e-ISSN 2540-9840 p-ISSN 1412-3770 Jurnal Ilmiah Sains, April 2024, 24(1): 57-69
Accredited by Ministry of Education, Culture, DOI: https://doi.org/10.35799/jis.v24i1.52967
Research and Technology No: 105/E/KPT/2022 Available online at https://ejournal.unsrat.ac.id/v3/index.php/jis

Implementasi Algoritma *Vector Space Model* untuk Pencarian Dokumen pada Aplikasi *Monitoring* Uji Material (Studi Kasus: Lab. BPJN Jayapura)

Serwin Gonzaga Rumagit[1)], Debby Paseru[2*)], Thomas Ch. Suwanto[3]

[1,2,3)]**Program Studi Teknik Informatika, Universitas Katolik De La Salle Manado, Indonesia**
*Corresponding author: dpaseru@unikadelasalle.ac.id

ABSTRAK

Aplikasi *monitoring* uji material yang dibangun pada tahun 2021 bertujuan untuk memudahkan pekerjaan BPJN Jayapura dalam memasukkan berkas data administrasi hingga memudahkan proses *monitoring* uji material yang dilakukan dari tahap proses sampai tahap hasil pengujian material. Namun, dalam penggunaannya selama 2 tahun ini, aplikasi tersebut memiliki masalah. Aplikasi tersebut tidak mampu melakukan pencarian dokumen yang tersimpan dalam basis data sehingga pencarian dilakukan satu per satu. Hal ini menyulitkan pengguna dikarenakan dokumen yang ada lebih dari 30 buah. Olehnya penelitian ini akan berfokuskan pada pembuatan mesin pencari dokumen (*search engine*) dengan menggunakan metode *Vector Space Model*. Algoritma ini akan mengukur tingkat kesamaan dokumen dengan kata kunci atau *query* yang dimasukkan oleh pengguna. Penelitian ini mengimplementasi metode *Vector Space Model* untuk memudahkan pencarian dokumen dalam aplikasi. Dalam melaksanakan penelitian ini, akan digunakan metodologi *Software Development Life Cycle* sebagai metode dalam pembuatan *search engine* tersebut dengan kakas pemodelan *Unified Modelling Language* versi 2.5.1 dan bahasa pemrograman PHP. Berdasarkan pengujian yang telah dilakukan, metode *Vector Space Model* berhasil diimplementasikan dan mampu berfungsi sebagai *search engine* dengan tingkat presisi sebesar 0.83 dan *recall* sebesar 1 untuk pencarian 40 dokumen sebagai sampel.

Kata Kunci: Pencarian dokumen; Jayapura; *Vector Space Model*

Implementation of the Vector Space Model Algorithm for Document Searching in Material Test Monitoring Applications (Case Study: BPJN Jayapura Lab)

ABSTRACT

The material test monitoring application built in 2021 aims to facilitate BPJN Jayapura's work in entering administrative data files to facilitate the material test monitoring process carried out from the process stage to the material test results stage. However, in using it for 2 years, the application has had problems. The application is unable to search for documents stored in the database so the search is carried out one by one. This makes it difficult for users because there are more than 30 documents. Therefore, this research will focus on creating a document search engine (search engine) using the Vector Space Model method. This algorithm will measure the level of similarity of documents to the keywords or queries entered by the user. This research implements the Vector Space Model method to make it

easier to search for documents in the application. In carrying out this research, the Software Development Life Cycle methodology will be used as a method for creating the search engine using the Unified Modeling Language version 2.5.1 modeling tool and the PHP programming language. Based on the tests that have been carried out, the Vector Space Model method has been successfully implemented and is able to function as a search engine with a precision level of 0.83 and a recall of 1 for searching 40 documents as samples.

Keywords: Document search; Jayapura; vector space model

(Article History: Received 05-12-2023; Accepted 09-04-2024; Published 19-04-2024)

PENDAHULUAN

Aplikasi *monitoring* uji material merupakan aplikasi yang dibangun untuk membantu Balai Pelaksana Jalan Nasional (BPJN) Jayapura dalam melakukan proses pengujian material. Aplikasi ini telah digunakan oleh BPJN Jayapura sejak tahun 2021 sampai saat ini. Aplikasi ini memudahkan BPJN Jayapura dalam memasukkan data administrasi, memudahkan pengecekan berkas, memudahkan *monitoring* secara langsung tahap pengujian material dari proses hingga hasil pengujian dan melakukan *control* terhadap proses dari pengujian. Perusahaan yang melakukan pengujian material pada lab BPJN Jayapura tentunya bukan hanya sekali, tetapi setiap perusahaan dapat melakukan pengujian berkali-kali setiap harinya dengan material yang sama atau berbeda-beda sehingga terdapat banyak sekali laporan pengujian material.

