BAB II
KAJIAN LITERATUR

2.1 Resep Makanan

Resep makanan merupakan elemen kuliner fundamental yang berfungsi sebagai panduan prosedural terstruktur dalam mempersiapkan bahan-bahan, melakukan proses pengolahan, hingga menyajikan suatu hidangan [13]. Sebagai salah satu bentuk teks prosedur, sebuah resep bertugas memberikan arahan langkah demi langkah yang sistematis agar pembaca dapat memahami dan mengikuti instruksi dengan mudah untuk mencapai hasil akhir yang diharapkan [14]. Keberadaan resep sangat krusial dalam menuntun seseorang untuk menciptakan hidangan yang tidak hanya lezat dan menarik secara visual, tetapi juga telah teruji cita rasanya serta terjamin kualitas kesehatannya [13]. Oleh karena itu, resep harus disusun menggunakan bahasa yang jelas, ringkas, dan mematuhi kaidah kebahasaan agar fungsinya sebagai media komunikasi antara penulis dan pembaca dapat tercapai secara efektif [14].

Secara struktural, sebuah resep makanan yang lengkap terdiri dari beberapa komponen utama, yaitu nama makanan, daftar bahan beserta takarannya, bumbu, metode memasak, serta instruksi penyajian [13], [4]. Bahan-bahan di dalam resep biasanya diklasifikasikan ke dalam kategori tertentu, seperti bahan pokok, bumbu-bumbuan, lauk-pauk, sayuran, dan bahan tambahan lainnya [4]. Selain sekadar daftar komposisi, resep juga memberikan petunjuk mendalam mengenai proporsi bahan dan cara memperlakukannya, termasuk teknik fisik seperti memotong, mencampur, atau menghaluskan [6]. Penggunaan teknik pokok ini sangat penting untuk membantu pengguna, terutama pemula, dalam memahami prosedur kerja dan urutan pengolahan makanan secara logis dari tahap persiapan hingga siap dihidangkan [4], [6].

Dalam aspek gaya hidup dan kesehatan, resep makanan memegang peranan penting dalam mendorong budaya memasak di rumah yang terbukti lebih sehat dibandingkan mengonsumsi makanan cepat saji [3]. Hal ini disebabkan oleh kemampuan individu untuk mengontrol penggunaan energi, lemak, dan gula yang cenderung lebih rendah pada makanan rumah [3]. Penulisan resep yang akurat dan mudah diikuti menjadi faktor penentu keberhasilan dalam menghasilkan makanan yang bergizi seimbang sesuai dengan kebutuhan nutrisi maupun batasan diet tertentu dari pengguna [14]. Melalui panduan resep yang

terstandarisasi, individu dapat mengelola pola konsumsi mereka dengan lebih baik sekaligus mengurangi risiko penyakit yang disebabkan oleh pola makan tidak sehat [3].

Seiring dengan kemajuan teknologi informasi, dokumentasi resep telah mengalami transformasi besar dari media cetak tradisional menuju platform digital seperti aplikasi seluler dan situs *web* [13], [2]. Inovasi ini menjawab kebutuhan masyarakat modern akan mobilitas dan efisiensi, di mana ribuan koleksi resep digital kini dapat disimpan, dikelola, dan dicari secara instan melalui perangkat pintar [13]. Saat ini, melimpahnya informasi resep di internet memungkinkan pengguna untuk mengeksplorasi variasi menu yang sangat beragam dari berbagai budaya di seluruh dunia [2], [3]. Pengelolaan basis data resep yang optimal dalam sistem digital menjadi kunci utama untuk membantu masyarakat mengatasi kesulitan dalam menentukan ide makanan harian yang variatif dan kreatif [13], [4].

2.2 Aplikasi

Aplikasi merupakan perangkat lunak yang dikembangkan untuk membantu pengguna dalam menyelesaikan tugas tertentu melalui pemanfaatan teknologi informasi secara efektif dan efisien. Aplikasi berfungsi sebagai media pengolahan dan penyajian informasi, sarana interaksi antara pengguna dan sistem, serta alat pendukung dalam pengelolaan data dan peningkatan produktivitas kerja. Keberadaan aplikasi memungkinkan proses kerja dilakukan secara lebih terstruktur, cepat, dan akurat, baik pada lingkungan berbasis *web* maupun perangkat bergerak, sehingga mampu meningkatkan kualitas layanan serta kemudahan akses informasi bagi pengguna [15].

Seiring perkembangannya, aplikasi pada zaman sekarang dapat dibagi menjadi tiga jenis utama, yaitu:

1. Aplikasi *Desktop*

   Aplikasi *desktop* merupakan perangkat lunak yang dijalankan secara lokal pada komputer dan diinstal pada sistem operasi tertentu untuk mendukung pengolahan dan manajemen data secara optimal. Penerapan aplikasi desktop banyak digunakan pada sistem informasi internal karena menawarkan kinerja yang stabil dan kontrol penuh terhadap sistem [16]. Fungsi utama aplikasi desktop meliputi pengelolaan data, pemrosesan informasi, serta penyimpanan data secara terstruktur dengan tingkat keamanan yang relatif tinggi. Selain itu, aplikasi desktop sering dimanfaatkan pada lingkungan organisasi dan institusi untuk mendukung aktivitas operasional yang membutuhkan akses data lokal dan performa sistem yang konsisten, seperti sistem administrasi dan pengelolaan informasi internal [17].

2. Aplikasi *Web*

Aplikasi *web* merupakan perangkat lunak yang diakses melalui peramban dengan memanfaatkan jaringan internet dan dirancang untuk mendukung aktivitas kerja secara fleksibel dan terintegrasi. Aplikasi *web* memungkinkan pengguna mengakses sistem dari berbagai perangkat tanpa instalasi lokal serta mempermudah pengelolaan dan pembaruan sistem secara terpusat [18]. Selain itu, aplikasi web memiliki karakteristik utama berupa ketergantungan pada arsitektur *client–server*, kemampuan akses lintas *platform*, serta kemudahan dalam pengembangan dan pemeliharaan sistem karena perubahan dapat dilakukan pada sisi server tanpa memengaruhi perangkat pengguna. Karakteristik tersebut menjadikan aplikasi *web* lebih adaptif terhadap perkembangan kebutuhan pengguna dan perubahan lingkungan teknologi [19].

3. Aplikasi *Mobile*

Aplikasi *mobile* merupakan perangkat lunak yang dirancang untuk dijalankan pada perangkat bergerak seperti *smartphone* dan tablet, serta dapat diunduh dan diinstal oleh pengguna melalui jaringan internet. Aplikasi *mobile* dikembangkan untuk mendukung mobilitas pengguna dan memungkinkan akses sistem secara fleksibel sesuai dengan kebutuhan pengguna [20]. Selain itu, aplikasi *mobile* memiliki karakteristik utama berupa kemampuan memanfaatkan fitur bawaan perangkat, seperti kamera, sensor, lokasi (*GPS*), dan notifikasi, serta antarmuka yang dirancang khusus untuk interaksi sentuh dan ukuran layar yang terbatas. Karakteristik tersebut menjadikan aplikasi *mobile* lebih kontekstual dan responsif terhadap kondisi serta lingkungan penggunaan [21].

2.3 Waterfall

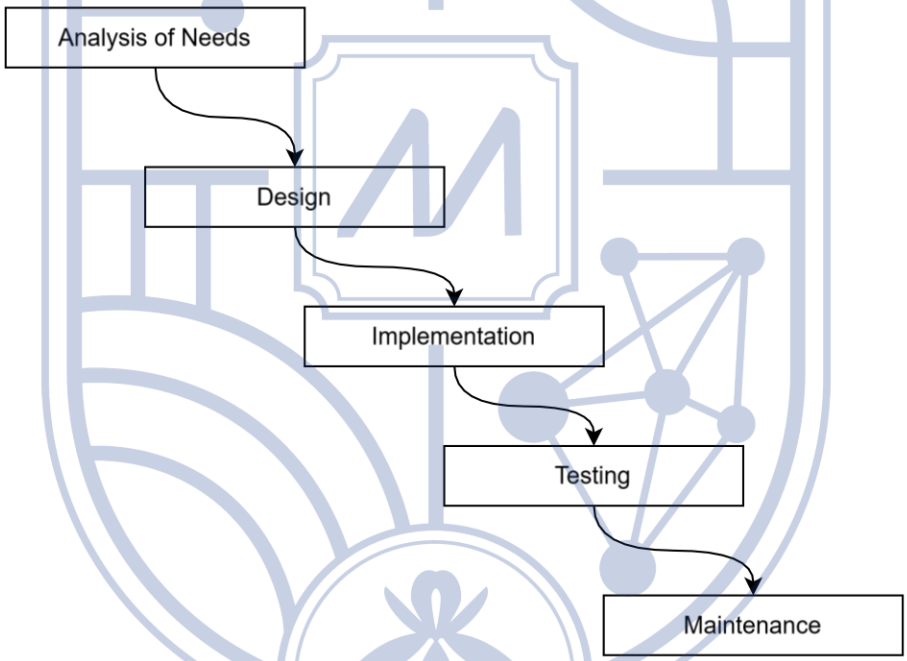
Metode *waterfall* adalah model pengembangan perangkat lunak yang sering dikenal sebagai *linear sequential model* atau bagian dari *classical life cycle* [22], [23]. Metode ini memiliki pendekatan yang berfokus pada aspek sistematis dan berurutan dalam pengembangan perangkat lunak, di mana setiap tahapan harus diselesaikan satu per satu dari atas ke bawah, layaknya aliran air terjun [24]. Dalam model ini, satu fase tidak dapat dilakukan secara bersamaan dengan fase lainnya karena prosesnya mengarah pada satu arah [24]. Fokus ini diterapkan untuk memastikan proses proyek pengembangan perangkat lunak dari awal sampai akhir pengembangan berhasil secara efektif [25].

