# Text Similarity Measurement using PGTRGM Module Function PostgreSQL

**I Made Indrayana[a1], I Made Sukarsa[a2], Kadek Suar Wibawa[a3]**
[a]Program Studi Teknologi Informasi, Fakultas Teknik, Universitas Udayana
Bukit Jimbaran, Bali, Indonesia, telp. (0361) 701806
e-mail: [1]indrayanakadek@gmail.com,[2]sukarsa@unud.ac.id, [3]suar_wibawa@unud.ac.id

***Abstrak***

Perkembangan pemanfaatan teknologi dalam pengolahan informasi memberikan dampak besar dalam memudahkan manusia untuk menemukan suatu data yang spesifik dalam kumpulan data yang besar. Algoritma pencarian informasi dalam bentuk teks dapat memanfaatkan metode pengukuran kemiripan teks untuk membandingkan suatu teks atau informasi yang dicari dengan teks lain dalam kumpulan data yang besar. Penelitian ini melakukan pengukuran nilai kemiripan kumpulan kata dengan kumpulan kata lain menggunakan modul yang tersedia pada basis data PostgreSQL. Algoritma pengukuran kemiripan teks menggunakan *extension* modul pg-trgm yang berbasis n-gram pada PostgreSQL. Pengujian dilakukan dengan membandingkan dua teks berbeda dengan menggunakan *function* yang terdapat pada modul pg-trgm. Hasil pengujian menggunakan sepuluh skenario pengujian diperoleh sebanyak lima hasil pengukuran dengan nilai kemiripan yang sama antara *function similarity* dengan *word similarity*. Perhitungan dengan algoritma *function word similarity* menghasilkan lima hasil pengukuran yang lebih baik.

***Kata kunci:*** *Pencarian Informasi, Pengukuran Kemiripan Teks, Pg-trgm, PostgreSQL*

***Abstract***

*The development of information processing technology has a major impact in making it easier for humans to find specific data in large data sets. Information search algorithms in the form of text can take advantage of the method of measuring the similarity to compare a text sought with other texts in large data sets. This study measures the similarity value of a word set with another word set using the available modules in the PostgreSQL database. The text similarity measurement algorithm uses the extension pg-trgm module based on n-grams in PostgreSQL. The test is carried out by comparing two different texts using the functions contained in the pg-trgm module. The test results using ten test scenarios obtained five measurement results with the same similarity value between function similarity and word similarity. Calculation with algorithm function word similarity produces five better measurement results.*

***Keywords:*** *Pg-trgm, PostgreSQL, Searching for Information, Text Similarity Measurement*

## 1. Pendahuluan

Perkembangan pemanfaatan teknologi dalam pengolahan informasi memberikan dampak besar dalam memudahkan manusia untuk menemukan suatu data yang spesifik dalam kumpulan data yang besar. Proses pencarian informasi dalam bentuk teks dapat memanfaatkan metode pengukuran kemiripan teks untuk membandingkan suatu teks atau informasi yang dicari dengan teks lain dalam kumpulan data yang besar. Untuk memperoleh informasi dengan efisien dapat menerapkan metode yang banyak ditemui seperti teknik *pattern matching*, jaccard, pengukuran kemiripan kalimat dan metode lainnya [1]. Penyaringan sekumpulan data dapat memberikan rekomendasi hasil yang dibutuhkan pengguna [2]. Teknologi pengolahan *text mining* sebagai teknik pengolahan data teks dengan melakukan pengukuran tingkat kemiripan sebuah teks dokumen yang berkaitan dengan pencarian kalimat pada sekumpulan informasi

atau data yang besar dan banyak [3]. Untuk menemukan data dan informasi tidak dapat dipisahkan dari pengenalan pola [4].

