# Penerapan metode content based filtering dalam rekomendasi software developer

**Muhammad Fiarus Ramadhani** *

Program Studi Informatika, Fakultas Ilmu Komputer, Universitas Pembangunan Nasional "Veteran" Jawa Timur
*Email: 21081010090@student.upnjatim.ac.id

## Abstrak

Penugasan developer yang tepat pada proyek perangkat lunak merupakan faktor krusial dalam meningkatkan produktivitas dan kualitas pengembangan perangkat lunak. Penelitian ini mengembangkan sistem rekomendasi developer berbasis content-based filtering dengan memanfaatkan data log aktivitas harian developer. Sebanyak 4.505 data log dan 45 data uji dengan deskripsi tugas yang lebih detail digunakan dalam penelitian ini. Metode yang diterapkan meliputi tahap preprocessing teks (lowercase, penghapusan karakter khusus, stopwords removal, dan stemming), transformasi teks menjadi representasi numerik menggunakan TF-IDF, serta perhitungan similarity menggunakan Euclidean Distance dan Cosine Similarity. Hasil evaluasi menunjukkan bahwa kedua metode similarity memiliki performa yang konsisten dengan Hit Rate mencapai 84,4%, Recall@5 sebesar 74,4%, dan Mean Reciprocal Rank (MRR) sebesar 57,6%. Penelitian ini membuktikan bahwa pendekatan content-based filtering dengan representasi TF-IDF efektif dalam mengidentifikasi developer yang sesuai berdasarkan kesamaan pola kerja historis mereka dengan kebutuhan tugas yang diberikan, sehingga dapat membantu otomatisasi proses penugasan developer pada proyek perangkat lunak.

**Kata Kunci:** sistem rekomendasi; content-based filtering; TF-IDF; rekomendasi developer; text similarity

# *Implementation of content based filtering method in software developer recommendations*

## *Abstract*

*Assigning the right developers to software projects is a crucial factor in improving productivity and software development quality. This research develops a content-based filtering developer recommendation system utilizing daily developer activity log data. A total of 4,505 log data entries and 45 test data with more detailed task descriptions were used in this study. The applied method includes text preprocessing stages (lowercase conversion, special character removal, stopwords removal, and stemming), text transformation into numerical representation using TF-IDF, and similarity calculation using Euclidean Distance and Cosine Similarity. Evaluation results show that both similarity methods have consistent performance with a Hit Rate of 84,4%, Recall@5 of 74,4%, and Mean Reciprocal Rank (MRR) of 57,6%. This research demonstrates that the content-based filtering approach with TF-IDF representation is effective in identifying suitable developers based on the similarity of their historical work patterns with given task requirements, thereby supporting the automation of developer assignment processes in software projects.*

***Keywords:*** *recommender system; content-based filtering; TF-IDF; developer recommendation; text similarity*

## 1. Pendahuluan

Rekomendasi memegang peranan penting dalam sistem informasi modern untuk membantu pengguna menemukan informasi relevan dari volume data yang besar (Son & Kim, 2017). Berbagai metode filtering seperti item-based collaborative filtering dan content-based filtering berbasis text similarity telah menunjukkan efektivitasnya dalam menghasilkan rekomendasi yang relevan (Anggraeny et al., 2019). Dalam bidang rekayasa perangkat lunak, kebutuhan serupa muncul untuk merekomendasikan developer yang tepat misalnya untuk menangani bug, melakukan code review, atau menempati tugas spesifik agar produktivitas dan kualitas perangkat lunak meningkat (Xia et al., 2015). Proses penugasan tugas di proyek perangkat lunak sering memakan waktu dan rentan menimbulkan *bug tossing* atau bottleneck pada code review, sehingga otomatisasi triaging dan rekomendasi developer memiliki nilai praktis yang tinggi bagi tim pengembang (Di Rocco et al., 2021). Penelitian ini