Berdasarkan hasil wawancara yang penulis lakukan pada tanggal 8 Februari 2023, diketahui bahwa terdapat kekurangan dalam aplikasi *monitoring* tersebut karena para pengguna seringkali merasa kesulitan dalam mencari suatu dokumen. Pencarian dokumen yang ada dalam aplikasi *monitoring* BPJN Jayapura hanya mencari dokumen berdasarkan dari judul dokumen atau nama *file* dan seringkali mereka hanya ingin mencari sebuah data dalam dokumen, namun mereka harus membuka dokumen yang mempunyai nama *file* atau judul dokumen yang sama satu per satu sehingga menurut mereka proses tersebut harus dapat diefisiensikan kembali agar mampu meningkatkan kinerja dan kecepatan serta ketepatan kerja dari pengguna. Karena dengan perkembangan infrastruktur yang sangat masif dibangun oleh Pemerintah di wilayah Jayapura, nantinya aplikasi *monitoring* tersebut akan menyimpan sangat banyak sekali data pengujian material. Kekurangan tersebut diakibatkan karena tidak adanya suatu algoritma dalam fitur *search engine* yang dapat melakukan pencarian kalimat atau kata secara mendalam pada aplikasi tersebut.

Optimasi pada suatu *search engine* dilakukan dengan cara memberikan algoritma tertentu dalam suatu *search engine* yang dapat memproses kata atau kalimat yang dimasukkan oleh pengguna dan memberikan hasil yang cocok dan relevan dengan yang dimasukkan oleh pengguna. Salah satunya adalah dengan menggunakan algoritma *Vector Space Model* (Wicaksono *et al*., 2015). Algoritma *Vector Space Model* adalah suatu algoritma dengan model matematika yang bertujuan untuk menghitung dan menentukan korelasi kesamaan yang ada dalam dokumen dengan informasi yang diberikan oleh pengguna (Sulianta, 2010). Model ini digunakan untuk mengukur informasi yang ada pada suatu sistem dengan *query* yang dimasukkan oleh pengguna dengan konsep *document-matching* (Muktyas & Abdillah, 2013). Algoritma ini nantinya dapat menghitung dan

memberikan hasil berupa dokumen yang berisi kata-kata yang mempunyai tingkat kesamaan dengan *query* yang dimasukkan oleh pengguna (Ningtyas *et al*., 2018). Algoritma *Vector Space Model* digunakan dalam penelitian ini karena dapat melakukan pengolahan cepat sekaligus efisien terhadap data teks dengan jumlah yang sedikit dan jumlah yang sangat besar. Hal ini didukung oleh beberapa penelitian sebelumnya, yaitu Perbandingan Metode *Vector Space Model* dan *Weighted Tree Similarity* dengan *Cosine Similarity* pada Kasus Pencarian Informasi Pedoman Pengobatan Dasar di Puskesmas (Wicaksono *et al*., 2015), Aplikasi Pencarian Film Rekognisi dengan Metode *Vector Space Model* berbasis *Android* (Budisetyo *et al*., 2018), Aplikasi Pencarian Bahan Pustaka di Perpustakan Menggunakan Metode *Vector Space Model* (Bahri, 2020), Implementasi Metode Vector Space Model pada Search Engine Perpustakaan (Salmon et al., 2020) dan Penerapan Vector Space Model (VSM) Pada Sistem Pencarian Artikel Arkeologi (Makmun *et al*., 2022). Penelitian ini bertujuan untuk mengimplementasikan metode *Vector Space Model* dalam aplikasi pencarian dokumen hasil uji material. Penelitian ini juga mendukung penambahan dokumen yang akan dicari sehingga jumlah dokumen perbandingan akan terus bertambah.

METODE PENELITIAN

Metode penelitian yang digunakan dalam penelitian ini terdiri dari beberapa tahapan (Dahlan, 2017), yaitu:

1. Identifikasi Kebutuhan Aplikasi
   Tahapan ini dilakukan bertujuan untuk melakukan perencanaan terhadap kebutuhan yang ada di dalam aplikasi untuk menyelesaikan masalah yang dibahas dalam penelitian. Tahapan ini mengikuti fase pertama dan fase kedua dari metodologi SDL, yaitu fase perencanaan dan analisis.
2. Perancangan
   Pada tahap ini akan dilakukan pembuatan rancangan-rancangan yang dibutuhkan dalam membangun aplikasi pada penelitian ini. Perancangan ini akan mengikuti fase dalam metodologi pengembangan aplikasi, yaitu SDLC, khususnya fase perancangan.
3. Implementasi
   Pada tahap ini akan dilakukan pembangunan aplikasi dengan berdasarkan pada analisis dan perancangan yang telah dilakukan pada tahap-tahap sebelumnya. Tahapan ini akan mengikuti fase dalam metodologi pengembangan aplikasi, yaitu SDLC, khususnya fase implementasi.
4. Pengujian
   Pada tahap ini pengguna akan melakukan pengujian untuk mengetahui apakah aplikasi telah berhasil dibangun dan diselesaikan dengan baik tanpa adanya *error* ataupun *bug* yang terjadi. Tahapan ini sesuai dengan fase pada SDLC, khususnya fase pengujian.
5. Rekomendasi Perbaikan
   Pada tahap ini akan dilakukan perbaikan-perbaikan terhadap aplikasi dengan berdasarkan pada pengujian yang telah dilakukan oleh pengguna.