Metode waterfall memastikan proses pengembangan sistem informasi menjadi lebih terstruktur, sistematis, dan terfokus [24], [25]. Dengan mengikuti alur yang jelas,

pengembang dapat memantau keberhasilan proyek secara lebih efektif melalui tahapan-tahapan yang telah ditentukan [25].

Metode ini dipakai karena sifatnya yang sangat sederhana dan telah digunakan secara luas dalam berbagai studi pengembangan metode [22]. *Waterfall* dianggap efektif untuk mendefinisikan kebutuhan perangkat lunak secara intens di awal proses, sehingga pengguna dapat memahami spesifikasi sistem dengan jelas sebelum masuk ke tahap teknis [22]. Selain itu, metode ini sangat membantu dalam menghasilkan sistem yang lebih tertata dan terdokumentasi dengan baik [25].

Jika dibandingkan dengan metode lain, seperti metode Agile, perbedaan utamanya terletak pada fleksibilitas tahapan SDLC (*System Development Life Cycle*) [24].
Berikut adalah tahapan-tahapan dalam metode *waterfall*:



Gambar 2.1 Gambar Waterfall

1. *Analysis of Needs* (*Requirements*)

   Tahap awal yang dilakukan secara intens untuk mendefinisikan dan menganalisis kebutuhan perangkat lunak melalui observasi, wawancara, survei, atau studi pustaka agar dapat dipahami oleh pengguna [22], [24].

2. *Design*

   Proses menerjemahkan spesifikasi kebutuhan ke dalam desain proposal atau model perangkat lunak, yang mencakup struktur data, arsitektur perangkat lunak, visualisasi antarmuka (*interface*), dan teknik pengkodean [22], [24].

3. *Implementation* (*Code Generation*)

   Tahap di mana hasil desain diterjemahkan ke dalam kode program menggunakan bahasa pemrograman tertentu (seperti Python, Javascript, PHP) untuk menjadi sebuah aplikasi atau program komputer yang utuh [22], [24].

4. *Testing* (*Integration*)

   Pengujian program dari sudut pandang logis maupun praktis untuk memastikan semua komponen berfungsi dengan baik, meminimalkan kesalahan (*error*), dan menjamin output yang dihasilkan sesuai dengan harapan [22], [24].

5. *Maintenance (Operation)*

   Tahap pemeliharaan yang meliputi perbaikan kesalahan yang ditemukan setelah aplikasi digunakan oleh pengguna serta melakukan pengembangan sistem lebih lanjut jika diperlukan [22], [24].

2.3 Unified Modeling Language

Unified Modeling Language (UML) merupakan bahasa pemodelan yang digunakan untuk memvisualisasikan, mendokumentasikan, serta merancang sistem perangkat lunak dalam bentuk diagram yang terstruktur. UML berperan sebagai alat bantu dalam proses analisis dan desain sistem karena mampu menggambarkan sistem secara jelas melalui pendekatan berorientasi objek. Dengan adanya UML, pengembang dapat memahami kebutuhan sistem secara lebih sistematis sebelum tahap implementasi dilakukan [26].

Penggunaan UML dalam pengembangan sistem informasi sangat penting karena dapat meningkatkan efektivitas dan efisiensi dalam proses perancangan. UML memungkinkan proses pemodelan sistem dilakukan secara visual sehingga memudahkan dalam menggambarkan alur kerja, kebutuhan pengguna, serta struktur sistem. Selain itu, pemanfaatan UML juga dapat membantu mengurangi kesalahan dalam pengembangan sistem karena setiap komponen telah dirancang dan divisualisasikan terlebih dahulu [27].

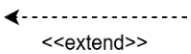
Dalam proses perancangan sistem, UML menyediakan berbagai jenis diagram yang dapat digunakan sesuai kebutuhan, seperti *Use Case Diagram, Component Diagram, Sequence Diagram,* dan *Class Diagram*. Setiap diagram memiliki fungsi yang berbeda, di mana *Use Case Diagram* digunakan untuk menggambarkan interaksi antara pengguna dengan sistem, *Component Diagram* digunakan untuk menunjukkan struktur dan hubungan antar komponen dalam sistem, serta diagram lainnya untuk mendukung pemodelan sistem secara menyeluruh. Dengan adanya berbagai diagram tersebut, UML mampu memberikan gambaran sistem yang lebih lengkap dan terstruktur [28].

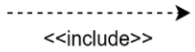
Penerapan UML bertujuan untuk memodelkan sistem perangkat lunak secara sistematis dan mudah dipahami. Dengan menggunakan UML, proses analisis kebutuhan serta perancangan sistem dapat dilakukan dengan lebih terarah, sehingga menghasilkan sistem yang sesuai dengan kebutuhan pengguna. Selain itu, UML juga berfungsi sebagai alat komunikasi antara pengembang dan pengguna agar memiliki pemahaman yang sama terhadap sistem yang akan dibangun [29].

1. *Use Case Diagram*

   Use case diagram merupakan salah satu teknik dalam pengembangan perangkat lunak atau sistem informasi yang digunakan untuk mengidentifikasi kebutuhan fungsional dari suatu sistem. *Use Case* menggambarkan interaksi yang terjadi antara aktor sebagai pengguna dengan sistem yang dikembangkan, di mana setiap interaksi direpresentasikan dalam bentuk urutan langkah-langkah yang sederhana dan mudah dipahami. Melalui *Use Case,* pengembang dapat mengetahui bagaimana sistem akan digunakan serta fungsi-fungsi apa saja yang harus disediakan [30].

Tabel 2.1 Simbol-Simbol Use Case Diagram

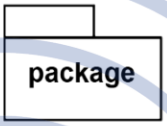
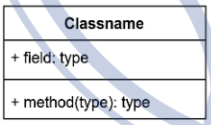
Simbol	Deskripsi
Use Case 	Fungsionalitas yang disediakan oleh sistem sebagai bagian yang dapat berinteraksi dengan aktor atau bagian lain. Biasanya dinyatakan dalam bentuk aktivitas kerja, seperti proses input data atau operasi tertentu.
Aktor 	Pihak yang berinteraksi dengan sistem yang akan dikembangkan, baik berupa manusia, proses, maupun sistem lain di luar sistem utama. Meskipun sering digambarkan sebagai manusia, aktor tidak selalu berupa orang, melainkan bisa berupa peran seperti admin, pengguna, atau pihak lainnya.
Asosiasi	Menunjukkan adanya hubungan atau komunikasi antara aktor dengan *use case*, maupun antara satu *use case* dengan *use case* lainnya.
Extend 	Hubungan yang menunjukkan adanya penambahan fungsi pada suatu *use case*. *Use case* tambahan ini dapat berdiri sendiri tanpa harus selalu bergantung pada *use case* utama. Arah panah menunjukkan ke *use case* yang diperluas.

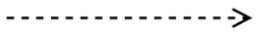
Include  	Hubungan di mana suatu *use case* membutuhkan *use case* lain sebagai bagian dari prosesnya. *Use case* yang di-include harus dijalankan terlebih dahulu sebelum *use case* utama dapat dijalankan. Arah panah menunjukkan ke use case yang menjadi bagian yang harus diproses lebih dulu.

2. *Class Diagram*

Class Diagram merupakan diagram struktur statis dalam UML yang digunakan untuk menggambarkan susunan sistem. Diagram ini menampilkan kelas, atribut, metode, serta hubungan antar objek. *Class Diagram* termasuk diagram struktur karena menunjukkan komponen-komponen yang harus ada dalam sistem. Komponen tersebut dapat merepresentasikan kelas yang akan diimplementasikan, objek utama, maupun hubungan antara kelas dan objek. Kelas sendiri merupakan sekumpulan objek yang memiliki peran dan karakteristik yang sama dalam suatu sistem [30].