mengusulkan penerapan metode content-based filtering untuk merekomendasikan developer dengan memadankan representasi teks deskripsi tugas (bug/issue) dan profil kontribusi developer menggunakan teknik NLP seperti TF-IDF atau embedding teks sebagai vektor fitur (Çetin et al., 2021). Data yang relevan untuk masalah ini meliputi artefak perangkat lunak heterogen seperti laporan bug, riwayat commit, dan metadata modul yang perlu dipra-proses dan direpresentasikan secara informatif agar CBF dapat bekerja efektif dalam domain SE (Zanjani et al., 2016). Skema pencocokan berbasis kemiripan dan mekanisme pembobotan fitur domain-spesifik akan diadopsi, lalu dievaluasi dengan metrik rekomendasi standar serta indikator operasional seperti waktu penugasan dan frekuensi reassign (Yadav et al., 2019). Telaah pustaka sebelumnya menunjukkan bahwa pendekatan CBF dan teknik expert-finding dapat menjadi pijakan kuat untuk merancang rekomender developer, sementara studi-studi tentang reviewer/dev-recommendation dan bug triaging menyediakan berbagai baseline dan metrik evaluasi yang relevan (Cavalcanti et al., 2016). Laporan pengalaman implementasi sistem rekomendasi di konteks software engineering menyoroti tantangan praktis seperti heterogenitas data, masalah reproducibility, dan kebutuhan integrasi ke workflow developer yang harus dipertimbangkan saat menerapkan solusi CBF di lapangan (Saha et al., 2015). Dengan dasar tersebut, penelitian ini bertujuan merancang, mengimplementasikan, dan mengevaluasi sistem rekomendasi developer berbasis content-based filtering dengan hipotesis bahwa representasi teks yang baik dan pembobotan fitur domain-spesifik akan meningkatkan metrik relevansi rekomendasi dibandingkan baseline heuristik (Tecimer et al., 2022).

## 2. Metode

Metode penelitian ini menjelaskan secara sistematis tahapan yang dilakukan dalam pengembangan sistem rekomendasi. Alur metode dimulai dari tahap persiapan data, preprocessing data, pembobotan fitur menggunakan TF-IDF, penerapan metode content-based filtering, hingga evaluasi kinerja sistem rekomendasi yang dihasilkan.

### 2.1. Persiapan Data

Tahap persiapan data merupakan langkah krusial sebelum pembentukan model dilakukan, di mana seluruh data log aktivitas developer yang telah dikumpulkan diproses dan disesuaikan agar memenuhi kebutuhan analisis serta pemodelan pada tahap berikutnya; data tersebut berasal dari catatan log para developer yang berisi teks singkat mengenai pekerjaan harian beserta nama pengguna masing-masing, dengan total 4.505 data yang digunakan dalam penelitian ini, serta ditambahkan 45 data uji berupa teks panjang yang memuat deskripsi tugas secara lebih rinci untuk menguji kemampuan model dalam menangani variasi panjang dokumen dan memastikan model tetap mampu memberikan rekomendasi yang relevan meskipun menerima input deskripsi pekerjaan yang lebih mendalam.

### 2.2. Preprocessing Data

Pada tahap preprocessing, data teks terlebih dahulu diterjemahkan ke bahasa Inggris menggunakan Google Translate berbasis deep learning, kemudian diubah menjadi huruf kecil (*lowercase*). Selanjutnya dilakukan penghapusan tanda baca dan angka, dilanjutkan dengan tokenisasi untuk memecah teks menjadi kata-kata. Setelah itu, *stopwords* dihapus agar hanya tersisa kata penting, dan tahap terakhir adalah stemming untuk mengubah kata ke bentuk dasarnya.

### 2.3. TF-IDF

Selanjutnya adalah mengubah data menjadi vektor untuk digunakan fitur dalam perhitungan dalam metode content based filtering. Salah satu langkah penting dalam analisis sentimen adalah pembobotan kata. Dua metode pembobotan kata yang paling umum adalah TF-IDF dan Word2Vec(Ahmad et al., 2024). Term Frequency-Inverse Document Frequency (TF-IDF) merupakan sebuah metode yang digunakan untuk menilai tingkat kepentingan suatu kata dalam sebuah dokumen. Teknik ini banyak diterapkan karena dikenal memiliki efisiensi tinggi, mudah digunakan, serta mampu menghasilkan output yang akurat (Putri Gabriella, 2023). TF-IDF mengombinasikan dua prinsip utama, yaitu Term Frequency (TF), yang menentukan seberapa sering sebuah kata muncul dalam dokumen dengan memberikan bobot sebesar 1 untuk setiap kata, serta Inverse Document Frequency (IDF), yang mengukur frekuensi kemunculan kata tersebut di seluruh dokumen