.

HASIL DAN PEMBAHASAN

Penelitian ini dibuat untuk mengimplementasikan metode *Vector Space Model (VSM)* dalam pencarian dokumen. Metode yang digunakan dalam pencarian dokumen terdiri dari berbagai metode, namun metode *Vector Space Model* ini yang paling sering digunakan apalagi tingkat presisinya cukup baik. Hal ini didukung oleh beberapa penelitian sebelumnya, yaitu (Ningtyas *et al*., 2018; Budisetyo *et al*., 2018; Wicaksono *et al*., 2015; Amin, 2012). Berdasarkan hal tersebut, pemilihan metode ditetapkan dan digunakan dalam penelitian ini untuk pencarian dokumen uji monitoring di laboratorium yang menjadi tempat studi kasus. Dikarenakan penelitian ini akan mengimplementasikan metode *Vector Space Model* (VSM), maka akan dibuatkan aplikasi dengan menggunakan metode tersebut. Tahapan dalam pembuatan aplikasi tersebut mengikuti fase dalam metodologi SDLC, yang terdiri dari: analisis kebutuhan, perancangan sistem, implementasi dan pengujian.

Analisis Kebutuhan

Untuk memudahkan pencarian dokumen sering digunakan fitur/menu "pencarian" atau "*search*" dalam suatu aplikasi/sistem. Demikian juga yang dihadapi oleh Lab BPJN di Jayapura. Olehnya untuk memudahkan pencarian dokumen tersebut, maka aplikasi akan mengimplementasikan metode VSM sebagai solusi. Aplikasi akan memiliki 2 pengguna, yaitu pengguna administrasi dan Kepala Seksi dengan tanggung jawab yang sama.

Pada bagian ini pula akan dijelaskan mengenai penerapan metode VSM dengan menggunakan 5 contoh kalimat dari 5 laporan yang telah ada sebelumnya untuk melakukan perbandingan hasil pencarian dengan kata kunci yang dimasukkan oleh pengguna. Kelima contoh kalimat ini kemudian akan disebut sebagai dokumen uji.

Tabel 1. Dokumen Uji

No	Laporan	Laporan yang sudah ada
1.	Laporan-1	Pihak kedua melakukan pengujian bahan material di laboratorium
2.	Laporan-2	PIHAK PERTAMA memberikan material untuk diuji
3.	Laporan-3	Pengujian bahan MATERIAL dilakukan oleh PIHAK KEDUA dan akan diuji untuk beberapa hari ke depan
4.	Laporan-4	Pihak Kedua menyelesaikan pengujian di laboratorium
5.	Laporan-5	Hasil uji material telah didapatkan

Kelima kalimat yang terdapat pada Tabel 1 merupakan dokumen uji dan akan dibandingkan dengan *keyword* atau kata kunci yang dimasukkan oleh pengguna. Kata kunci atau *keyword* tersebut akan disebut sebagai dokumen pembanding. *Keyword* tersebut dapat dilihat pada Tabel 2.

Tabel 2. Kata Kunci

No	Pengguna	Kata Kunci
1.	Pengguna-1	Pengujian bahan material di laboratorium

Selanjutnya, tahapan dalam metode VSM (Yayasan Multimedia Nusantara & Xeratic, 2021; Amin, 2012) terdiri dari:

1. Tahap *Preprocessing*

Dalam tahapan ini, terdapat 4 langkah yang dilakukan, yakni:

a. Case Folding

Langkah ini akan melakukan konversi seluruh kata yang ada dalam kalimat menjadi suatu bentuk dasar yang berupa *lower case* agar mudah untuk diproses.

Tabel 3. *Case Folding*

No	Sebelum	Sesudah
1.	Pihak	pihak
2.	PIHAK PERTAMA	pihak pertama
3.	Pengujian	pengujian
4.	MATERIAL	material
5.	PIHAK KEDUA	pihak kedua

Tabel 3 menjelaskan perubahan kata-kata yang ditulis dalam huruf besar atau campuran huruf besar/kecil ke huruf kecil semua, sebagai contoh kata PIHAK PERTAMA akan diubah ke pihak pertama.

b. Filtering

Tahap ini akan melakukan penyaringan terhadap keseluruhan isi teks pada kalimat dengan menghapus karakter-karakter dalam kalimat yang tidak mempunyai makna dan arti khusus. Pada tahap ini umumnya kata yang dihapuskan adalah kata-kata penghubung yang sering memiliki tidak memiliki arti atau makna khusus.