Penelitian lain yang telah dilakukan dengan menggunakan metode jaccard index dalam pencarian informasi untuk melakukan perhitungan *similarity keyword* pada artikel yang ada pada basis data. Perhitungan dengan menggunakan jumlah *intersect* dan jumlah *union* dari sebuah kata atau kalimat tanpa mengurangi jumlah karakter yang ada atau disebut sebagai N-gram. *Jaccard similarity* menghasilkan hasil pencarian atau pencocokan yang lebih akurat dengan waktu proses yang lebih lama dibandingkan dengan pencocokan pola MySQL. Pencocokan teks dengan menggunakan koefisien jaccard lebih cocok diimplementasikan untuk teks yang sangat pendek [5]. Penggunaan n-gram, *pattern matching*, *threshold boundary* dari metode yang digunakan dapat meningkatkan nilai presisi dan *recall* [6]. Penelitian lain dengan menggunakan metode *jaccard similarity* dan *cosine similarity* dapat melakukan pencocokan kalimat dengan membatasi kalimat pencocokan dalam format kata kunci sehingga tidak memunculkan kalimat lain yang dapat menurunkan nilai *similarity* [7].

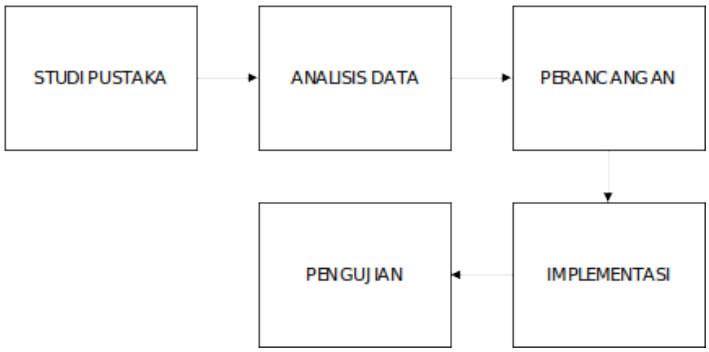
Penelitian yang dilakukan berfokus untuk melakukan pengukuran nilai kemiripan kumpulan kata dengan kumpulan kata lain menggunakan modul yang tersedia pada basis data PostgreSQL. Algoritma pengukuran kemiripan teks menggunakan *extension* modul pg-trgm yang berbasis n-gram pada PostgreSQL. Pengujian dilakukan dengan membandingkan dua teks berbeda dengan menggunakan *function* yang terdapat pada modul pg-trgm. Teks yang digunakan dalam pengujian disusun kedalam format kata kunci.

## 2. Metode Penelitian

Penelitian menggunakan metode *waterfall* sederhana yang dimulai dengan tahapan analisa kebutuhan, perancangan alur, implementasi rancangan algoritma, hingga akhirnya pengujian rancangan yang digambarkan secara rinci dalam bentuk diagram alur, diagram umum dan diagram proses *preprocessing*.

### 2.1. Alur Proses Penelitian

Alur proses penelitian merupakan tahapan umum yang dilalui selama melakukan kegiatan penelitian. Alur proses penelitian dimulai dengan tahap studi Pustaka hingga berakhir dengan pengujian sistem hasil implementasi. Alur proses penelitian umum secara rinci dijabarkan dalam Gambar 1.

