id="prev">&#10094;</button>
27    <button id="next">&#10095;</button>
28  </div>
29 </div>

```

Bagian **thumbnail** digunakan untuk menampilkan daftar kecil gambar artikel yang berfungsi sebagai navigasi slider. Setiap thumbnail menampilkan **gambar pratinjau** dari artikel yang relevan, memudahkan pengguna untuk melihat representasi visual dari hasil pencarian mereka. Selain itu, pada setiap thumbnail, **skor kemiripan** antara query pencarian dan artikel ditampilkan, dengan nilai skor yang dihitung menggunakan **Cosine** atau **Jaccard Similarity**. Skor kemiripan ini dipertunjukkan dengan **pembulatan hingga tiga desimal**, memberikan gambaran yang jelas tentang seberapa relevan artikel tersebut dengan query yang dimasukkan. Dengan menampilkan thumbnail dan skor kemiripan, pengguna dapat dengan mudah menavigasi hasil pencarian dan memilih artikel yang paling relevan berdasarkan tampilan visual dan skor yang diberikan.

iv. Thumbnail Navigasi


```

1 <div class="thumbnail">
2     {% for result in results %}
3         <div class="item {% if loop.index == 1 %}active{% endif %}">
4             
5             <div class="content">
6                 <!-- Display cosine or jaccard score -->
7                 <p>Score: {{ result.score | round(3) }}</p>
8             </div>
9         </div>
10     {% endfor %}
11 </div>

```

Bagian **thumbnail** digunakan untuk menampilkan daftar kecil gambar artikel sebagai navigasi slider, memudahkan pengguna untuk melihat pratinjau artikel yang relevan tanpa perlu menggulir seluruh halaman. Setiap thumbnail menampilkan gambar artikel yang sesuai, yang berfungsi sebagai representasi visual untuk menarik perhatian pengguna. Selain gambar, **skor kemiripan** artikel terhadap query pencarian juga ditampilkan pada setiap thumbnail, dengan nilai skor yang dihitung menggunakan metode **Cosine Similarity** atau **Jaccard Similarity**. Skor kemiripan ini dipertunjukkan dengan **pembulatan hingga tiga desimal**, memberikan pengguna informasi yang lebih presisi tentang sejauh mana artikel tersebut relevan dengan pencarian mereka. Dengan cara ini, pengguna dapat dengan mudah menavigasi hasil pencarian dan memilih artikel yang paling relevan berdasarkan tampilan visual dan skor kemiripan yang jelas.

v. Pesan Jika Tidak Ada Hasil

```

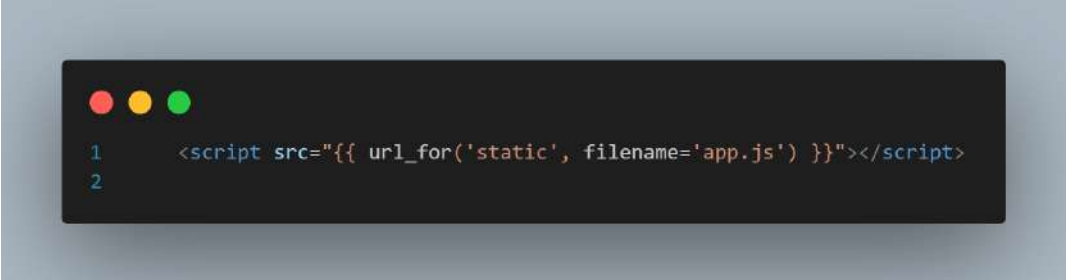
1 {% else %}
2     <!-- No Results Found -->
3     <div class="no-results">
4         <h2>No results found for "{{ query }}"</h2>
5         <p>Please try using different keywords or check your spelling.</p>
6     </div>
7     {% endif %}

```

Jika tidak ada hasil pencarian yang relevan dengan query yang dimasukkan, sebuah pesan akan ditampilkan kepada pengguna yang memberi tahu bahwa tidak ada artikel yang ditemukan. Pesan ini juga akan menyarankan pengguna untuk mencoba **menggunakan kata kunci yang berbeda** atau **memeriksa ejaan** query mereka. Ini bertujuan untuk memberikan petunjuk yang berguna agar pengguna dapat memperbaiki pencarian mereka dan mencoba lagi, serta untuk menghindari kebingungan atau frustrasi akibat hasil pencarian yang

kosong. Dengan cara ini, pengguna tetap mendapatkan umpan balik yang jelas dan dapat melanjutkan pencarian dengan lebih efektif.