Tabel 2.2 Simbol-Simbol *Class Diagram*

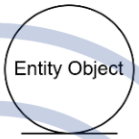
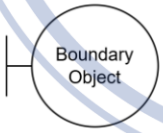
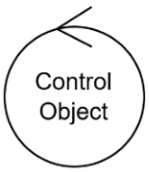
Simbol	Deskripsi
Package  	*Package* merupakan sebuah bungkusan dari satu atau lebih kelas.
Kelas  	Kelas pada struktur sistem, tiap kelas memiliki nama, *attribute*, dan *operation* atau *method*.
Asosiasi 	Relasi antar kelas dengan pengertian umum.
Asosiasi berarah 	Relasi antar kelas dengan pengertian kelas yang satu digunakan oleh kelas yang lain.
Generalisasi 	Relasi antar kelas dengan pengertian generalisasi spesialisasi (umum-khusus).
Kebergantungan	Relasi antar kelas dengan pengertian kebergan tungan antar kelas.

	
Agregasi 	Relasi antar kelas dengan makna semua-sebagian (*whole-part*).

3. *Sequence Diagram*

Sequence diagram merupakan diagram yang digunakan untuk menggambarkan urutan interaksi antar objek dalam suatu sistem secara kronologis. Diagram ini menunjukkan tahapan proses yang terjadi secara berurutan dan logis dalam menjalankan suatu fungsi sesuai dengan skenario pada *Use Case Diagram*. Dengan adanya *Sequence Diagram*, alur komunikasi antar objek dalam sistem dapat dipahami dengan lebih jelas dan terstruktur [30].

Tabel 2.3 Simbol-Simbol *Sequence Diagram*

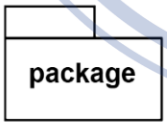
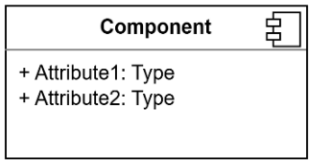
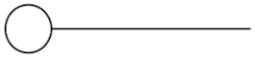
Simbol	Deskripsi
Actor 	Menggambar orang yang sedang berinteraksi dengan sistem.
Entity Class 	Menggambarkan hubungan yang akan di lakukan.
Boundary Class 	Menangani komunikasi antar lingkungan sistem.
Cobtrol Class 	Bertanggung jawab terhadap kelas-kelas terhadap objek yang berisi logika.
Activation	Mewakili proses durasi aktivasi sebuah operasi.

	
Life Line	Komponen yang di gambarkan garis putus terhubung dengan objek
Message	Komunikasi anatar objek yang mengantarkan informasi dan menimbulkan aktifitas.

4. *Component Diagram*

Component diagram merupakan diagram yang digunakan untuk menggambarkan struktur organisasi komponen dalam suatu sistem beserta hubungan dan ketergantungan antar komponen tersebut. Diagram ini menunjukkan bagaimana setiap bagian sistem saling terhubung dan bekerja sama dalam membentuk suatu sistem yang utuh. Dengan menggunakan *component diagram*, susunan komponen serta keterkaitan antar bagian sistem dapat dipahami secara lebih jelas dan terstruktur [30].

Tabel 2.4 Simbol-Simbol *Component Diagram*

Simbol	Deskripsi
Package 	*Package* merupakan wadah yang digunakan untuk mengelompokkan beberapa komponen dalam sistem agar lebih terorganisir dan mudah dikelola.
Component 	*Component* merupakan bagian atau modul dalam sistem yang memiliki fungsi tertentu serta dapat berdiri sendiri dan berinteraksi dengan komponen lainnya.
Interface 	*Interface* merupakan penghubung yang digunakan untuk mendefinisikan layanan atau fungsi yang disediakan oleh suatu

	komponen agar dapat digunakan oleh komponen lain.

2.4 Sistem Rekomendasi

Sistem Rekomendasi adalah sistem penyaringan informasi yang dirancang untuk memprediksi preferensi individu dan menyarankan item seperti produk, film, musik, atau artikel yang kemungkinan besar akan diminati oleh pengguna [9], [31], [7], [32]. Tujuan utama dari sistem ini adalah untuk meningkatkan pengalaman pengguna, meningkatkan keterlibatan, serta memfasilitasi proses pengambilan keputusan di tengah fenomena kelebihan informasi dalam lingkungan data besar [9], [31], [33]. Secara teknis, sistem rekomendasi bekerja dengan mengekstrapolasi fungsi utilitas yang awalnya hanya terdefinisi pada subset pasangan pengguna-item untuk memprediksi nilai utilitas pada seluruh ruang interaksi yang tersedia [32].

Terdapat berbagai faktor yang memengaruhi proses pengambilan keputusan pada sistem dalam memberikan rekomendasi kepada pengguna. Faktor-faktor tersebut mencakup pola perilaku pengguna, deskripsi suatu item, serta preferensi dan kebiasaan kelompok pengguna yang memiliki kemiripan dalam memberikan penilaian terhadap item tertentu. Saat ini, sistem rekomendasi berfungsi sebagai sarana yang efektif dalam proses penyaringan informasi, dengan penekanan pada karakteristik mekanisme penyaringan tersebut. Dalam konteks ini, sistem rekomendasi dirancang untuk mengidentifikasi dan menyarankan produk yang berpotensi menarik perhatian pengguna [7].

Sistem rekomendasi terus mengalami perkembangan pesat dan umumnya dikelompokkan ke dalam tiga jenis metode yang memiliki peran berbeda. *Content-based filtering* (CBF) berfungsi memberikan rekomendasi dengan menganalisis kesamaan informasi antaritem, serta memperhitungkan elemen tambahan yang relevan seperti teks, gambar, maupun video. Sementara itu, *Collaborative filtering* (CF) merupakan metode yang mempelajari hubungan historis antara pengguna dan item, baik melalui penilaian yang pernah diberikan pengguna maupun melalui aktivitas penelusuran sebelumnya. Adapun *Hybrid recommender system* merupakan pendekatan yang menggabungkan kedua metode tersebut, yaitu CBF dan CF [34].

Evolusi sistem rekomendasi modern saat ini semakin berkembang dengan memanfaatkan umpan balik implisit, seperti riwayat penelusuran, klik, dan perilaku transaksi, yang lebih mudah dikumpulkan dalam skala besar dibandingkan penilaian eksplisit berupa rating. Perkembangan ini didukung oleh integrasi algoritma cerdas dan

penggunaan *Application Programming Interface* (API) eksternal yang memungkinkan sistem memberikan rekomendasi secara dinamis, terutama dalam membantu pengguna mengolah bahan makanan yang tersedia di rumah secara efisien. Dengan dukungan arsitektur saraf dan pemrosesan data besar, sistem rekomendasi masa depan diharapkan mampu menyerap informasi kontekstual yang lebih luas untuk menghasilkan rujukan yang semakin personal dan relevan bagi gaya hidup digital saat ini [32], [4].

2.4.1 Content-Based Filtering

Content-Based Filtering (CBF) merupakan pendekatan sistem rekomendasi yang memberikan saran item kepada pengguna berdasarkan analisis kemiripan antara atribut item dengan profil preferensi pengguna yang dibangun dari interaksi di masa lalu [31], [32]. Mekanisme utama pendekatan ini berakar pada proses *profiling*, di mana sistem membangun *Item profile* yang terdiri dari kumpulan fitur deskriptif item (seperti genre, penulis, atau kata kunci teks) dan *user profile* yang mencerminkan bobot ketertarikan individu terhadap fitur-fitur tersebut [32]. Profil pengguna ini dapat diperoleh secara eksplisit melalui kuesioner atau secara implisit dengan mempelajari perilaku transaksi dan riwayat pencarian pengguna dari waktu ke waktu [33], [32]. Melalui pemodelan ini, nilai utilitas atau prediksi ketertarikan pengguna terhadap item baru dihitung berdasarkan sejauh mana fitur item tersebut selaras dengan karakteristik item yang pernah disukai pengguna sebelumnya [31], [7], [32].

Penerapan CBF menawarkan keunggulan signifikan dalam hal independensi data, karena sistem tidak memerlukan akses ke profil atau data pengguna lain untuk menghasilkan rekomendasi bagi pengguna tertentu, sehingga lebih menjamin privasi informasi [33]. Selain itu, pendekatan ini merupakan solusi efektif bagi masalah item *cold-start*, di mana item baru yang masuk ke dalam sistem dapat segera direkomendasikan selama deskripsi kontennya tersedia, tanpa harus menunggu adanya rating dari komunitas pengguna terlebih dahulu [31], [33]. CBF juga memiliki kemampuan adaptasi yang dinamis terhadap perubahan preferensi pengguna secara cepat [33]. Namun, model ini menghadapi tantangan besar terkait analisis konten yang terbatas, terutama kesulitan dalam melakukan ekstraksi fitur otomatis pada data non-tekstual yang kompleks seperti citra, audio, dan video [33], [32].