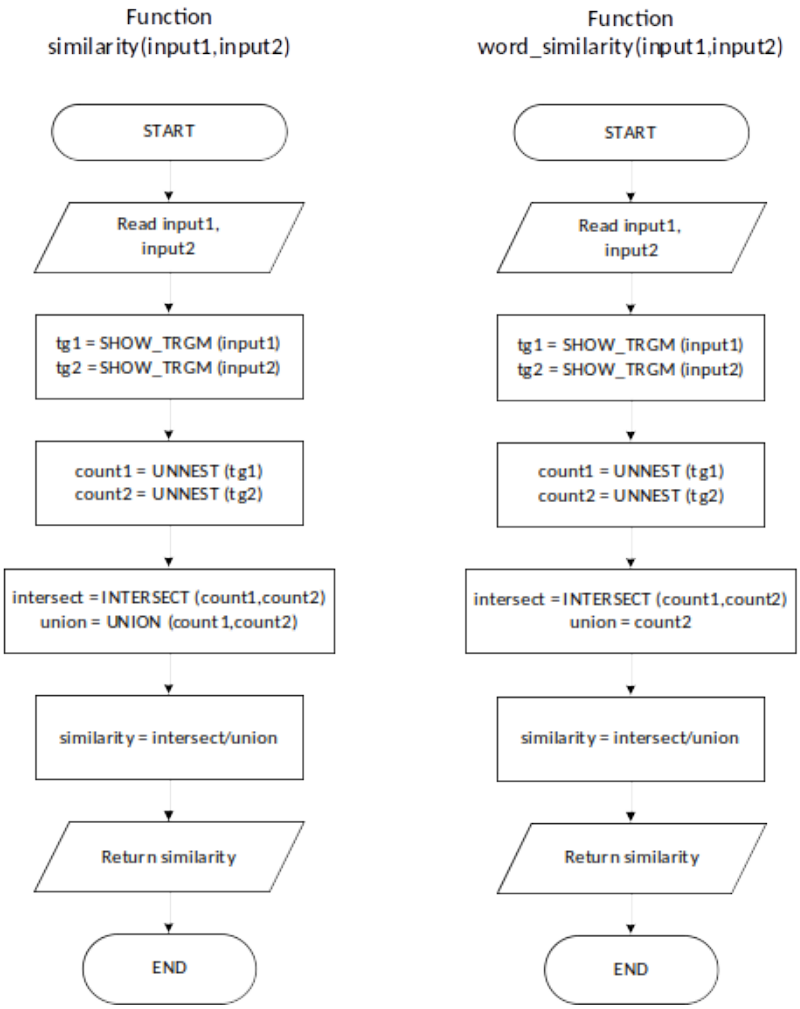



Gambar 1. Alur Penelitian

Alur pengerjaan sistem yang dilakukan dimulai dari studi pustaka dari berbagai sumber, kemudian analisis data hasil studi pustaka dan penyusunan data uji serta instrumen data dilanjutkan proses perancangan sistem perhitungan tingkat kemiripan kalimat, implementasi sistem dan berakhir dengan pengujian sistem serta analisa hasil pengujian.

### 2.2. Diagram Proses Algoritma

Diagram proses algoritma merupakan gambaran proses sistem secara garis besar yang dimulai dari proses menerima input berupa teks hingga proses perhitungan nilai *similarity*. *Flowchart* proses algoritma dapat dilihat pada Gambar 2.

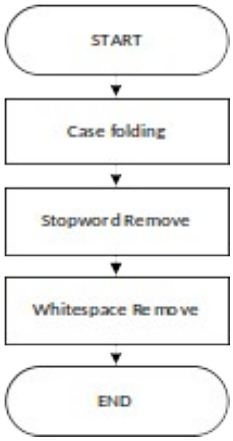



Gambar 2. Diagram Proses Algoritma

Diagram proses algoritma terdiri atas tahapan *read input string* pertama dan kedua, *extract* tri-gram *string input*, *apply unnest function query* pada hasil *set* tri-gram, hitung nilai *intersect* dan terakhir hitung nilai *similarity* berdasarkan *formula*. Perbedaan utama dari *function similarity* dan *word similarity* ada pada pembagi *intersect* dalam perhitungan *similarity* dimana *function similarity* menggunakan *union* dari tri-gram *string* pertama dan kedua sedangkan *word similarity* hanya menggunakan tri-gram *string* kedua.

### 2.3. Diagram Alur *Preprocessing*

Alur proses *preprocessing* terdiri atas proses *case folding* (*lowercase*), *stopword remove*, dan diakhiri dengan proses *whitespace remove*. *Flowchart* alur *preprocessing* dapat dilihat pada Gambar 3.