Tabel 4. *Filtering*

Filtering					
di	oleh	untuk	dan	akan	yang

Tabel 4 menjelaskan mengenai kata-kata yang tidak memiliki makna khusus dalam suatu kalimat sehingga kata-kata ini dapat dihapus sebagai hasil proses *filtering*. Sebagai contoh: “Hasil uji material telah didapatkan” akan menjadi “hasil uji material telah dapatkan”. Jadi kata “di” yang tidak memiliki arti dapat dihapus/dihilangkan.

c. Stemming

Tahapan ini akan mengubah suatu kata imbuhan yang ada dalam kalimat menjadi kata dasar tanpa memiliki *suffix* dan *preffix* sehingga menjadi seperti Tabel 5. Berdasarkan Tabel 5, kata-kata berimbuhan pada Tabel 1 kolom 3 akan dijadikan sebagai kata dasar. Sebagai contoh: “hasil uji material telah dapatkan” sebagai hasil dari proses *filtering* akan diubah menjadi “hasil uji material telah dapat”. Tabel 5 merupakan kata-kata berimbuhan pada Tabel 1 kolom 3 yang diubah ke kata dasar.

Tabel 5. *Stemming*

Stemming	
Sebelum	**Sesudah**
melakukan	laku
dilakukan	laku
pengujian	uji
diuji	uji
memberikan	beri
kedepan	depan
beberapa	berapa
menyelesaikan	selesai
dapatkan	dapat

d. Tokeninzing

Pada tahap *tokenizing* ini, setiap kata yang ada dalam kalimat akan dipecah menjadi kata-kata yang dapat berdiri sendiri. Hasil dari tahap *tokenizing* adalah kata-kata terpisah yang dapat menyusun suatu kalimat.

Tabel 6. *Tokenizing*

No	Laporan	Laporan yang sudah ada	*Tokenizing*
1.	Laporan-1 (L1)	pihak kedua melakukan pengujian bahan material di laboratorium	[‘pihak’], [‘dua’], [‘laku’], [‘uji’], [‘bahan’], [‘material’], [‘laboratorium’]
2.	Laporan-2 (L2)	pihak pertama memberikan material untuk diuji	[‘pihak’], [‘pertama’], [‘beri’], [‘material’], [‘uji’]
3.	Laporan-3 (L3)	pengujian bahan material dilakukan oleh pihak kedua dan akan diuji untuk beberapa hari ke depan	[‘uji’], [‘bahan’], [‘material’], [‘laku’], [‘pihak’], [‘dua’], [‘uji’], [‘berapa’], [‘hari’], [‘depan’]
4.	Laporan-4 (L4)	Pihak kedua menyelesaikan pengujian di laboratorium	[‘pihak’], [‘dua’], [‘selesai’], [‘uji’], [‘laboratorium’]
5.	Laporan-5 (L5)	Hasil uji material telah didapatkan	[‘hasil’], [‘uji’], [‘material’], [‘telah’] [‘dapat’]

Tabel 6 menjelaskan pemisahan kalimat menjadi kata per kata yang berdiri sendiri. Misalnya kalimat “hasil uji material telah dapat” akan diubah menjadi “hasil”, “uji”, “material”, “telah” dan “dapat”.

2. Tahap *Processing*

Tahapan ini dilakukan untuk menghitung bobot dari setiap kata yang ada dalam dokumen dengan menggunakan algoritma *Vector Space Model*. Tahap ini akan terdiri dari beberapa proses, yaitu: Pembobotan yang terdiri dari: *Term Frequency* (TF), *Document Frequency* (DF), *Inverse Document Frequency* (IDF) dan *Term Frequency – Inverse Document Frequency* (TF-IDF) lalu kemudian akan dilanjutkan menghitung *Dot Product*,

menghitung panjang *Vector, Cosine Similarity* dan mengurutkan berdasarkan peringkat (Siregar, 2017).

a. Pembobotan

Pembobotan dimulai dengan menghitung *Term Frequency* (TF), yaitu berapa kali suatu kata muncul dalam suatu dokumen.

Tabel 7. *Term Frequency*

Term	*Keyword*	L1	L2	L3	L4	L5
uji	1	1	1	1	1	1
bahan	1	1	0	1	0	0
material	1	1	1	1	0	1
laboratorium	1	1	0	0	1	0
pihak	0	1	1	1	1	0
dua	0	1	0	1	1	0
laku	0	1	0	1	0	0
pertama	0	0	1	0	0	0
beri	0	0	1	0	0	0
berapa	0	0	0	1	0	0
hari	0	0	0	1	0	0
depan	0	0	0	1	0	0
selesai	0	0	0	0	1	0
hasil	0	0	0	0	0	1
telah	0	0	0	0	0	1
dapat	0	0	0	0	0	1

Tabel 7 menjelaskan jumlah kata yang muncul pada masing-masing dokumen yang terdapat pada tabel 1. Misalnya kata “uji” muncul sebanyak 1 kali pada dokumen L1 sampai L5.