vi. Skrip

A screenshot of a code editor window with a dark background and three colored window control buttons (red, yellow, green) in the top left corner. The editor contains two lines of code: line 1 is `<script src="{ url_for('static', filename='app.js') }"></script>` and line 2 is empty. The code is color-coded: `<` and `>` are blue, `script` is green, `src=` is red, and the rest is black.

```
1 <script src="{ url_for('static', filename='app.js') }"></script>
2
```

Skrip `app.js` digunakan untuk memberikan interaktivitas pada halaman, seperti navigasi slider.

2.Repository GitHub

Keseluruhan kode sudah kami upload kedalam repositori GitHub.

Link GitHub: [https: Search Engine](https://search-engine)



MINDSEEK

SISTEM PENELUSURAN INFORMASI
KESEHATAN

Kelompok 1

TEAM



Ganang Setyo Hadi
2208107010052



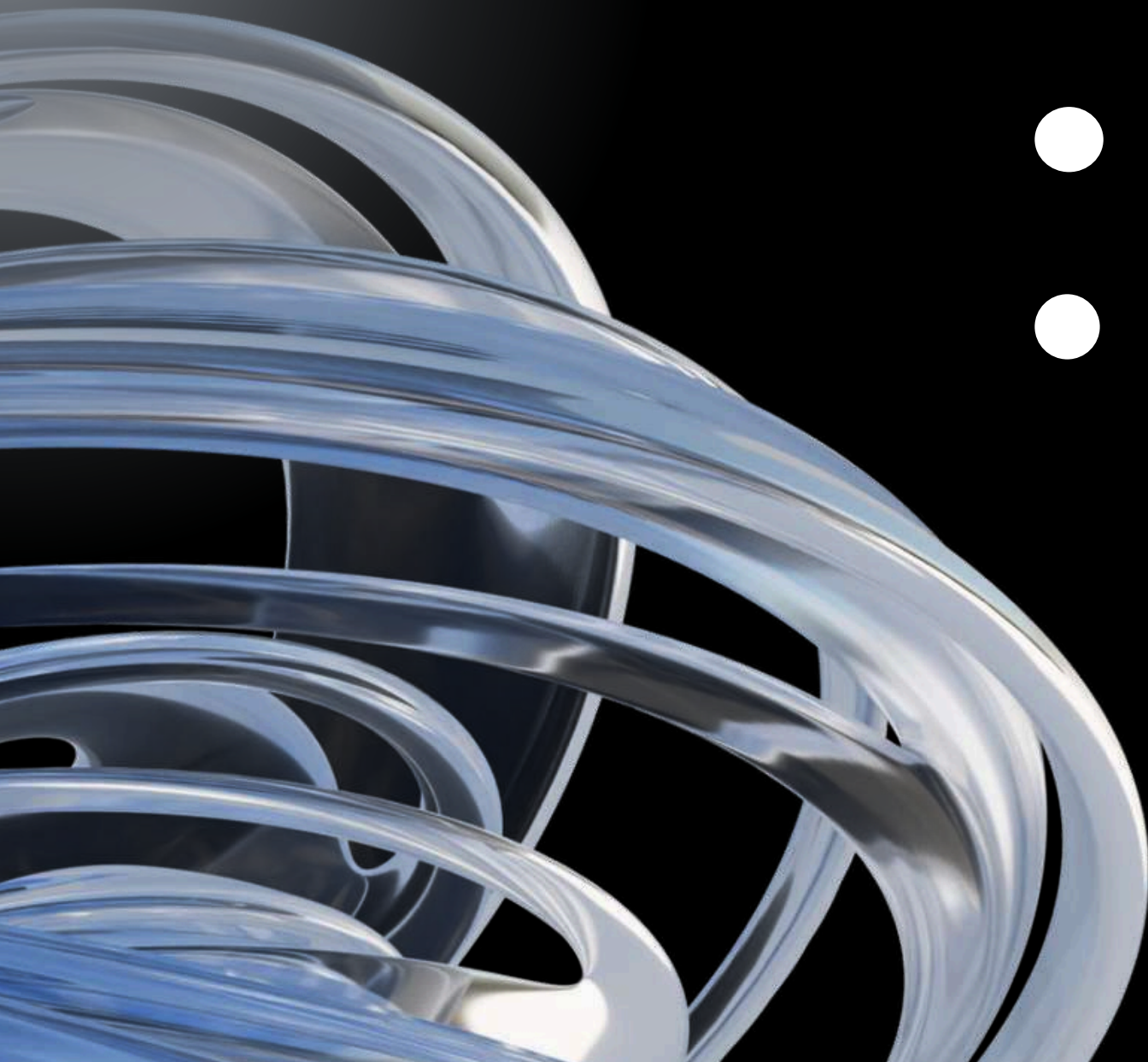