Gambar 3. Diagram Alur *Preprocessing*

*Case folding* merupakan tahapan mengubah semua *input* dalam bentuk teks menjadi bentuk huruf kecil atau *lowercase*, *stopword remove* merupakan tahapan membuang kata yang kurang penting, *whitespace remove* untuk membuang penggunaan *space* yang berlebihan.

## 3. Studi Literatur

Kajian pustaka merupakan tahapan mengumpulkan informasi dalam bentuk konsep dan teori mengenai penelitian yang telah dilakukan yang memiliki keterkaitan dengan penelitian yang dilakukan untuk dijadikan pedoman penelitian. Kajian pustaka pada penelitian ini bersumber dari *paper* jurnal, *student paper*, dan internet.

### 3.1. Text Mining

*Text mining* merupakan sebuah proses ekstraksi atau pencocokan pola dengan basis dari sejumlah besar data pengetahuan. *Text mining* berperan penting dalam proses *data mining* yang pengaplikasinya dapat dilihat proses ekstraksi pola-pola data. Perbedaan utama antara *data mining* dengan *text mining* adalah perbedaan *source* infromasi. *Data mining* melakukan proses ekstraksi *pattern* pada *database* yang telah disusun dengan format tertentu sedangkan *text mining* proses ekstraksi bersumber dari data dalam bentuk teks atau bahasa alami [8].

### 3.2. PostgreSQL

*Database* PostgreSQL merupakan salah satu teknologi DBMS *open source* yang mendukung sebagian besar *query* SQL, menawarkan fitur untuk kueri kompleks, *concurrent* control, *triggers*, *view* dan penambahan fitur ekstensi *function*, tipe data dan bahasa *procedural* [9]. Modul pg-trgm merupakan salah satu ekstensi yang disediakan dalam *database* PostgreSQL untuk kebutuhan *index* dan *text similarity*.

### 3.3. N-gram

Metode n-gram merupakan konsep pemisahan atau pembagian string dengan jumlah tertentu pada teks. Urutan n-gram selanjutnya akan bergeser sesuai dengan nilai yang bervariasi tergantung pendekatan yang digunakan. Jenis n-gram dapat berupa uni-gram dengan panjang n bernilai 1, bi-gram dengan panjang n bernilai 2, tri-gram dengan panjang n bernilai 3 dan seterusnya. Perhitungan n-gram menghasilkan tingkat kesamaan basis pengetahuan yang sesuai dengan teks yang diinputkan. Kesalahan pada string hanya akan menghasilkan sedikit perbedaan tergantung dari jenis n-gram [10].

### 3.4. Modul pg-trgm

Modul pg-trgm pada *database* PostgreSQL menyediakan *function* untuk menghitung *similarity* pada teks alfanumerik ASCII dengan menggunakan algoritma n-gram atau secara spesifik tri-gram, serta menyedikan *class index* yang dapat digunakan dalam proses pencarian cepat untuk *string* pada *database* PostgreSQL [11]. Fungsi yang disediakan modul pg-trgm secara rinci dijelaskan pada Tabel 1.

Tabel 1. Function Modul pg-trgm

| Function | Return | Description |
|---|---|---|
| similarity(text, text) | real | menampilkan nilai kemiripan dua buah string (0-1) |
| show_trgm(text) | text[] | menampilkan array trigram dari sebuah string |
| show_limit() | real | menampilkan nilai threshold yang digunakan |
| set_limit(real) | real | mengatur nilai threshold yang akan digunakan |
| word_similarity(text,text) | real | Menampilkan nilai kemiripan terbaik dari string pertama dengan substring kedua (0-1) |

Function modul pg-trgm yang digunakan untuk menampilkan token tri-gram adalah show_trgm dengan *return value* berupa *array* data dengan tipe *text*. Menghitung nilai kemiripan dari *string* dapat menggunakan *function word similarity* maupun *similarity*.