Tantangan dalam penerapan CBF adalah risiko *overspecialization*, di mana sistem cenderung memberikan rekomendasi yang terlalu homogen atau sangat mirip dengan item yang sudah diketahui pengguna, sehingga membatasi penemuan item baru yang berbeda namun mungkin diminati (*limited serendipity*) [9], [31], [32]. Tantangan signifikan CBF

lainnya adalah potensi kurangnya personalisasi yang mendalam, di mana banyak implementasi metode ini cenderung mengabaikan atribut personal pengguna sehingga menghasilkan rekomendasi yang bersifat non-personalisasi [9]. Hal ini terjadi karena sistem sering kali hanya terpaku pada pencocokan fitur teknis atau kata kunci item tanpa benar-benar menangkap preferensi unik individu secara dinamis [9], [3].

Untuk mengimplementasikan strategi di atas, terdapat beberapa algoritma atau model CBF yang umum digunakan:

1. TF-IDF (*Term Frequency-Inverse Document Frequency*): Algoritma ini digunakan untuk menentukan bobot kepentingan sebuah kata dalam sebuah dokumen berdasarkan frekuensi kemunculannya secara lokal dan kelangkaannya secara global di seluruh korpus [32]. Skor TF-IDF membantu sistem mengidentifikasi istilah kunci yang mencerminkan minat spesifik pengguna terhadap konten tekstual [9].
2. BM25: Sebuah fungsi pemeringkatan probabilitas yang digunakan dalam pencarian informasi untuk memperkirakan relevansi dokumen terhadap kueri pencarian tertentu, yang juga diaplikasikan untuk merepresentasikan item dalam korpus dokumen ilmiah [35].
3. Word2Vec: Metode berbasis saraf yang memetakan kata-kata ke dalam ruang vektor sehingga hubungan semantik antar kata dapat ditangkap berdasarkan konteks distribusinya [35].

2.4.2 Collaborative Filtering

Collaborative Filtering (CF) merupakan pendekatan sistem rekomendasi yang memberikan saran item pada pengguna dengan cara memprediksi preferensi pengguna berdasarkan riwayat interaksi dan pola perilaku komunitas pengguna secara keseluruhan [31], [7]. Berbeda dengan CBF yang mengandalkan atribut intrinsik item, CF bekerja sepenuhnya berdasarkan interaksi pengguna-item (seperti *rating*, klik, atau pembelian) yang disusun dalam sebuah matriks utilitas (*utility matrix*) [7], [33]. Data interaksi yang digunakan dalam CF secara umum dikategorikan menjadi dua jenis, yaitu umpan balik eksplisit (*explicit feedback*) dan umpan balik implisit (*implicit feedback*) [10], [36]. Pendekatan ini berpegang pada asumsi dasar bahwa pengguna yang memiliki pola preferensi serupa di masa lalu cenderung akan memiliki ketertarikan yang sama terhadap item baru di masa depan [9], [31], [33]. Secara mekanis, sistem akan mengidentifikasi "tetangga" (*neighborhood*) bagi pengguna target, yaitu sekelompok pengguna lain yang memberikan

penilaian serupa terhadap sekumpulan item yang sama, kemudian menyarankan item yang disukai oleh kelompok tersebut namun belum pernah dilihat oleh pengguna target [7], [33].

Penerapan CF sangat populer karena memiliki keunggulan dalam memberikan elemen serendipitas dan keragaman, di mana sistem mampu menyarankan item yang berada di luar profil historis pengguna selama item tersebut disukai oleh komunitas yang memiliki selera serupa [7]. Selain itu, CF tidak memerlukan analisis konten item secara mendalam, sehingga sangat efektif digunakan pada domain di mana metadata item sulit diekstraksi secara otomatis atau tidak tersedia secara lengkap [31], [7].

Penerapan CF menghadapi tantangan kritis berupa masalah data *sparsity* (data jarang), di mana sebagian besar sel dalam matriks utilitas kosong karena pengguna biasanya hanya berinteraksi dengan sebagian kecil item yang tersedia [31], [7], [32]. Tantangan lainnya mencakup masalah *cold-start*, di mana sistem kesulitan memberikan rekomendasi akurat bagi pengguna atau item baru yang belum memiliki riwayat interaksi yang cukup [31], [33]. Selain itu, seiring bertambahnya jumlah pengguna dan item, sistem CF tradisional sering kali menghadapi kendala skalabilitas dan efisiensi komputasi [31], [33].

Untuk mengimplementasikan strategi ini, metode CF dibagi ke dalam dua kategori utama beserta model-model turunannya:

1. *Memory-based*

   Metode ini (juga dikenal sebagai *heuristic-based*) menghasilkan rekomendasi dengan menggunakan seluruh koleksi item yang telah diberi *rating* oleh pengguna secara langsung [7], [32]. Efisiensi kategori ini umumnya bergantung pada metrik seperti *Cosine Similarity*, *Pearson Correlation Coefficient* (PCC), dan *Mean Squared Difference* (MSD) untuk menghitung kedekatan antar entitas [7], [32].

   a. *User-based Collaborative Filtering* (UBCF): Menyarankan item yang disukai oleh pengguna lain yang memiliki selera serupa dengan pengguna target [7], [32].

   b. *Item-based Collaborative Filtering* (IBCF): Menyarankan item yang serupa dengan item yang pernah disukai pengguna berdasarkan pola rating dari seluruh pengguna sistem [7], [32].

2. *Model-based*

   Metode ini menggunakan data interaksi untuk melatih model pembelajaran mesin atau statistik yang dapat mengungkap faktor laten yang mewakili preferensi pengguna dan karakteristik item [9], [33], [32].

a. *Matrix Factorization*: Teknik seperti *Singular Value Decomposition* (SVD) dan SVD++ memecah matriks interaksi menjadi vektor fitur laten berdimensi rendah untuk memprediksi nilai yang hilang [7], [33], [32].
b. *Advanced Neural Models*: Penggunaan *Deep Learning* seperti *Neural Collaborative Filtering* (NCF) dan DeepFM memungkinkan penangkapan interaksi non-linier yang kompleks [7].
c. *Graph-based Models*: Algoritma seperti LightGCN memanfaatkan struktur graf bipartit pengguna-item untuk menyebarkan sinyal kolaboratif melalui agregasi lingkungan [7].

2.4.3 Hybrid Filtering

Hybrid Based Filtering adalah pendekatan sistem rekomendasi yang mengintegrasikan dua atau lebih teknik filtrasi yang berbeda untuk meningkatkan akurasi prediksi dan efisiensi sistem secara keseluruhan [9], [37], [31]. Mekanisme utama dari strategi hibrida ini melibatkan penggabungan kekuatan unik dari CF, yang mengandalkan pola perilaku komunitas, dengan CBF, yang berfokus pada atribut intrinsik item [37]. Secara teknis, sistem hibrida bekerja dengan mengagregasi data interaksi pengguna dan fitur konten item untuk membangun profil preferensi yang lebih mendalam, sehingga dapat menghasilkan keputusan rekomendasi yang lebih *personal* [37]. Salah satu implementasi yang umum dilakukan adalah melalui pendekatan skor terboboti (*weighted score*), di mana prediksi akhir merupakan kombinasi linear dari skor yang dihasilkan oleh model kolaboratif dan berbasis konten menggunakan faktor penyeimbang (α) tertentu untuk menentukan tingkat kepentingan masing-masing metode [37].

Penerapan filtrasi hibrida menawarkan keunggulan signifikan, terutama dalam memitigasi keterbatasan inheren pada metode tunggal, seperti masalah data *sparsity* (data jarang) dan *cold-start* [37]. Dengan menyatukan informasi atribut item ke dalam model interaksi komunitas, sistem hibrida tetap dapat memberikan rujukan yang relevan bagi item baru yang belum memiliki *rating* atau bagi pengguna baru yang memiliki riwayat interaksi terbatas [37]. Selain itu, metode hibrida terbukti secara empiris mampu memberikan tingkat presisi yang lebih tinggi dan hasil rujukan yang lebih beragam dibandingkan pendekatan tradisional [9], [37].

Sistem ini memiliki tantangan besar terkait peningkatan kompleksitas arsitektur dan beban komputasi yang lebih berat dibandingkan model tunggal [9], [7]. Selain itu, proses integrasi data dari berbagai sumber informasi yang berbeda sering kali memerlukan

standarisasi format data dan koordinasi protokol akses yang rumit untuk menjaga akurasi serta privasi informasi pengguna [37].