Hasil dari Tabel 7 akan dilanjutkan dengan menghitung *document frequency* (DF) yang bermanfaat untuk menjumlahkan *term* yang ada pada setiap kategori pada Tabel 7. Perhitungan DF hanya dilakukan pada kata-kata yang terdapat pada setiap kalimat laporan saja dan tidak menghitung kalimat yang muncul pada *keyword.*

Tabel 8. *Document Frequency*

Term	DF
uji	5
bahan	2
material	4
laboratorium	2
pihak	4
dua	3
laku	2
pertama	1
beri	1
berapa	1
hari	1
depan	1
selesai	1
hasil	1
telah	1
dapat	1

Tabel 8 hanya menjumlah seluruh TF pada seluruh dokumen yang ada. Misalnya kata "uji" muncul pada dokumen L1 sampai dokumen L5 sehingga DF- nya menjadi 5. Setelah DF dan TF diperoleh, maka selanjutnya akan dihitung *Inverse Document Frequency* untuk mendapatkan bobot nilai dari setiap kata atau *term* dengan menggunakan persamaan $IDF = \log \frac{n}{DF}$ sehingga hasilnya terlihat pada Tabel 9.

Tabel 9. *Inverse Document Frequency*

Term	IDF
uji	0
bahan	0,3979
material	0,0969
laboratorium	0,3979
pihak	0,0969
dua	0,2218
laku	0,3979
pertama	0,6989
berikan	0,6989
berapa	0,6989
hari	0,6989
depan	0,6989
selesai	0,6989
hasil	0,6989
telah	0,6989
dapat	0,6989

Tabel 9 merupakan hasil dari persamaan yang digunakan dengan n adalah jumlah dokumen atau data teks. Nilai IDF yang diperoleh akan digunakan untuk mencari nilai TF-IDF dengan persamaan $\boldsymbol{TF-IDF = TFxIDF}$, dengan hasil seperti pada Tabel 10.

Tabel 10. *Term Frequency – Inverse Document Frequency*

Term	*Keyword*	L1	L2	L3	L4	L5
Uji	0	0	0	0	0	0
Bahan	0,3979	0,3979	0	0,3979	0	0
Material	0,0969	0,0969	0,0969	0,0969	0	0,0969
Laboratorium	0,3979	0,3979	0	0	0,3979	0
Pihak	0	0,0969	0,0969	0,0969	0,0969	0
Dua	0	0,2218	0	0,2218	0,2218	0
Laku	0	0,3979	0	0,3979	0	0
Pertama	0	0	0,6989	0	0	0
Berikan	0	0	0,6989	0	0	0
Berapa	0	0	0	0,6989	0	0
Hari	0	0	0	0,6989	0	0
Depan	0	0	0	0,6989	0	0
Selesai	0	0	0	0	0,6989	0
Hasil	0	0	0	0	0	0,6989
Telah	0	0	0	0	0	0,6989
Dapat	0	0	0	0	0	0,6989

Tabel 10 merupakan tabel yang berisikan hasil perkalian antara TF dan IDF untuk setiap dokumen, yaitu dokumen L1 hingga dokumen L5.

b. Menghitung *Dot Product*

Setelah pembobotan selesai dilakukan, maka akan dihitung *dot product* (Amin, 2012) menggunakan persamaan $TF - IDF(DP)\ x\ TF - IDF(DU(n))$ sehingga diperoleh hasil pada Tebal 11.

Tabel 11. *Dot Product*

Term	***Dot Product***				
	L1	**L2**	**L3**	**L4**	**L5**
Uji	0	0	0	0	0
Bahan	0,1583	0	0,1583	0	0
Material	0,0093	0,0093	0,0093	0	0,0093
laboratorium	0,1583	0	0	0,1583	0
Pihak	0	0	0	0	0
Kedua	0	0	0	0	0
lakukan	0	0	0	0	0
Pertama	0	0	0	0	0
berikan	0	0	0	0	0
Berapa	0	0	0	0	0
Hari	0	0	0	0	0
depan	0	0	0	0	0
selesai	0	0	0	0	0
hasil	0	0	0	0	0
Telah	0	0	0	0	0
dapat	0	0	0	0	0
Total	**0,3259**	**0,0093**	**0,1676**	**0,1583**	**0,0093**

Tabel 11 merupakan tabel yang berisikan hasil hitungan *dot product* untuk setiap dokumen dari dokumen L1 hingga dokumen L5.