### 3.5. Perhitungan Nilai Similarity

Perhitungan nilai *similarity* terbagi menjadi dua *formula* yaitu *formula* menggunakan *function word similarity* dan *similarity*. Perhitungan nilai *word similarity* modul pg-trgm antara dua *string* didapatkan dari hasil pembagian antara irisan tri-gram string pertama dan *string* kedua dengan setiap *substring* dari *string* kedua. Persamaan dari perhitungan nilai *word similarity* dapat dilihat pada Persamaan 1 berikut.

$$similarity(A,B) = \frac{|trigram(A) \cap trigram(B)|}{|trigram(B)|} \quad (1)$$

A dan B merupakan kalimat atau kumpulan kata yang akan diukur nilai kemiripannya, |A ∩ B| merupakan *intersect* kumpulan tri-gram kalimat A dan tri-gram kalimat B. Nilai *similarity* dihitung berdasarkan hasil pembagian dari irisan dari kumpulan tri-gram set A dan B dengan tri-gram set B. Perhitungan nilai *similarity* modul pg-trgm antara dua *string* didapatkan dari hasil pembagian antara irisan tri-gram *string* pertama dan *string* kedua dengan *union* dari kedua *string*. Persamaan dari perhitungan nilai *similarity* dapat dilihat pada Persamaan 2 berikut.

(2)

|A ∩ B| merupakan irisan (*intersect*) dari kumpulan tri-gram set A dan tri-gram set B sedangkan |A ∪ B| merupakan kesatuan (*union*) dari kumpulan tri-gram set A dan tri-gram set B. Nilai *similarity* dihitung berdasarkan hasil pembagian dari *intersect* dari kumpulan tri-gram set A dan B dengan *union* tri-gram set A dan B.

## 4. Hasil dan Pembahasan

Bagian yang menjelaskan hasil implementasi rancangan dan hasil uji coba pengukuran tingkat kemiripan teks dengan menggunakan modul pg-trgm pada PostgreSQL yang telah dilakukan dalam penelitian.

### 4.1. Implementasi Algoritma Similarity

Implementasi algoritma perhitungan nilai *similarity* dilakukan dengan membandingkan dua kalimat berbeda. Implementasi dilakukan menggunakan aplikasi pgAdmin 4 yang menjalankan PostgreSQL 12. Hasil implementasi *query* perhitungan nilai *similarity* sesuai dengan Persamaan 1 dan Persamaan 2 secara rinci dijelaskan pada Gambar 4.

Query Editor | Query History

```
WITH kalimat AS
( SELECT
      'Ini kalimat satu'::text AS string1,
      'Ini kalimat dua'::text AS string2
),
unnest1 AS (SELECT unnest(show_trgm(string1)) FROM kalimat),
unnest2 AS (SELECT unnest(show_trgm(string2)) FROM kalimat),
x AS
( SELECT string1 as string1, string2 as string2,
    (SELECT count(*) FROM (SELECT * FROM unnest1 INTERSECT SELECT * FROM unnest2) AS s0)::integer AS intersect_result,
    (SELECT count(*) FROM (SELECT * FROM unnest1 UNION SELECT * FROM unnest2) AS s0)::integer AS union_result,
    (SELECT count(*) FROM (SELECT * FROM unnest2) AS s0)::integer AS unnest2
FROM kalimat
)
SELECT string1, string2, intersect_result, union_result, intersect_result::real/union_result::real AS similarity,
    intersect_result::real/unnest2::real AS word_similarity
FROM x;
```

Data Output | Explain | Messages | Notifications

| | string1 text | string2 text | intersect_result integer | union_result integer | similarity real | word_similarity real |
|---|---|---|---|---|---|---|
| 1 | Ini kalimat satu | Ini kalimat dua | 12 | 21 | 0.5714286 | 0.75 |

Gambar 4. Implementasi Algoritma

Hasil implementasi algoritma pengukuran nilai *similarity* menggunakan dua perhitungan yaitu perhitungan dengan *similarity* hasil pembagian nilai *intersect* dengan *union* kedua *string* dan perhitungan dengan *word similarity* hasil pembagian *intersect* kedua *string* dengan jumlah *unnest string* kedua. Perbandingan kalimat “Ini kalimat satu” dengan “Ini kalimat dua” menghasilkan nilai *intersect* sebanyak 12 dan *union* sebanyak 21 dengan nilai *similarity* sebesar 0,5714286 dan *word similarity* sebesar 0,75.