Untuk mengimplementasikan strategi ini, terdapat beberapa metode hibridisasi yang umum dikategorikan dalam sebuah sistem rekomendasi:

1. Metode Skor Terboboti (*Weighted Score*): Menggabungkan hasil prediksi dari model CF dan CBF dengan memberikan bobot numerik tertentu pada masing-masing hasil. Penelitian menunjukkan bahwa kombinasi seperti 80% *Collaborative Filtering* dan 20% *Content-Based Filtering* dapat memberikan efisiensi rujukan yang lebih tinggi dibandingkan pendekatan tradisional [33].
2. *Feature Combination*: Menyuntikkan fitur dari satu metode ke metode lainnya, misalnya fitur konten dimasukkan ke dalam model kolaboratif [33].
3. *Cascading*: Menggunakan satu teknik untuk menghasilkan daftar kandidat awal, yang kemudian diperhalus oleh teknik kedua [33].
4. *Switching*: Sistem memilih salah satu teknik rekomendasi berdasarkan situasi atau kriteria kualitas tertentu [33], [32].
5. *Feature Augmentation*: Merupakan pendekatan *hybrid* yang menghasilkan fitur tambahan dari keluaran suatu model rekomendasi, seperti skor prediksi atau representasi laten, untuk digunakan sebagai masukan pada model rekomendasi lain guna meningkatkan kualitas prediksi [33].

2.5 Model-Based Collaborative Filtering

Model-Based Collaborative Filtering merupakan teknik sistem rekomendasi yang memanfaatkan algoritma pembelajaran mesin dan penggalian data untuk membangun model prediktif berdasarkan riwayat interaksi pengguna [33], [38], [39]. Berbeda dengan metode *memory-based* yang mengandalkan seluruh *dataset* secara langsung untuk setiap prediksi, pendekatan ini mengekstraksi fitur-fitur penting dari data untuk membentuk representasi faktor laten dalam ruang berdimensi rendah[33], [39], [31]. Melalui pemodelan matematis ini, sistem tidak hanya bergantung pada kecocokan langsung antar pengguna, tetapi mampu mengungkap pola tersembunyi yang mendasari hubungan antara pengguna dan item guna menghasilkan rujukan yang lebih akurat dan andal[39], [31].

Salah satu teknik yang paling populer dalam kategori ini adalah *Matrix Factorization* (MF), yang bekerja dengan mendekomposisi matriks interaksi pengguna-item menjadi dua matriks laten yang mewakili karakteristik unik pengguna dan item [31], [10], [7], [40]. Proses optimasi pada model ini sering kali dilakukan menggunakan algoritma *Alternating*

Least Squares (ALS) atau *Stochastic Gradient Descent* (SGD) untuk meminimalkan fungsi kerugian dan kesalahan prediksi [10], [36], [41], [38]. Keunggulan utama MF terletak pada kemampuannya dalam menangani masalah *data sparsity* (data jarang) dengan lebih efisien serta memiliki skalabilitas yang sangat baik untuk menangani dataset berskala besar dalam ekosistem *big data* [11], [7], [42].

Evolusi sistem rekomendasi *model-based* saat ini telah mengintegrasikan arsitektur jaringan saraf tiruan melalui kerangka kerja *Neural Collaborative Filtering* (NCF) [10], [40], [39]. NCF mengatasi keterbatasan linieritas pada MF tradisional dengan menggantikan fungsi produk titik (*inner product*) sederhana dengan lapisan saraf non-linier seperti *Multi-Layer Perceptron* (MLP) guna menangkap struktur interaksi pengguna yang lebih kompleks [10], [40], [42]. Dengan menggabungkan kekuatan linieritas dari *Generalized Matrix Factorization* (GMF) dan fleksibilitas non-linieritas dari MLP, model ini mampu memberikan presisi yang lebih tinggi, terutama saat bekerja dengan data umpan balik implisit (*implicit feedback*) seperti riwayat klik dan transaksi [7], [40] , [3].

Secara keseluruhan, pendekatan *model-based* CF menawarkan solusi yang lebih kuat dalam hal generalisasi dan efisiensi penyimpanan dibandingkan metode berbasis heuristik [11], [33]. Selain MF dan NCF, terdapat model modern seperti EASE-R yang menggunakan solusi bentuk tertutup untuk mencapai kecepatan pelatihan yang drastis, serta LightGCN yang memanfaatkan propagasi pada graf interaksi untuk memperkaya representasi pengguna [11], [7]. Meskipun menawarkan akurasi yang lebih tinggi, pengembangan model-model ini tetap menghadapi tantangan berupa kebutuhan sumber daya komputasi yang besar serta kompleksitas dalam penyetelan hyperparameter agar tetap relevan dalam lingkungan aplikasi waktu nyata [11], [7], [42].

2.5.1 Alternating Least Squares (ALS)

Alternating Least Squares (ALS) adalah algoritma *Matrix Factorization* (MF) yang digunakan untuk memprediksi preferensi pengguna dengan memetakan pengguna dan item ke dalam ruang laten berdimensi rendah. Algoritma ini bekerja dengan menguraikan matriks interaksi pengguna-item menjadi produk dari dua matriks yang lebih kecil, yaitu matriks fitur pengguna dan matriks fitur item. ALS sangat populer dalam sistem rekomendasi karena kemampuannya untuk menangani data umpan balik eksplisit maupun implisit secara efektif. Dengan meminimalkan fungsi kerugian kuadrat (*square loss*), sistem berusaha menemukan nilai faktor laten yang paling akurat untuk menggambarkan hubungan antar entitas dalam lingkungan data besar [41].

Inti dari mekanisme ALS terletak pada sifatnya yang bergantian (*alternating*) dalam proses optimasi parameter model. Dalam setiap iterasi, sistem melakukan dua langkah utama yaitu langkah-P untuk menghitung ulang fitur pengguna dengan menganggap fitur item tetap, dan langkah-Q untuk menghitung ulang fitur item dengan menganggap fitur pengguna tetap. Karena salah satu variabel dianggap konstan dalam satu langkah, masalah optimasi yang kompleks berubah menjadi serangkaian sub-masalah regresi linear (*ridge regression*) yang dapat diselesaikan secara analitik. Untuk mencegah terjadinya *overfitting* atau bobot model yang terlalu besar, sistem menerapkan parameter regularisasi (λ) selama proses perhitungan [41].

Secara garis besar, langkah-langkah algoritma ALS dimulai dengan inisialisasi matriks fitur pengguna (P) dan matriks fitur item (Q) dengan nilai acak. Setelah inisialisasi, algoritma memasuki siklus iteratif yang dimulai dengan memperbaiki matriks Q untuk memperbarui setiap vektor pengguna P_u berdasarkan interaksi historisnya. Langkah selanjutnya adalah memperbaiki matriks P yang baru saja diperbarui untuk menghitung ulang setiap vektor item Q_i menggunakan prinsip yang sama. Proses ini terus diulang selama beberapa putaran, biasanya antara sepuluh hingga beberapa puluh kali, hingga mencapai kondisi terminasi atau hingga nilai parameter menjadi stabil dan konvergen [41].

Keunggulan teknis utama dari ALS adalah efisiensinya dalam menangani dataset skala besar melalui kemungkinan penggunaan komputasi paralel. Kompleksitas waktu algoritma ini secara teoritis bergantung pada jumlah interaksi dan jumlah faktor laten, yang menjadikannya sangat skalabel untuk sistem dengan jutaan pengguna. Penelitian terbaru bahkan telah mengembangkan varian ALS yang lebih cepat, seperti ALS1, yang mengurangi kompleksitas komputasi dari kubik (K3) menjadi linear (K) terhadap jumlah fitur laten. Berkat performa dan kecepatannya yang andal, ALS telah menjadi salah satu algoritma standar industri yang luas digunakan pada *platform* hiburan dan *e-commerce* [41].

2.5.2 Neural Collaborative Filtering (NCF)

Neural Collaborative Filtering (NCF) adalah kerangka kerja umum yang dikembangkan untuk mengatasi keterbatasan inheren dari *Matrix Factorization* (MF) tradisional yang sangat bergantung pada produk titik (*inner product*) sederhana untuk memodelkan interaksi pengguna-item. Meskipun MF sangat populer, fungsi interaksi liniernya seringkali dianggap tidak cukup untuk menangkap struktur interaksi pengguna yang kompleks dan non-linier dalam data interaksi nyata. Untuk mengatasi hal ini, NCF mengganti produk titik tersebut dengan arsitektur saraf berlapis yang mampu mempelajari

fungsi interaksi arbitrer secara langsung dari data. Kerangka kerja ini memanfaatkan representasi berlapis yang dimulai dari lapisan embedding untuk memetakan identitas pengguna dan item yang jarang (*sparse*) menjadi vektor laten padat, yang kemudian diproses melalui lapisan-lapisan saraf kolaboratif untuk memprediksi skor interaksi pengguna terhadap item [40].