Ahmad Syah Ramadhan
2208107010033



Naufal Aqil
2208107010043

LATAR BELAKANG

- Data online terus bertambah pesat, termasuk informasi kesehatan
- Dibutuhkan sistem pencarian yang dapat memfilter dan mengurutkan informasi berdasarkan relevansi
- Pengguna kesulitan menemukan informasi yang relevan secara efisien



IDENTIFIKASI MASALAH

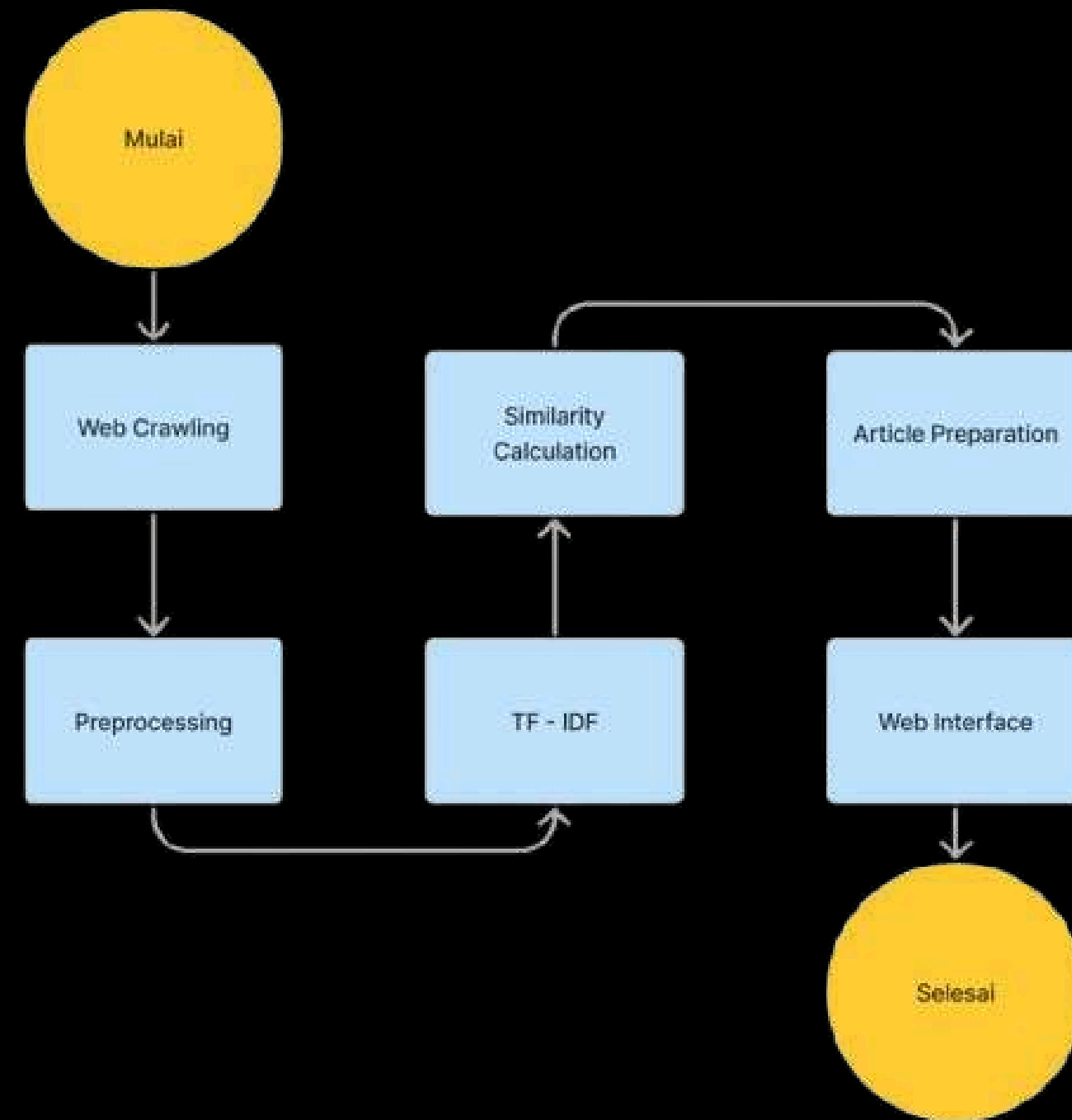
- Pencarian yang tidak efisien seringkali menghasilkan banyak hasil yang tidak relevan
- Sistem pencarian yang ada belum mengoptimalkan akurasi hasil pencarian menggunakan metode pembobotan yang terbukti
- Pengguna kesulitan menemukan artikel kesehatan yang tepat sesuai dengan kebutuhan mereka



VISION

Kami menghadirkan inovasi pencarian cerdas yang memadukan akurasi tinggi dan kemudahan penggunaan, untuk memberikan solusi penelusuran informasi kesehatan yang tepat sesuai kebutuhan Anda.