### 4.2. Pengujian Modul

Hasil pengujian perancangan algoritma pengukuran nilai *similarity* teks modul pg-trgm pada PostgreSQL menggunakan sepuluh skenario pengujian yaitu dengan menggunakan dua kata yang sama, dua kalimat yang sama, dua kalimat berbeda, kalimat tanpa spasi, suku kata dari sebuah kata, kata dari sebuah kalimat, ada kesalahan ketik pada kalimat, dan kata dalam kalimat diacak. Hasil pengukuran nilai *similarity* dari dua perhitungan dijelaskan secara rinci pada Tabel 2.

Tabel 2. Hasil Pengujian Similarity

| String 1 | String 2 | Similarity | Word Similarity |
|---|---|---|---|
| pengujian | pengujian | 1 | 1 |
| uji | pengujian | 0.07692308 | 0.25 |
| pengujian | ujian | 0.33333334 | 0.4 |
| contoh kalimat pengujian | contoh kalimat pengujian | 1 | 1 |
| pengujian | contoh kalimat pengujian | 0.4 | 1 |
| contoh kalimat pengujian | pengujian | 0.4 | 0.4 |
| pengujian contoh kalimat | contoh kalimat pengujian | 1 | 1 |
| cntoh klimt pngujian | contoh kalimat pengujian | 0.3939394 | 0.4 |
| contohkalimatpengujian | contoh kalimat pengujian | 0.6551724 | 0.6551724 |
| bukan sebuah percobaan | contoh kalimat pengujian | 0.06818182 | 0.10344828 |

Hasil pengujian dari sepuluh skenario pengujian yang telah dilakukan diperoleh sebanyak lima hasil pengukuran dengan nilai kemiripan yang sama. Perhitungan dengan *word similarity* hasil pembagian *intersect* kedua *string* dengan jumlah *unnest string* kedua menghasilkan lima hasil pengukuran *similarity* yang tinggi yaitu pada pencocokan suku kata pada sebuah kata, pencocokan sebuah kata dalam sebuah kalimat, pencocokan yang mengandung kata salah ketik, dan pencocokan kalimat yang cukup berbeda.

## 5. Kesimpulan

Modul pg-trgm pada *database* PostgreSQL menyediakan *function* untuk menghitung *similarity* pada teks alfanumerik ASCII dengan menggunakan algoritma n-gram atau secara spesifik tri-gram. Implementasi algoritma pengukuran nilai *similarity* menggunakan dua perhitungan yaitu perhitungan dengan *function similarity* dan *word similarity*. Perbedaan utama dari *function similarity* dan *word similarity* ada pada pembagi *intersect* dalam perhitungan *similarity* dimana *function similarity* menggunakan *union* dari tri-gram *string* pertama dan kedua sedangkan *word similarity* hanya menggunakan tri-gram *string* kedua. Hasil pengujian pengukuran nilai kemiripan dari dua kalimat menggunakan sepuluh skenario pengujian diperoleh sebanyak lima hasil pengukuran dengan nilai kemiripan yang sama. Perhitungan dengan *word similarity* menghasilkan lima hasil pengukuran yang lebih baik pada pencocokan suku kata pada sebuah kata, pencocokan sebuah kata dalam sebuah kalimat, pencocokan yang mengandung kata salah ketik, dan pencocokan kalimat berbeda. Hasil pengujian menunjukkan bahwa *function* untuk menentukan kemiripan teks pada modul pg-trgm lebih cocok diimplementasikan untuk teks yang pendek karena semakin banyak jumlah irisan (*intersect*) dan sedikit suku kata pada kalimat yang dicocokkan maka nilai *similarity* akan semakin baik.