Salah satu implementasi inti dari kerangka kerja ini melibatkan penggunaan *Multi-Layer Perceptron* (MLP) untuk menangkap pola-pola non-linier yang kompleks melalui tumpukan lapisan dengan fungsi aktivasi non-linier, seperti *Rectified Linear Unit* (ReLU). NCF juga mengintegrasikan *Generalized Matrix Factorization* (GMF) yang memungkinkan model untuk mengekspresikan dan menggeneralisasi standar MF, sekaligus memberikan fleksibilitas untuk mempelajari tingkat kepentingan dari berbagai dimensi laten yang berbeda. Kedua pendekatan ini dapat digabungkan ke dalam satu arsitektur tunggal yang dikenal sebagai *Neural Matrix Factorization* (NeuMF), yang menyatukan kekuatan pemodelan linier dari GMF dan fleksibilitas pemodelan non-linier dari MLP untuk meningkatkan kinerja rekomendasi secara keseluruhan. Dalam proses pelatihannya, NCF umumnya menggunakan fungsi objektif *log loss* (atau *binary cross-entropy*) untuk menangani data *implicit feedback* secara efektif sebagai masalah klasifikasi biner [40]. NCF memiliki keunggulan signifikan dalam hal kemampuan non-linier, di mana arsitekturnya mampu menangkap pola interaksi pengguna-item yang kompleks melampaui batasan operasi *dot product* linier [40], [42]. Selain itu, model ini sangat fleksibel secara arsitektural, memungkinkan peneliti untuk melakukan modifikasi pada kedalaman lapisan MLP maupun teknik integrasi *embedding* guna mengoptimalkan performa sesuai dengan kebutuhan dataset yang digunakan [40]. Meskipun memiliki berbagai kelebihan, NCF memiliki kelemahan utama pada aspek efisiensi, di mana beban komputasinya memerlukan waktu pelatihan yang panjang dan dukungan perangkat keras (GPU) yang mumpuni. Selain itu, dari sisi performa praktis, beberapa penelitian menunjukkan bahwa NCF tidak secara konsisten mengungguli metode *Matrix Factorization* yang telah di-*tuning* dengan baik, namun tetap menghasilkan *overhead* operasional yang lebih tinggi [43].

Neural Collaborative Filtering (NCF) dapat dijelaskan sebagai kerangka kerja (*framework*) yang menggantikan fungsi produk titik (*inner product*) linier pada model *Matrix Factorization* tradisional dengan arsitektur jaringan saraf tiruan untuk mempelajari fungsi interaksi pengguna-item yang non-linier dari data. Berikut adalah gambaran besar langkah-langkah algoritma NCF:

1. Input Layer (Sparse)

Proses dimulai dengan mengambil identitas unik pengguna (u) dan item (i) yang dikonversi menjadi vektor sparse menggunakan one-hot encoding. Vektor ini berfungsi sebagai representasi awal yang sangat jarang (*sparse*) dari setiap entitas dalam sistem.

2. Embedding Layer: Vektor sparse tersebut kemudian diproyeksikan melalui lapisan terhubung penuh (*fully connected layer*) ke dalam ruang laten berdimensi rendah untuk menghasilkan vektor laten (*embedding*) yang padat (*dense*). Vektor ini mewakili fitur-fitur laten dari pengguna dan item dalam konteks model faktor laten.
3. Neural CF Layers: Vektor laten pengguna dan item kemudian dimasukkan ke dalam arsitektur saraf berlapis yang disebut sebagai *Neural CF Layers*. Pada tahap ini, interaksi antara pengguna dan item diproses melalui dua jalur utama:
    a. Jalur MLP (*Multi-Layer Perceptron*): Menggabungkan vektor pengguna dan item melalui konkatenasi, diikuti oleh serangkaian lapisan tersembunyi (*hidden layers*) dengan fungsi aktivasi non-linier seperti ReLU untuk menangkap struktur interaksi yang kompleks.
    b. Jalur GMF (*Generalized Matrix Factorization*): Menggunakan produk elemen-demi-elemen (*element-wise product*) untuk menangkap interaksi linier yang mirip dengan Matrix Factorization standar.
4. *Output Layer*: Hasil dari lapisan tersembunyi terakhir dipetakan ke lapisan *output* untuk menghasilkan skor prediksi y^{ui}. Skor ini biasanya menggunakan fungsi aktivasi seperti Sigmoid untuk membatasi nilai dalam rentang yang merepresentasikan probabilitas seberapa besar kemungkinan seorang pengguna akan berinteraksi dengan item tersebut.
5. Optimasi (*Training*): Model dilatih dengan meminimalkan fungsi *binary cross-entropy loss* (*log loss*). Pendekatan ini memperlakukan proses rekomendasi sebagai masalah klasifikasi biner, di mana sistem mengoptimalkan prediksi untuk membedakan antara interaksi yang teramati (label 1) dan interaksi yang tidak teramati melalui proses negative sampling.

Dengan struktur ini, NCF mampu memberikan fleksibilitas yang lebih besar dalam memodelkan hubungan antara pengguna dan item dibandingkan model linier konvensional, menjadikannya standar pembanding yang kuat bagi model-model sistem rekomendasi berbasis pembelajaran dalam (*deep learning*) lainnya [40]

2.5.3 EASE-R

Embarrassingly Shallow AutoEncoder atau EASE-R adalah model *collaborative filtering* linier untuk sistem rekomendasi yang dirancang khusus untuk menangani data yang

bersifat jarang (*sparse*) seperti umpan balik implisit (*implicit feedback*). Berbeda dengan tren *deep learning* yang menggunakan banyak lapisan, model ini dikategorikan sebagai *autoencoder* yang sangat dangkal karena tidak memiliki lapisan tersembunyi (*hidden layer*) sama sekali. Model Ini diperkenalkan oleh Harald Steck pada tahun 2019. Bagian nama model yaitu "*embarrassingly shallow*" dicetuskan karena memang sangat dangkal. Tidak ada *deep network*, tidak ada *nonlinear activation*. Hanya satu *layer linear*. Ironisnya, justru itu yang membuatnya kuat, dan efisien [12].

Model ini memprediksi item preferensi pengguna dengan mereproduksi vektor input (interaksi masa lalu pengguna) menjadi vektor *output*, di mana vektor *input* biner menunjukkan tidak ada atau adanya interaksi antara *item* dan pengguna. Parameter utama dari model ini adalah matriks bobot antar-*item* (*item-item weight matrix*) dan regularisasi L2. Skor prediksi $S_{u,j}$ untuk pengguna u terhadap item j dihitung melalui produk titik (*dot product*) sebagai berikut [12]:

$$S_{u,j} = X_u \cdot B_j \text{.....(2.1)}$$

Dimana:

X_u = Baris interaksi pengguna u

$S_{u,j}$ = Skor prediksi (skalar) untuk pengguna Xu pada bobot *item* Bj

B_j = kolom matriks bobot antar *item* j

Pelatihan EASE^R dilakukan dengan meminimalkan fungsi kerugian kuadrat (*square loss*) menggunakan Frobenius Norm, ditambah dengan regularisasi L2 untuk mencegah *overfitting*. Dalam proses ini, parameter λ adalah satu-satunya hyper-parameter yang perlu dioptimalkan dalam model ini. EASE^R memaksakan batasan bahwa diagonal matriks bobot harus nol (diag(B)=0). Batasan ini secara matematis mencegah solusi trivial di mana model hanya menyarankan item yang sama dengan yang sudah pernah dibeli pengguna (menghindari B=I), Hal ini memaksa model untuk belajar memprediksi suatu item berdasarkan hubungannya dengan *item-item* lain dalam riwayat pengguna [12].