ARSITEKTUR SISTEM OVERVIEW



KOMPONEN INTI

Web Crawler

Framework: Scrappy

Target: Artikel Halodoc

Data: Judul, konten, gambar, tanggal, URL

TF-IDF Processor

Term frequency & pembobotan

Pembuatan inverted index

Web Interface

Framework Flask

Responsive, filter & pagination

Preprocessing Process

Tokenisasi, stopwords, stemming

Pembersihan teks

Similarity Calculator

Cosine & Jaccard Similarity

Perankingan relevansi

TECH STACK

BACKEND

- Python 3.11.1
- Flask Framework
- Scrapy
- NLTK
- Scikit-learn

FRONTEND

- HTML5/CSS3
- JavaScript
- Responsive Design
- Interactive UI Elements

DEV TOOLS

- Visual Studio Code
- Git version control
- Github
- VirtualEnv

FITUR UTAMA

Web crawling dari sumber terpercaya

Preprocessing
text dan
perhitungan TF-
IDF

Dual similarity
methods (Cosine
& Jaccard)

Breathtaking web interface

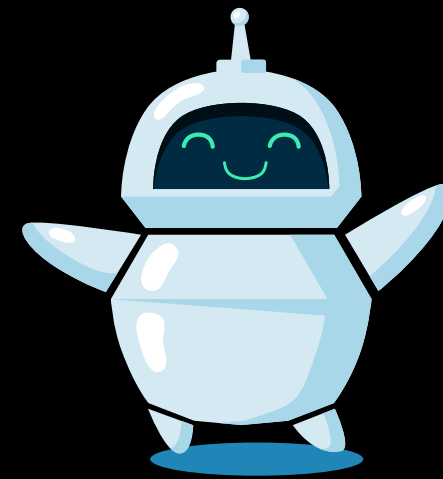
Kami menghadirkan inovasi
pencarian cerdas yang
memadukan akurasi tinggi dan
kemudahan penggunaan

CRAWLING & INDEXING



CRAWLER

- Crawler menggunakan Scrapy framework
- Target: Artikel kesehatan dari halodoc.co



Batasan crawling

- Maks 2000 halaman
- Jeda 0.2 detik antar request
- Patuh pada robots.txt



Ekstraksi data

- Judul artikel
- URL
- Konten
- Tanggal publikasi
- Gambar terkait

PERHITUNGAN TF - IDF

Perhitungan TF-IDF dalam dua pass:

- Pass 1: Hitung Document Frequency
- Pass 2: Hitung TF-IDF untuk setiap kata

Preprocessing

- Normalisasi teks (lowercase)
- Tokenisasi