c. Menghitung Panjang Vektor

Selanjutkan akan dihitung panjang vektor dengan menggunakan persamaan berikut ini, yakni: (Fauziah et al., 2019)

$$TF - IDF(DP)\ x\ TF - IDF(DU(n))\text{^}2$$

Tabel 12. ***Panjang*** *Vector*

Term	***Keyword***	**L1²**	**L2²**	**L3²**	**L4²**	**L5²**
Uji	0	0	0	0	0	0
Bahan	0,1583	0,1583	0	0,1583	0	0
Material	0,0093	0,0093	0,0093	0,0093	0	0,0093
Laboratorium	0,1583	0,1583	0	0	0,1583	0
Pihak	0	0,0093	0,0093	0,0093	0,0093	0
Dua	0	0,0491	0	0,0491	0,0491	0
Lakukan	0	0,1583	0	0,1583	0	0
Pertama	0	0	0,4884	0	0	0
Beri	0	0	0,4884	0	0	0
Berapa	0	0	0	0,4884	0	0
Hari	0	0	0	0,4884	0	0
Depan	0	0	0	0,4884	0	0
Selesai	0	0	0	0	0,4884	0
Hasil	0	0	0	0	0	0,4884
Telah	0	0	0	0	0	0,4884
Dapat	0	0	0	0	0	0,4884
Total	0,3259	0,5426	0,9954	1,8495	0,7051	1,4745

Tabel 12 merupakan hasil perhitungan panjang vektor dari dokumen L1 hingga L5.

d. Menghitung *Cosine Similarity*

Cosine similarity dihitung dengan menggunakan persamaan berikut (Yasni *et al*., 2018):

$$sim\,(d_j) = \frac{\sum_{i}^{n}=1\ \omega_{i,j}\ \ \omega_{i,q}}{\sqrt{\sum_{i}^{n}=1\ \omega_{i,j}}\ x\ \sqrt{\sum_{i}^{n}=1\ \omega_{i,q}}}$$

sehingga diperoleh hasil L1= 77.52%, L2= 1.63%, L3= 21.59%, L4= 33.02% dan L5= 1.34%

e. Mengurutkan berdasarkan peringkat

Nilai yang didapatkan tersebut akan diurutkan berdasarkan nilai tertinggi ke nilai terendah yang dapat dilihat pada Tabel 13 berikut ini **Invalid source specified.**:

Tabel 13. Mengurutkan Berdasarkan Peringkat

No	Pembanding	Penguji	*Cosine Similarity*	Persentase	Peringkat
1.	Kata Kunci	Laporan 1	0,7752	77,52%	#1
2.	Kata Kunci	Laporan 2	0,0163	1,63%	#4
3.	Kata Kunci	Laporan 3	0,2159	21,59%	#3
4.	Kata Kunci	Laporan 4	0,3302	33,02%	#2
5.	Kata Kunci	Laporan 5	0,01341	1,34%	#5

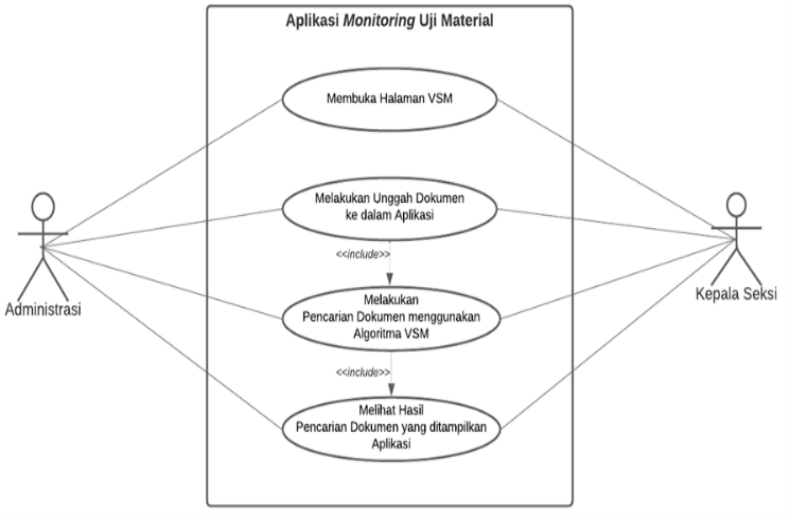
3. Tahap *Post Processing*

Pada tahap ini akan dilakukan perhitungan untuk menentukan nilai dari presisi dan *recall*. Nilai – nilai tersebut berguna untuk mengetahui apakah hasil pencarian informasi yang direspons oleh aplikasi relevan pada pengguna atau tidak **Invalid source specified.**

Berdasarkan 40 dokumen yang diujikan dan kata kunci yang dimasukkan “agregat kelas a” diperoleh presisi sebesar 0.83 dan *recall* sebesar 1.