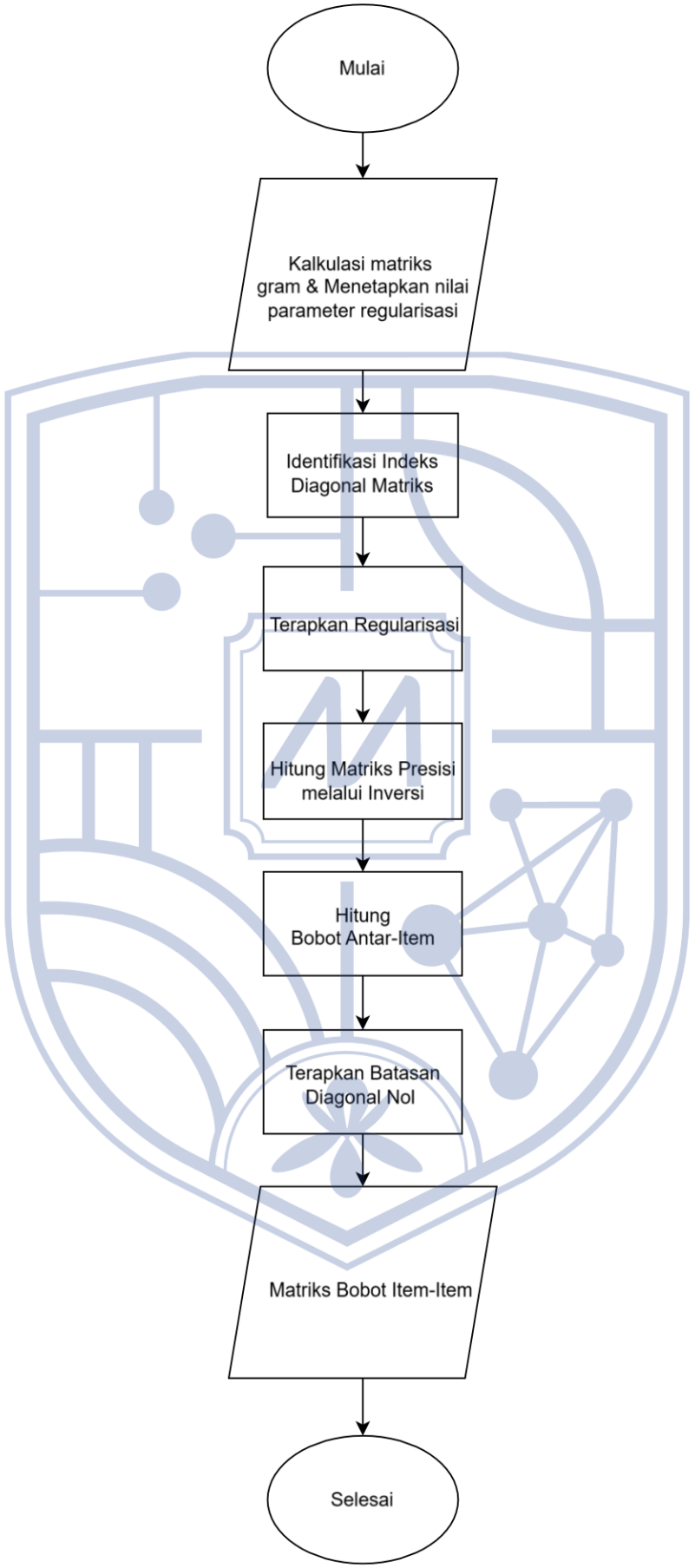
Tantangan utama dalam model linear dengan batasan seperti ini adalah proses optimasi yang biasanya sangat berat secara komputasi; model pendahulu seperti SLIM harus memecahkan masalah optimasi ini secara iteratif untuk setiap item secara terpisah, yang mengakibatkan waktu pelatihan menjadi sangat lama pada dataset skala besar. Untuk mengatasi kendala efisiensi tersebut, Steck memanfaatkan sifat matematis *square loss* sebagai fungsi objektif karena sifat matematisnya yang memungkinkan pencapaian solusi bentuk tertutup (*closed-form solution*). Karena model ini memiliki batasan khusus di mana

diagonal matriks bobot harus nol, Steck menggunakan Metode Pengali Lagrangian (*Lagrangian Multipliers*) untuk mengintegrasikan batasan tersebut langsung ke dalam fungsi objektif. Kombinasi antara fungsi kerugian kuadrat dan metode Lagrangian ini membuktikan bahwa parameter optimal dapat ditemukan secara analitis melalui satu rangkaian operasi aljabar matriks saja. Hal ini secara radikal memangkas waktu pelatihan dari hitungan hari menjadi hanya beberapa menit, tanpa memerlukan proses iteratif seperti *Stochastic Gradient Descent* (SGD). Bagian ini adalah keunggulan teknis EASE^R yang membedakannya dari model deep learning iteratif [12].

2.5.4 Tahapan Algoritma Rekomendasi EASE-R

Penjabaran mengenai tahapan algoritma pada bagian ini bertujuan untuk memberikan panduan sistematis mengenai implementasi teknis model EASE-R dari perspektif komputasi. Penjelasan ini berfungsi sebagai jembatan yang menghubungkan derivasi matematis dari solusi bentuk tertutup (*closed-form solution*) dan metode Pengali Lagrangian yang telah dibahas sebelumnya ke dalam rangkaian operasi aljabar matriks yang konkret [12].

Melalui pemahaman alur kerja ini, dapat ditunjukkan bagaimana kompleksitas teori mengenai batasan diagonal nol dan regulasi disederhanakan menjadi prosedur yang sangat efisien, sehingga memungkinkan model untuk dilatih dalam waktu yang jauh lebih singkat dibandingkan metode iteratif konvensional seperti *Stochastic Gradient Descent* (SGD) atau model SLIM. Selain itu, pemaparan tahapan ini juga dimaksudkan untuk menggarisbawahi efektivitas desain model yang "sangat dangkal" (*embarrassingly shallow*), di mana hanya dengan beberapa baris kode operasi matriks, sistem mampu menghasilkan akurasi pemeringkatan yang kompetitif pada data yang jarang (*sparse*). Berikut *flowchart* tahapan algoritma ease-r dan penjelasan setiap tahapannya..



Gambar 2.2 *Flowchart Algoritma EASE-R*

1. Kalkulasi Matriks Gram & Penetapan Nilai Parameter Regularisasi Sebagai Input

   Algoritma dimulai dengan menerima dua input utama: matriks Gram $G = X^T X$ dan parameter regulasi $L_2(\lambda)$. Matriks Gram merupakan statistik yang cukup (*sufficient statistics*) untuk mengestimasi bobot, yang jika dibangun dari data biner *sparse* X, akan merepresentasikan matriks *co-occurrence* antar item. Parameter λ adalah satu-satunya *hyper-parameter* dalam model ini yang berfungsi untuk mengontrol *overfitting* dan memastikan stabilitas perhitungan matematis. Berikut rumus tahap pertama secara lengkap [12].

   $$G = X^T X \quad \text{.....(2.2)}$$

   Dimana:

   G = Matriks Gram Matriks item-item yang dihasilkan dari perkalian $X^T X$ dan berfungsi sebagai statistik yang cukup untuk estimasi model

   X = Matriks interaksi

   X^T = Transpose Matriks X

   Dari tahap ini, kita akan memperoleh 2 hal penting, yaitu:

   a. Jumlah total interaksi suatu item dari keseluruhan user (*item popularity)*

      $$G_{i,i} = \sum_{u=1}^{n} X_{u,i} \cdot X_{u,i} \quad \text{.....(2.3)}$$

      Dimana:

      $G_{i,i}$ = Elemen diagonal matriks Gram yang merepresentasikan jumlah interaksi pada item ke-i

      $x_{u,i}$ = Nilai interaksi antara user ke-u dengan item ke-i, bernilai 1 jika terjadi interaksi dan 0 jika tidak

      n = Jumlah total user

   b. Jumlah total interaksi antara dua item yang berbeda

      $$G_{i,j} = \sum_{u=1}^{n} X_{u,i} \cdot X_{u,j} \quad \text{.....(2.4)}$$

      Dimana:

      $G_{i,j}$ = Elemen matriks Gram yang merepresentasikan jumlah interaksi bersama antara item ke-i dan item ke-j

      $X_{u,i}$ = Nilai interaksi user ke-u terhadap item ke-i

      $X_{u,i}$ = Nilai interaksi user ke-u terhadap item ke-j

n = Jumlah total user

2. Identifikasi Indeks Diagonal Matrix G

Langkah kedua adalah mengidentifikasi indeks diagonal dari matriks G. Hal ini dilakukan untuk memisahkan elemen-elemen di mana suatu item berinteraksi dengan dirinya sendiri $i = j$ dari elemen-elemen off-diagonal yang merepresentasikan interaksi antar item yang berbeda. Identifikasi ini krusial untuk penerapan regulasi dan batasan diagonal nol nantinya. Berikut rumus tahap kedua secara lengkap [12].

$$diag(G) = \{G_{ii} | \, i = 1, \ldots, |I|\} \quad \text{.....(2.5)}$$

Dimana:

diag(G) = Kumpulan elemen diagonal dari matriks G

G_{ii} = Elemen spesifik pada matriks G di mana indeks baris (i) sama dengan indeks kolom (i)

i = 1,…|I| = indeks urutan item, mulai dari item pertama hingga item terkahir dalam dataset.

|I| = simbol yang menyatakan ukuran set item atau total jumlah item unik yang ada dalam sistem.

3. Terapkan Regularisasi Lambda

Parameter regulasi λ kemudian ditambahkan secara khusus pada elemen-elemen di sepanjang diagonal matriks Gram ($G_{ii} + \lambda$). Penambahan nilai λ ini memastikan bahwa matriks tersebut bersifat invertible atau memiliki invers, yang merupakan syarat utama untuk menemukan solusi bentuk tertutup (*closed-form solution*). Berikut rumus tahap ketiga secara lengkap [12] :

$$L_2 : G_{ii} = G_{ii} + \lambda \quad \text{.....(2.6)}$$

Dimana:

L_2 = Menunjukan jenis regularisasi Tikhonov atau *Ridge Regularization* yang diterapkan pada model untuk mengkontrol kompleksitas bobot dan mencegah *overfitting*

G_{ii} = Elemen diagonal pada baris ke-I dan kolom ke-I dari matriks gram (G). Elemen ini merepresentasikan jumlah interaksi atau kemunculan suatu item dalam data. Lambda = Satu-satunya hyper-parameter dalam model EASE-R yang mewakili kekuatan regularisasi. Nilai ini biasanya dioptimasi melalui *validation set*.