yang tersedia (Herwijayanti et al., 2018).Proses perhitungan diawali dengan menentukan nilai TF, yaitu dengan membagi jumlah kemunculan suatu kata dalam dokumen dengan total jumlah kata yang ada dalam dokumen tersebut. Perhitungan ini dapat dijelaskan melalui rumus berikut:

$$TF_{t,d} = \frac{jumlah\ kata\ t\ dalam\ dokumen\ d}{total\ jumlah\ kata\ dalam\ dokumen\ d} \quad (2.1)$$

Kemudian dilakukan penghitungan IDF atau Inverse Document Frequency untuk menghitung jumlah kemunculan suatu kata dalam semua dokumen yang ada.

$$IDF_t = \log\left(\frac{N}{df_t}\right) \quad (2.2)$$

$N$ merupakan total jumlah dokumen yang tersedia dan $dft$ adalah jumlah dari dokumen yang mengandung kata $t$. Hasil dari nilai $TFt,d$ dan $IDFt$ selanjutnya akan dikalikan untuk mendapatkan hasil dari teknik TF-IDF

$$TF - IDF = TF_{t,d} \times IDF_t \quad (2.3)$$

Hasil TF-IDF ini merepresentasikan tingkat kepentingan suatu kata dalam dokumen tertentu dibandingkan dengan seluruh dokumen yang tersedia.

### 2.4. Content Based Filtering

Content-based filtering merupakan metode sistem rekomendasi yang bekerja dengan menganalisis karakteristik konten item dan mencocokkannya dengan profil preferensi pengguna berdasarkan interaksi historis mereka (Lops et al., 2011). Pendekatan ini merepresentasikan item dan profil pengguna dalam bentuk vektor fitur hasil ekstraksi teks menggunakan teknik seperti TF-IDF (Term Frequency-Inverse Document Frequency), yang mengukur kepentingan suatu kata dalam dokumen relatif terhadap koleksi dokumen [2]. Untuk mengukur tingkat kemiripan antara vektor item dan profil pengguna, digunakan metrik kesamaan seperti cosine similarity dan Euclidean distance (Aggarwal, 2016). Cosine similarity mengukur kesamaan berdasarkan sudut antara dua vektor, sehingga efektif untuk data berdimensi tinggi dan sparse karena tidak terpengaruh oleh magnitude vektor (Singhal Google, 2001). Berikut perhitungan Cosine Similarity:

$$\boldsymbol{cosine(a,b)} = \frac{\sum_{i=1}^{n} \boldsymbol{a_i}\, \boldsymbol{b_i}}{\sqrt{\sum_{i=1}^{n} \boldsymbol{a_i^2}}\sqrt{\sum_{i=1}^{n} \boldsymbol{b_i^2}}} \quad \textbf{(2.1)}$$

Sedangkan Euclidean distance mengukur jarak absolut antar vektor dan lebih sensitif terhadap perbedaan skala nilai fitur . Ber(Singhal Google, 2001)ikut perhitungan Euclidean Distance:

$$d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \quad (2.2)$$

Keterangan :
$d$(x,y) = jarak euclidean
$x_i$ = nilai titik 1
$y_i$ = nilai titik 2
Item dengan nilai cosine similarity tertinggi atau Euclidean distance terendah akan mendapatkan peringkat teratas dan direkomendasikan kepada pengguna, sehingga menghasilkan rekomendasi yang personal dan sesuai dengan pola preferensi pengguna (Pazzani & Billsus, n.d.)