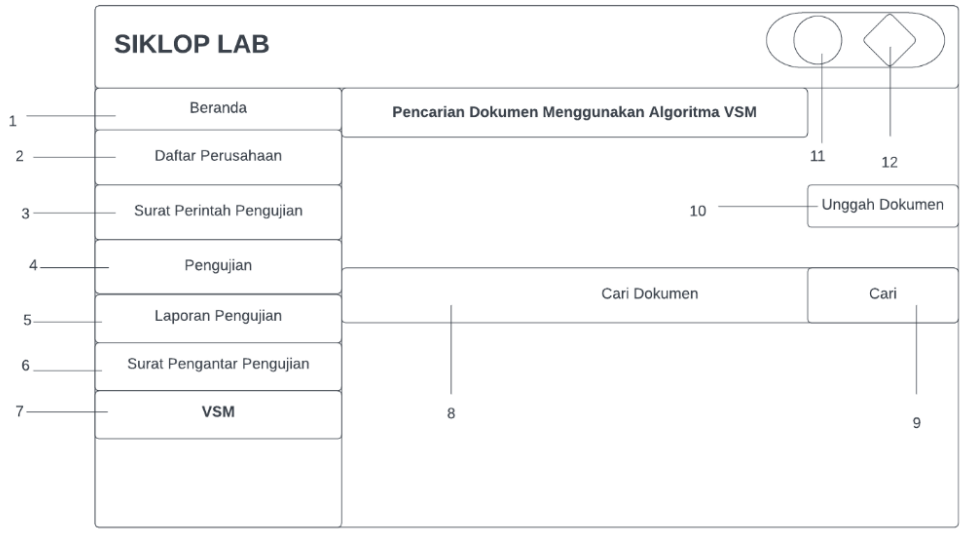
Perancangan Sistem

Sesuai dengan analisis kebutuhan sebelumnya dan perhitungan terhadap metode Vector Space Model, maka dirancanglah aplikasi yang mengimplementasikan metode VSM tersebut. Aplikasi nantinya akan digunakan oleh 2 pengguna, yang digambarkan dalam bentuk *use case diagram*.Pemodelan *use case diagram* dilakukan untuk menggambarkan mengenai bagaimana pengguna dan sistem akan berinteraksi (Heriyanto, 2018). *Use case diagram* akan menggambarkan interaksi antara pengguna dan sistem, sebagai berikut:



Gambar 1. *Use Case Diagram*

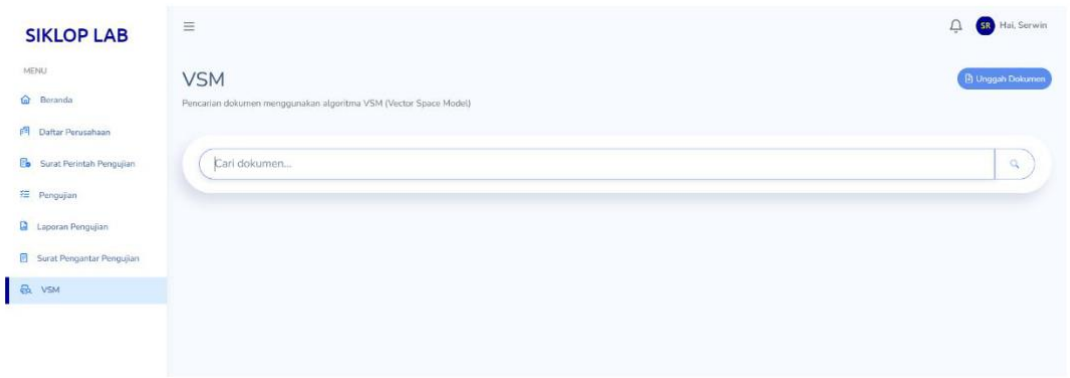
Seperti yang terlihat pada Gambar 1, akan ada 2 pengguna pada aplikasi yang dibuat, yaitu administrasi dan Kepala Seksi. Pengguna administrasi dan Kepala Seksi dapat melakukan unggah dokumen, melakukan pencarian dokumen dan melihat hasilnya. Gambaran antarmuka dirancang sesuai dengan analisis kebutuhan pengguna dapat dilihat pada gambar 2. Angka 1-angka 12 merupakan penjelasakan dari setiap tombol yang akan digunakan.



Gambar 2. *Rancangan antarmuka aplikasi*

Implementasi

Dari perancangan yang telah dilakukan, aplikasi dibuat menggunakan framework Laravel 8 dan bahasa pemrograman PHP versi 8.0. Sedangkan untuk pengelolaan basis data digunakan MySQL versi 8 sebagai DBMS. Hasil dari implementasi terhadap rancangan yang dibuat adalah:



Gambar 3. Hasil implementasi a*ntarmuka aplikasi*

Pengujian

Setelah implementasi dilakukan bahwa perlu dilakukan pengujian terhadap aplikasi dan metode yang ada. Pengujian dilakukan terhadap fungsi dari aplikasi dan juga metode VSM yang digunakan. Pengujian dilakukan terhadap 40 dokumen dan *query* yang digunakan adalah "agregat kelas a". Hasil pengujian terhadap fungsi dari aplikasi dan metode diperoleh hasil bahwa semua fungsi berjalan dengan baik dan metodre VSM juga berhasil diterapkan pada aplikasi. Sedangkan pencarian dokumen terhadap *query* yang ada menunjukkan bahwa terdapat 33 dokumen yang sesuai sehingga hasil presisi 0.83. Demikian juga dengan *recall*

diperoleh nilai 1 karena dokumen yang ditemukan sesuai dengan data dokumen yang tersimpan.

KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan bahwa metode *Vector Space Model* berhasil diiimplementasikan pada suatu aplikasi dan aplikasi berjalan dengan baik tanpa adanya *error* dan *bug*. Selain itu pula, pengujian metode *Vector Space Mode* terhadap pencarian 40 dokumen yang dimiliki oleh laboratorium BPJN Jayapura memberikan hasil dokumen sesuai keinginan pengguna, yaitu penelitian ini menghasilkan nilai presisi sebesar 0.83 dan nilai *recall* sebesar 1 berdasarkan kata kunci "agregat kelas a" terhadap 40 dokumen yang ada.

DAFTAR PUSTAKA

Amin, F. (2012). Sistem Temu Kembali Informasi dengan Metode Vector Space Model. *Jurnal Sistem Informasi Bisnis*, *2*(2), 78–83. https://doi.org/10.21456/vol2iss2pp078-083

Bahri, S. (2020). Aplikasi Pencarian Bahan Pustaka Di Perpustakaan. *Jurnal Informatika Merdeka Pasuruan*, *5*(5), 27–37.

Budisetyo, I. H., Brata, A. H., & Dewi, R. K. (2018). Pengembangan Aplikasi Pencarian Film (Rekonifi) Dengan Metode Vector Space Model (VSM) Berbasis Android. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *2*(7), 2834–2842.

Dahlan, A. (2017). *Merancang Aplikasi Perpustakaan Menggunakan SDLC*. CV. Sefa Bumi Persada.

Fauziah, S., Sulistyowati, D. N., & Asra, T. (2019). Optimasi Algoritma Vector Space Model Dengan Algoritma K-Nearest Neighbour Pada Pencarian Judul Artikel Jurnal. *Journal of Computing and Information System*, *15*(1), 21–25.

Heriyanto, Y. (2018). Perancangan Sistem Informasi Rental Mobil Berbasis Web Pada PT. APM Rent Car. *Jurnal Intra-Tech.*, *2*(2), 64–77.

Makmun, W. W., Ningrum, I. P., & Sajiah, M. A. (2022). Penerapan Vector Space Model (VSM) pada Pencarian Artikel Arkeologi. *Semantik*, *8*(8), 69–76.

Muktyas, B. I., & Abdillah, A. A. (2013). Implementasi Vector Space Model untuk Pencarian Dokumen. *Seminar Nasional Matematika Dan Pendidikan Matematika 2013*, 1–7.

Ningtyas, P. T. W., Fitriansyah, & Oraseotiana, D. M. (2018). Aplikasi Mesin Pencarian Alat Elektronik Berbasis Web menggunakan Metode Vector Space Model. *Journal of Big Data Analytic and Artificial Intelligence*, *4*(1), 29–34.

Salmon, S., Paseru, D., & Kumenap, V. (2020). Implementasi Metode Vector Space Model pada Serach Engine Perpustakaan. Prosiding Seminar Nasional, Sisfotek, 84-92. *Prosiding Seminar Nasional, Sisfotek*, 84–92.

Siregar, A. M. (2017). Perbandingan Pembobotan Kata Dalam Sistem Temu Balik Informasi. *Information System Application*, *2*(1), 11–15.

Sulianta, F. (2010). *Search Engine Pilihan untuk Berbagai Kebutuhan*. PT. Elex Media Komputindo.

Wicaksono, V. B., Saptono, R., & Widya, S. (2015). Analisis Perbandingan Metode Vector Space Model dan Weighted Tree Similarity dengan Cosine Similarity pada kasus Pencarian Informasi Pedoman Pengobatan Dasar di Puskesmas. *ITSMART: Jurnal Teknologi Dan Informasi*, *4*(2), 73–83.

Yasni, L., Subroto, I. M., & Haviana, S. F. (2018). Implementasi Cosine Similarity Matching Dalam Penentuan Dosen Pembimbing Tugas Akhir. *Jurnal Ilmiah Teknik Elektro*, *20*(1), 22–28.

Yayasan Multimedia Nusantara & Xeratic. (2021). *DQ Lab. Retrieved from DQ La*. https://dqlab.id/tahapan-text-preprocessing-dalam-teknik-pengolahan-data.