## References

[1] Paliwahet, I. N. S., Sukarsa, I. M., & Gede Darma Putra, I. K. (2017). Pencarian Informasi Wisata Daerah Bali Menggunakan Teknologi Chatbot. Lontar Komputer : Jurnal Ilmiah Teknologi Informasi; Vol. 8, No. 3 Desember 2017DO - 10.24843/LKJITI.2017.V08.I03.P01 . https://ojs.unud.ac.id/index.php/lontar/article/view/33189

[2] Rana, A., & Deeba, K. (2019). Online Book Recommendation System using Collaborative

Filtering (With Jaccard Similarity). *Journal of Physics: Conference Series*, *1362*(1), 12130. https://doi.org/10.1088/1742-6596/1362/1/012130

[3] Kambey, G. E. I., Sengkey, R., & Jacobus, A. (2020). Penerapan Clustering pada Aplikasi Pendeteksi Kemiripan Dokumen Teks Bahasa Indonesia. *Jurnal Teknik Informatika*, *15*(2), 75–82.

[4] Gentia, D., Sukarsa, I. M., & Wibawa, K. S. (2020). Rancang Bangun Chatbot Sebagai Penghubung Komunikasi Antara Aplikasi Line Messenger Dengan Telegram Messenger. *Jurnal Ilmiah Merpati (Menara Penelitian Akademika Teknologi Informasi); Vol. 8, No. 3, December 2020DO - 10.24843/JIM.2020.V08.I03.P01* . https://ojs.unud.ac.id/index.php/merpati/article/view/62212

[5] Ermawati, M., & Buliali, J. L. (2018). Text Based Approach For Similar Traffic Incident Detection from Twitter. *Lontar Komputer : Jurnal Ilmiah Teknologi Informasi; Vol. 9, No. 2 August 2018DO - 10.24843/LKJITI.2018.V09.I02.P01*. https://ojs.unud.ac.id/index.php/lontar/article/view/38749

[6] Sukarsa, I., Putra, I., Sastra, N. P., & Jasa, L. (2018). A New Framework for Information System Development on Instant Messaging for Low Cost Solution. *Telkomnika (Telecommunication Computing Electronics and Control)*, *16*, 2799–2808. https://doi.org/10.12928/TELKOMNIKA.v16i6.8614

[7] Hasanah, U., & Mutiara, D. A. (2019). Perbandingan metode cosine similarity dan jaccard similarity untuk penilaian otomatis jawaban pendek. *SENSITIf: Seminar Nasional Sistem Informasi Dan Teknologi Informasi*, 1255–1263.

[8] Wahyudi, M. D. R. (2019). Penerapan Algoritma Cosine Similarity pada Text Mining Terjemah Al-Qur'an Berdasarkan Keterkaitan Topik. *Semesta Teknika*, *22*(1), 41–50.

[9] Viloria, A., Acuña, G. C., Alcázar Franco, D. J., Hernández-Palma, H., Fuentes, J. P., & Rambal, E. P. (2019). Integration of Data Mining Techniques to PostgreSQL Database Manager System. *Procedia Computer Science*, *155*, 575–580. https://doi.org/https://doi.org/10.1016/j.procs.2019.08.080

[10] Tiffani, I. E. (2020). Optimization of naïve bayes classifier by implemented unigram, bigram, trigram for sentiment analysis of hotel review. *Journal of Soft Computing Exploration*, *1*(1), 1–7.

[11] Postgresql.org. (2021). *PostgreSQL: The World's Most Advanced Open Source Relational Database*. The PostgreSQL Global Development Group. https://www.postgresql.org/docs/12/pgtrgm.html