### 2.5. Evaluasi

Evaluasi sistem rekomendasi pada metode yang digunakan dalam penelitian ini dilakukan untuk menilai kinerja sistem dalam memberikan rekomendasi tugas yang relevan kepada developer dengan menggunakan tiga metrik utama, yaitu Mean Reciprocal Rank (MRR), Hit Ratio, dan Recall (Shani & Gunawardana, 2011). MRR digunakan untuk mengukur seberapa cepat metode yang diusulkan menampilkan tugas relevan pertama pada peringkat awal daftar rekomendasi sehingga mencerminkan efektivitas sistem dalam memprioritaskan rekomendasi yang paling penting bagi developer (Craswell, 2009). Hit Ratio digunakan untuk mengetahui seberapa sering metode yang diterapkan berhasil menampilkan setidaknya satu tugas relevan dalam sejumlah rekomendasi teratas (top-k), sehingga metrik ini sesuai untuk mengevaluasi tingkat keberhasilan dasar sistem rekomendasi yang dibangun (Gunawardana & Shani, 2015). Recall digunakan untuk mengukur sejauh mana metode yang diusulkan mampu mencakup seluruh tugas relevan yang dimiliki oleh seorang developer, sehingga metrik ini menilai kemampuan sistem dalam menghasilkan rekomendasi yang komprehensif dan tidak terbatas pada satu tugas yang benar saja (Manning et al., 2008).

## 3. Hasil dan Pembahasan

Pada bagian ini, dijelaskan hasil penelitian/pengabdian kepada masyarakat dan pada saat bersamaan diberikan pembahasan yang komprehensif. Hasil dapat disajikan dalam gambar, grafik, tabel dan lain-lain yang membuat pembaca mudah mengerti. Diskusi bisa dilakukan di beberapa sub-bab.

### 3.1. Persiapan Data

Tahap persiapan data adalah langkah krusial sebelum proses pembentukan model, di mana data log aktivitas developer yang telah dikumpulkan diproses dan disesuaikan agar siap digunakan untuk kebutuhan analisis dan pemodelan pada tahap berikutnya.

| | | | | |
|---|---|---|---|---|
| **4498** | Ahfaitar Abdil Hakim | - fix: chat \| fix create merchant and pipeline... | 2025-04-24 17:03:00 | 2025-04-24 17:03:00 |
| **4501** | Ahfaitar Abdil Hakim | - analysis: analysis of app performance | 2025-04-24 17:03:00 | 2025-04-24 17:03:00 |
| **4503** | samm002 | ✅ Add the meeting template IDs property to the... | 2025-04-25 11:04:00 | 2025-04-25 11:04:00 |
| **4502** | samm002 | branch → https://github.com/tunaskreasidigital... | 2025-04-25 11:04:00 | 2025-04-25 11:04:00 |
| **4504** | samm002 | ✅ Added merchant template ID properties to the... | 2025-04-25 11:04:00 | 2025-04-25 11:04:00 |

4505 rows × 4 columns

***Gambar 3. 1** dataset*

Pada Gambar 3.1 dapat dilihat data yang digunakan berasal dari kumpulan log para developer yang berisi teks pendek mengenai pekerjaan harian beserta nama pengguna (username) masing-masing, dengan total 4.505 data log yang digunakan dalam penelitian ini, serta ditambahkan 45 data uji berupa teks panjang yang memuat deskripsi tugas secara lebih detail untuk menguji kemampuan model dalam menangani variasi panjang dokumen dan memastikan bahwa model tetap mampu memberikan rekomendasi yang relevan meskipun menerima input deskripsi pekerjaan yang lebih mendalam.

### 3.2. Preprocessing Data

Setelah data dikumpulkan, langkah selanjutnya adalah melakukan preprocessing data untuk membersihkan dan mengubah teks mentah menjadi format yang dapat diproses oleh model TF-IDF. Preprocessing merupakan tahap krusial dalam penelitian ini karena kualitas data yang dimasukkan akan berpengaruh langsung terhadap akurasi hasil analisis. Proses preprocessing yang dilakukan meliputi beberapa tahap transformasi teks, yaitu mengubah teks menjadi huruf kecil, menghapus karakter khusus dan tanda baca, menghilangkan stopwords , serta melakukan stemming untuk mengubah kata-kata ke bentuk dasarnya. Tahapan ini bertujuan untuk mengurangi dimensi data dan

memfokuskan analisis pada kata-kata yang memiliki nilai informasi tinggi.