- Hapus stopwords
- Stemming (Sastrawi)

Normalisasi Vektor

Norma Euclidean

SIMILIARITY



Cosine Similarity

- Mengukur sudut antara vektor dokumen
- Formula: $\cos(\theta) = (A \cdot B) / (||A|| * ||B||)$
- Nilai: 0-1, semakin dekat 1 semakin mirip



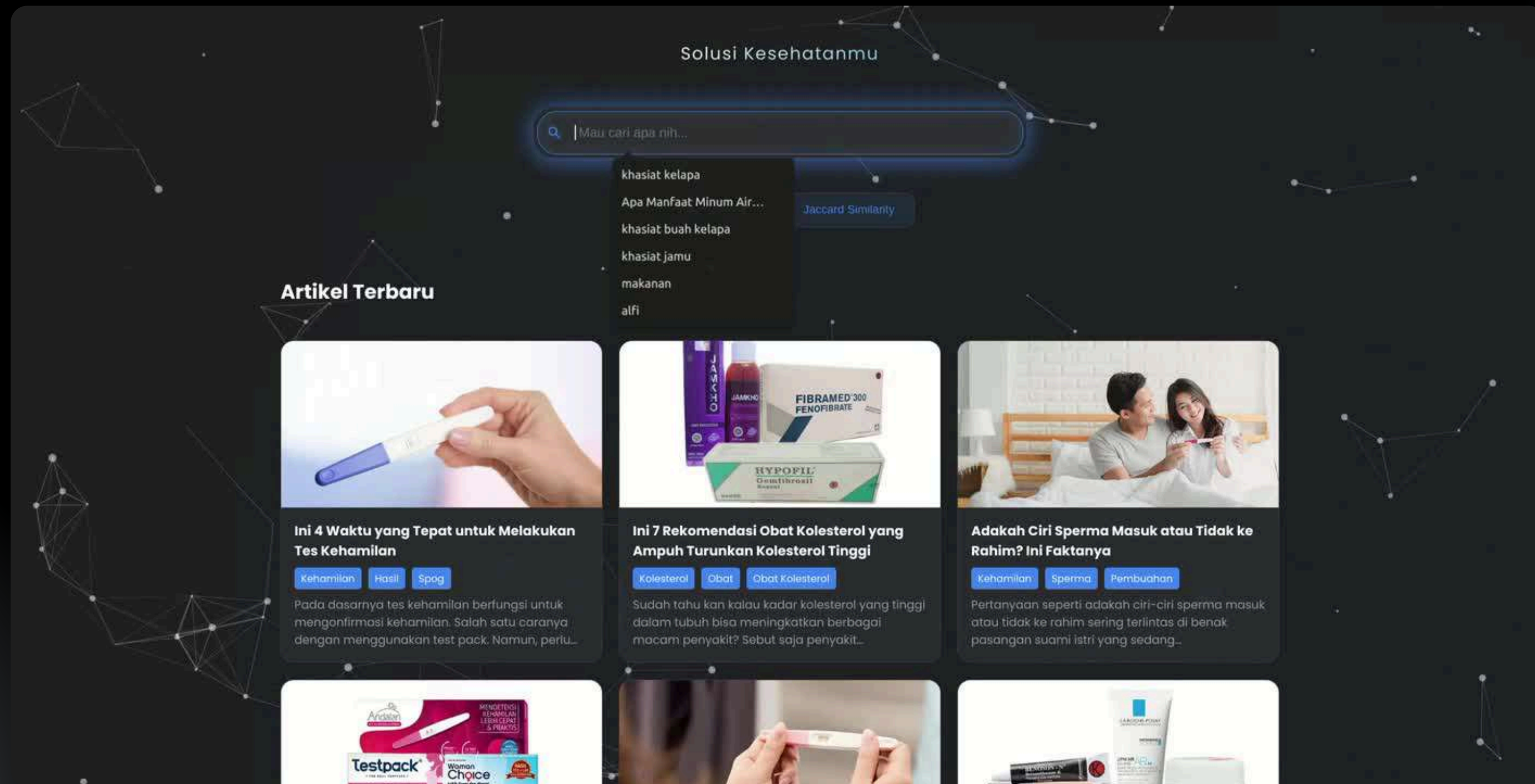
Jaccard Similarity

- Mengukur kesamaan antar himpunan kata
- Formula: $J(A,B) = |A \cap B| / |A \cup B|$
- Nilai: 0-1, 1 berarti identik

KEUNGGULAN

Penerapan dual similarity methods meningkatkan akurasi pencarian informasi kesehatan. Antarmuka pengguna yang intuitif dan responsif memastikan pengalaman pencarian yang lancar, memungkinkan pengguna untuk dengan mudah menemukan konten kesehatan yang terverifikasi dari sumber terpercaya. Sistem dirancang khusus untuk pemrosesan bahasa Indonesia, menjadikannya solusi yang efektif dan relevan bagi pengguna yang mencari informasi kesehatan di era digital yang terus berkembang.

DEMO



KESIMPULAN

"MindSeek" menegaskan pentingnya inovasi dalam penelusuran informasi kesehatan di era digital. Dengan menerapkan metode Cosine dan Jaccard Similarity, platform ini menawarkan akurasi tinggi dan antarmuka yang ramah pengguna, memenuhi kebutuhan akan informasi kesehatan yang relevan dan terpercaya. "MindSeek" tidak hanya memberikan solusi efektif saat ini, tetapi juga membuka jalan untuk pengembangan sistem serupa di masa depan, siap menghadapi tantangan pencarian informasi yang semakin kompleks.



THANK YOU

KELOMPOK 1