| | username | log | preprocessed | started | stopped |
|---|---|---|---|---|---|
| **4500** | Ahfaitar Abdil Hakim | - refactor: analytic \| refactor code in analytic | refactor analyt refactor code analyt | 2025-04-24 17:03:00 | 2025-04-24 17:03:00 |
| **4501** | Ahfaitar Abdil Hakim | - analysis: analysis of app performance | analysi analysi app perform | 2025-04-24 17:03:00 | 2025-04-24 17:03:00 |
| **4502** | samm002 | branch → https://github.com/tunaskreasidigital/wooblazz-crm/tree/sm-add-merchant-meeting-template-in-studio-for-lead-template-and-merchant-template | branch http github com tunaskreasidigit wooblazz crm tree sm add merchant meet templat studio lead templat merchant templat | 2025-04-25 11:04:00 | 2025-04-25 11:04:00 |
| **4503** | samm002 | ✅ Add the meeting template IDs property to the create merchant template payload | add meet templat id properti creat merchant templat payload | 2025-04-25 11:04:00 | 2025-04-25 11:04:00 |
| **4504** | samm002 | ✅ Added merchant template ID properties to the create lead template payload | ad merchant templat id properti creat lead templat payload | 2025-04-25 11:04:00 | 2025-04-25 11:04:00 |

***Gambar 3. 2** Hasil Preprocessing*

Pada Gambar 3.2 dapat dilihat hasil dari proses preprocessing terhadap data log developer. Kolom "log" menampilkan teks asli dari aktivitas harian, sedangkan kolom "preprocessed" menunjukkan hasil setelah pembersihan data. Terlihat bahwa teks telah disederhanakan dengan menghapus karakter khusus, mengubah kata ke bentuk dasar melalui stemming, dan menghilangkan kata-kata yang tidak penting, sehingga hanya menyisakan kata kunci yang relevan untuk proses TF-IDF selanjutnya.

### 3.3. TF-IDF

Setelah data di preprocessing, selanjutnya adalah mengubah teks menjadi representasi numerik berupa vektor TF-IDF.

| | ab | abandon | abl | absenc | absent | ac | accept | access | accommod | accompani | ... | yearli | yesterday | yet | yg | yml | youtub | zero | zone | zoom | zustand |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **2** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **3** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **4500** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4501** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4502** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4503** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **4504** | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

4505 rows × 1696 columns

*Gambar 3. 3 Hasil TF-IDF*

Pada Gambar 3.3 dapat dilihat hasil transformasi TF-IDF yang menghasilkan matriks berukuran 4505 baris × 1696 kolom, di mana setiap baris merepresentasikan satu log aktivitas dan setiap kolom merepresentasikan satu kata unik dari seluruh korpus. Nilai dalam matriks menunjukkan bobot TF-IDF untuk setiap kata dalam dokumen, dengan nilai 0.0 mengindikasikan bahwa kata tersebut tidak muncul dalam dokumen terkait. Matriks sparse ini menjadi representasi numerik dari data teks yang siap digunakan untuk perhitungan similarity dalam sistem rekomendasi.

### 3.4. Rekomendasi

Setelah mendapatkan representasi vektor TF-IDF dari setiap log aktivitas, sistem kemudian melakukan perhitungan similarity untuk menghasilkan rekomendasi developer. Proses ini dilakukan dengan menghitung kesamaan antara vektor TF-IDF dari deskripsi tugas baru dengan vektor TF-IDF dari log aktivitas historis setiap developer. Developer dengan nilai similarity tertinggi akan direkomendasikan sebagai kandidat yang paling sesuai untuk mengerjakan tugas tersebut berdasarkan pengalaman kerja mereka sebelumnya.

| | tugas | developer | rekomendasi | rekomendasi benar satu | rekomendasi benar semua |
|---|---|---|---|---|---|
| 0 | Feature Description: Add Chat Phone Number Input in Add User Form and Excel Import\n\nThis feature add... | Artha Gusfi, abdurrahmanff | abdurrahmanff, Artha Gusfi, richardasmarakandi17, Muhammad Teguh, fairus | ✓ | ✓ |
| 1 | Feature Description: Custom Dashboard\n\nThis feature introduces a customizable dashboard that display... | Syukur Iman Attaqwa | Muhammad Teguh, Izzah Dinillah, richardasmarakandi17, Syukur Iman Attaqwa, Reynaldi Kindarto | ✓ | ✓ |
| 2 | Bug Fix: Analytics Click Not Showing Correct Data When Custom Rule Filter Is Enabled\na. Location: An... | Syukur Iman Attaqwa,fairus | Artha Gusfi, Syukur Iman Attaqwa, Muhammad Teguh, abdurrahmanff, Reynaldi Kindarto | ✓ | ✗ |
| 3 | \nFeature Name: Set Progress Based on Role\na. Menu Location: Users & Roles > Roles > Three-dot Action... | richardasmarakandi17,Artha Gusfi | richardasmarakandi17, Reynaldi Kindarto, fairus, Muhammad Teguh, Enjirou | ✓ | ✗ |
| 4 | Feature Name: New Variables in Invoice\na. Menu Location: Studio > Invoice\nb. Description:\ni. Multi-p... | abdurrahmanff | Izzah Dinillah, Muhammad Teguh, Rifky Akhmad F, richardasmarakandi17, Enjirou | ✗ | ✗ |
| 5 | Feature Name: Custom Field in Custom Rule Analytics\na. Location: Analytics > Components > Custom Rul... | Syukur Iman Attaqwa,fairus | Rifky Akhmad F, Artha Gusfi, Syukur Iman Attaqwa, Muhammad Teguh, abdurrahmanff | ✓ | ✗ |
| 6 | Feature Name: "Paid Term Amount" Widget\na. Location: Dashboard\nb. Description:\n"Paid Term Amount" Wi... | Syukur Iman Attaqwa,fairus | Muhammad Teguh, richardasmarakandi17, Timothy JS, Syukur Iman Attaqwa, abdurrahmanff | ✓ | ✗ |
| 7 | 45. Feature Name: Auto-create Merchants from Chat (Maxy)a. Location: Chatb. Description:i. Automatic... | richardasmarakandi17 | richardasmarakandi17, Artha Gusfi, Ahfaitar Abdil Hakim, Muhammad Teguh, abdurrahmanff | ✓ | ✓ |

Strategy: Content-Based Filtering (Cosine Similarity)

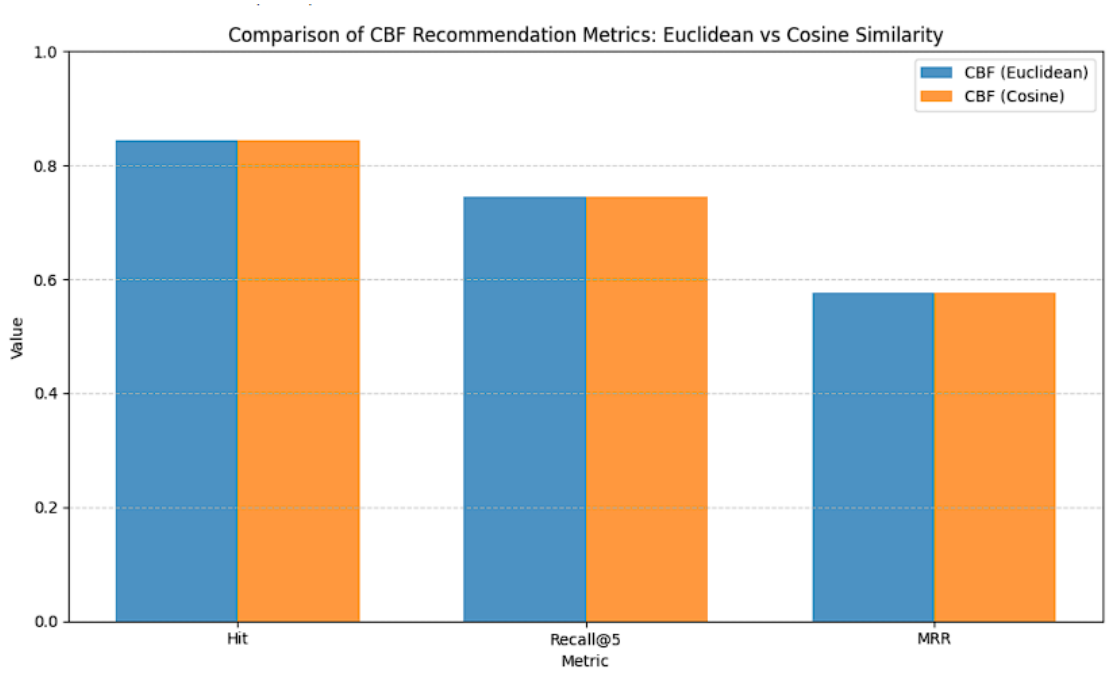
***Gambar 3. 4** Hasil Rekomendasi*

Pada Gambar 3.4 dapat dilihat hasil rekomendasi developer untuk setiap tugas yang diberikan. Kolom "tugas" berisi deskripsi pekerjaan yang perlu dikerjakan, kolom "developer" menampilkan daftar developer yang direkomendasikan beserta nama-nama terkait, dan kolom validasi menunjukkan kebenaran rekomendasi dengan tanda centang hijau untuk benar dan silang merah untuk tidak sesuai

.

## 3.5. Evaluasi

Untuk mengukur kinerja sistem rekomendasi yang telah dibangun, dilakukan evaluasi menggunakan beberapa metrik standar dalam sistem rekomendasi. Evaluasi ini bertujuan untuk membandingkan performa dua metode pengukuran similarity yang digunakan, yaitu Euclidean Distance dan Cosine Similarity. Metrik yang digunakan meliputi Hit Rate untuk mengukur persentase rekomendasi yang berhasil, Recall@5 untuk mengevaluasi kemampuan sistem dalam menemukan developer yang relevan di top-5 rekomendasi, dan MRR (Mean Reciprocal Rank) untuk mengukur posisi rata-rata developer yang relevan dalam daftar rekomendasi.




***Gambar 3. 5** Metrik Evaluasi*

Pada Gambar 3.5 dapat dilihat perbandingan performa sistem rekomendasi menggunakan dua metode pengukuran similarity. Hasil menunjukkan bahwa kedua metode memiliki performa yang sangat mirip, dengan CBF (Euclidean) dan CBF (Cosine) mencapai Hit Rate sekitar 0.844, Recall@5 sekitar 0.744, dan MRR sekitar 0.576, mengindikasikan bahwa kedua metode efektif dalam memberikan rekomendasi yang relevan.

## 4. Kesimpulan

Penelitian ini telah berhasil mengembangkan sistem rekomendasi developer berbasis Content-Based Filtering menggunakan metode TF-IDF untuk menganalisis log aktivitas harian developer. Dari 4.505 data log dan 45 data uji yang digunakan, sistem mampu menghasilkan rekomendasi developer yang relevan dengan tingkat akurasi yang baik. Hasil evaluasi menunjukkan bahwa kedua metode similarity yang digunakan, yaitu Euclidean Distance dan Cosine Similarity, memiliki performa yang sangat mirip dengan Hit Rate mencapai 84,4%, Recall@5 mencapai 74,4%, dan MRR sebesar 57,6%. Hal ini menunjukkan bahwa sistem rekomendasi yang dibangun efektif dalam mengidentifikasi developer yang sesuai berdasarkan kesamaan pola kerja historis mereka dengan kebutuhan tugas yang diberikan. Proses preprocessing yang meliputi lowercase, penghapusan karakter khusus, stopwords removal, dan stemming terbukti berperan penting dalam meningkatkan kualitas representasi teks dan akurasi rekomendasi.

## 5. Ucapan terimakasih

Penulis mengucapkan terima kasih kepada semua pihak yang telah memberikan dukungan dan kontribusi dalam pelaksanaan penelitian ini. Terima kasih kepada institusi yang telah menyediakan fasilitas penelitian, tim developer yang telah memberikan izin penggunaan data log aktivitas, serta pembimbing yang telah memberikan arahan dan masukan selama proses penelitian berlangsung.

## Daftar Pustaka

Aggarwal, C. C. (2016). *Recommender Systems*. Springer International Publishing. https://doi.org/10.1007/978-3-319-29659-3

Ahmad, *, Dani, H., Dani, A. H., Puspaningrum, E. Y., Mumpuni, R., Pembangunan, U., Veteran, N. ", & Timur, J. (2024). *Studi Performa TF-IDF dan Word2Vec Pada Analisis Sentimen Cyberbullying*. (2), 94–106. https://doi.org/10.62951/router.v2i2.76

Anggraeny, F. T., Purbasari, I. Y., & Wulandari, E. F. (2019). *5 th ICITB Undergraduate Thesis Supervisor Recommendation Based On Text Similarity*.

Cavalcanti, Y. C., Machado, I. do C., Neto, P. A. da M. S., & Almeida, E. S. de. (2016). Towards semi-automated assignment of software change requests. *Journal of Systems and Software*, *115*, 82–101. https://doi.org/10.1016/j.jss.2016.01.038

Çetin, H. A., Doğan, E., & Tüzün, E. (2021). A review of code reviewer recommendation studies: Challenges and future directions. *Science of Computer Programming*, *208*, 102652. https://doi.org/10.1016/j.scico.2021.102652

Craswell, N. (2009). Mean Reciprocal Rank. In *Encyclopedia of Database Systems* (pp. 1703–1703). Springer US. https://doi.org/10.1007/978-0-387-39940-9_488

Di Rocco, J., Di Ruscio, D., Di Sipio, C., Nguyen, P. T., & Rubei, R. (2021). Development of recommendation systems for software engineering: the CROSSMINER experience. *Empirical Software Engineering*, *26*(4), 69. https://doi.org/10.1007/s10664-021-09963-7

Gunawardana, A., & Shani, G. (2015). Evaluating Recommender Systems. In *Recommender Systems Handbook* (pp. 265–308). Springer US. https://doi.org/10.1007/978-1-4899-7637-6_8

Herwijayanti, B., Ratnawati, D. E., & Muflikhah, L. (2018). Klasifikasi Berita Online dengan menggunakan Pembobotan TF-IDF dan Cosine Similarity. *Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer*, *2*(1), 306–312.

Lops, P., de Gemmis, M., & Semeraro, G. (2011). Content-based Recommender Systems: State of the Art and Trends. In *Recommender Systems Handbook* (pp. 73–105). Springer US. https://doi.org/10.1007/978-0-387-85820-3_3

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. https://doi.org/10.1017/CBO9780511809071

Pazzani, M. J., & Billsus, D. (n.d.). Content-Based Recommendation Systems. In *The Adaptive Web* (pp. 325–341). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-72079-9_10

Putri Gabriella, Y. A. (2023). Optimasi Penerimaan Siswa Baru Dengan Penerapan Algortima Text Mining Dan Tf-Idf. *Journal of Computing and Informatics Research*, *2*(3), 110–117. https://doi.org/10.47065/comforch.v2i3.941

Saha, R. K., Khurshid, S., & Perry, D. E. (2015). Understanding the triaging and fixing processes of long lived bugs. *Information and Software Technology*, *65*, 114–128. https://doi.org/10.1016/j.infsof.2015.03.002

Shani, G., & Gunawardana, A. (2011). Evaluating Recommendation Systems. In *Recommender Systems Handbook* (pp. 257–297). Springer US. https://doi.org/10.1007/978-0-387-85820-3_8

Singhal Google, A. (2001). *Modern Information Retrieval: A Brief Overview*. http://trec.nist.gov

Son, J., & Kim, S. B. (2017). Content-based filtering for recommendation systems using multiattribute networks. *Expert Systems with Applications*, *89*, 404–412. https://doi.org/10.1016/j.eswa.2017.08.008

Tecimer, K. A., Tüzün, E., Moran, C., & Erdogmus, H. (2022). Cleaning ground truth data in software task assignment. *Information and Software Technology*, *149*, 106956. https://doi.org/10.1016/j.infsof.2022.106956

Xia, X., Lo, D., Wang, X., & Zhou, B. (2015). Dual analysis for recommending developers to resolve bugs. *Journal of Software: Evolution and Process*, *27*(3), 195–220. https://doi.org/10.1002/smr.1706

Yadav, A., Singh, S. K., & Suri, J. S. (2019). Ranking of software developers based on expertise score for bug triaging. *Information and Software Technology*, *112*, 1–17. https://doi.org/10.1016/j.infsof.2019.03.014

Zanjani, M. B., Kagdi, H., & Bird, C. (2016). Automatically Recommending Peer Reviewers in Modern Code Review. *IEEE Transactions on Software Engineering*, *42*(6), 530–543. https://doi.org/10.1109/TSE.2015.2500238