4. Hitung Matriks Presisi melalui Inversi

Setelah diregulasi, matriks Gram diinversi untuk menghasilkan matriks presisi ($\hat{P}$), di mana $\hat{P} = (X^T X + \lambda I)^{-1}$. Dalam konteks model grafis, matriks presisi dianggap sebagai matriks kesamaan (*similarity matrix*) yang paling benar secara konsep karena merepresentasikan ketergantungan kondisional antar variabel, berbeda dengan matriks kovarians tradisional. Berikut rumus tahap ketiga secara lengkap [12]:

$$\hat{P} = (G + \lambda I)^{-1} \quad \text{.....(2.7)}$$

Dimana:

P (Matriks Presisi) = Merupakan estimasi dari matriks presisi atau matriks konsentrasi

G = Matriks Gram

λ = Parameter regularisasi L_2

I = Matriks identitas Matriks persegi di mana elemen diagonal utamanya adalah satu dan elemen lainnya adalah no

$()^{-1}$(Inversi Matriks) = Operasi matematika untuk menemukan kebalikan Matriks (G + Lambda)

5. Hitung Bobot antar Item

Bobot antar item dalam matriks dihitung dengan membagi setiap elemen dalam baris matriks presisi dengan nilai negatif dari elemen diagonalnya sendiri, Matriks bobot dihitung menggunakan rumus berikut [12]:

$$B = -\frac{P}{diag(P)} \quad \text{.....(2.8)}$$

Dimana:

P = matriks hasil inversi gram matriks yang telah di regularisasi

$diag(P)$ = nilai diagonal dari matriks P

B = matriks bobot item–item yang dihasilkan oleh model

Proses ini memungkinkan model untuk mempelajari tidak hanya kesamaan (*similarity*) melalui bobot positif, tetapi juga ketidaksamaan (*dissimilarity*) melalui bobot negatif, yang terbukti sangat penting untuk akurasi pemeringkatan resep atau item.

$$\hat{B}_{i,j} = -\frac{\hat{P}_{ij}}{\hat{P}_{jj}} \quad \text{.....(2.9)}$$

Dimana:

$\hat{B}_{i,j}$ = nilai bobot optimal yang menghubungkan item I (*input*) dengan item (*output*)

$\hat{P}_{ij}$ = Elemen pada baris ke-I dan kolom ke-j dari matriks presisi (P). Nilai ini merepresentasikan hubungan atau ketergantungan kondisional antara item i dan item .

$\hat{P}_{jj}$ = Elemen diagonal pada kolom ke-j dari matriks presisi yang berfungsi sebagai faktor normalisasi untuk kolom tersebut.

Tanda negatif (-) = Bagian dari derivasi matematis menggunakan pengali *Langrangian* untuk memenuhi batasan diagonal nol.

6. Terapkan batasan Diagonal nol

Langkah terakhir dalam modifikasi matriks adalah memaksa semua elemen diagonal matriks bobot menjadi nol. Batasan ini sangat krusial guna mencegah model mempelajari solusi trivial (di mana model hanya merekomendasikan apa yang sudah pernah dilihat pengguna) dan memaksa model untuk melakukan generalisasi saat mereproduksi input sebagai *output* Berikut rumus tahap ketiga secara lengkap [12]:

$$diag(B) = 0 \quad \text{.....(2.10)}$$

Dimana:

B (Matriks Bobot) = Matriks parameter item-item berukutan |I| x |I| yang dipelajari oleh model.

diag() = Operasi yang merujuk pada elemen-elemen diagonal utama dari matriks B

Atau secara elemen-per-elemen:

$$B_{ii} = 0, untuk\ setiap\ i \in \{1, \dots, |I|\} \quad \text{.....(2.11)}$$

Dimana:

B_{ii} = Elemen spesifik pada baris ke-i kolom ke-I dalam matriks bobot

I ∈ {1,…,|I|} = indeks untuk setiap item dalam sistem,di mana |I| adalah total jumlah item unik.

0 = Nilai mutlak yang ditetapkan pada elemen diagonal untuk meniadakan pengaruh *self-similarity.*

7. Matriks Bobot item-item sebagai output

Hasil akhir dari algoritma ini adalah matriks bobot item-item yang padat (*dense*) yang telah diintegrasikan batasan diagonal nol ($diag(B) = 0$).

Matriks ini siap digunakan untuk menghasilkan skor prediksi bagi pengguna melalui operasi *dot product* dengan riwayat interaksi mereka, sehingga sistem dapat memberikan rekomendasi item yang paling relevan secara personal Berikut rumus tahap ketiga secara lengkap [12]:

$$\hat{B}_{i,j} = \begin{cases} 0 & jika\ i = j \\ -\frac{\hat{P}_{ij}}{\hat{P}_{jj}} & jika\ i \neq j \end{cases} \quad \text{.....(2.12)}$$

Dimana:

$\hat{B}_{i,j}$ = Nilai bobot optimal dalam matriks item-item yang menghubungkan item i (*input*) dengan item j (*output*).

$\hat{P}$ = matriks presisi.

$\hat{P}_{ij}$ = Elemen pada baris ke-i dan kolom ke-j dari matriks presisi yang menunjukkan ketergantungan kondisional antar item.

$\hat{P}_{jj}$ = Elemen diagonal pada kolom ke-i dari matriks presisi yang berfungsi sebagai faktor pembagi atau normalisasi untuk kolom tersebut.

2.6 Popularity Baseline

Popularity baseline sering disebut sebagai *MostPop* atau *ItemPop*, merupakan metode rekomendasi non-personalisasi yang paling sederhana, di mana sistem menyarankan item kepada pengguna murni berdasarkan jumlah interaksi atau frekuensi kemunculan item tersebut dalam data latih [44]. Dalam pengembangan sistem rekomendasi, metode ini berfungsi sebagai batas acuan minimal (*benchmark*) untuk mengevaluasi sejauh mana model utama mampu memberikan hasil yang lebih baik daripada sekadar mengikuti tren massa [40]. Kehadiran baseline ini sangat krusial karena sebuah model personalisasi dianggap benar-benar memberikan nilai tambah jika performanya mampu melampaui statistik popularitas pada metrik pemeringkatan yang ditentukan, seperti *Hit Ratio* (HR), *Recall* dan *Normalized Discounted Cumulative Gain* (NDCG) [40], [42].

Meskipun bersifat sederhana, metode ini seringkali memiliki daya prediksi yang cukup kuat karena kecenderungan pengguna dalam ekosistem digital untuk berkonsentrasi pada sejumlah kecil item yang sangat populer [36]. Namun, keterbatasan fundamental dari pendekatan ini adalah pengabaian total terhadap preferensi unik individu, di mana setiap pengguna akan menerima daftar rekomendasi yang identik terlepas dari riwayat interaksi atau selera pribadi mereka [44], [41]. Oleh karena itu, perbandingan dengan *Popularity Baseline* digunakan untuk memverifikasi apakah model utama, seperti pemfaktoran matriks atau jaringan saraf, telah berhasil menangkap sinyal personalisasi yang lebih mendalam daripada sekadar mengeksploitasi item-item yang sedang tren di kalangan komunitas [10], [40].

Analisis terhadap perilaku pengguna menunjukkan adanya perbedaan pola konsumsi: pengguna dengan riwayat aktivitas rendah cenderung mengikuti arus dan mengonsumsi item

populer, sementara pengguna yang sangat aktif biasanya memiliki preferensi yang lebih spesifik dan personal [44]. Model utama yang ideal harus mampu memberikan rujukan yang akurat bagi kedua kelompok tersebut, terutama dalam menyarankan item-item *long-tail* yang kurang populer namun sangat relevan bagi individu tertentu [12]. Dengan menempatkan popularitas sebagai pembanding, peneliti dapat memastikan bahwa model tidak hanya "bermain aman" dengan menyarankan item yang sudah pasti disukai banyak orang, tetapi benar-benar mempelajari pola preferensi yang unik dari data interaksi [36].

Selain sebagai metrik akurasi, penggunaan *Popularity Baseline* juga membantu mengidentifikasi adanya bias popularitas (*popularity bias*) yang sering kali menyebabkan item-item baru atau *niche* terpinggirkan dalam daftar rekomendasi [45]. Tanpa adanya pembanding ini, sulit untuk menentukan apakah model yang dikembangkan memberikan personalisasi sejati atau justru memperkuat dominasi item populer yang dapat menurunkan keragaman (*diversity*) dan serendipitas hasil rujukan [45]. Sebagai standar kontrol, *Popularity Baseline* menjamin bahwa evaluasi model utama dilakukan secara objektif untuk membuktikan kemampuannya dalam memberikan pengalaman pengguna yang lebih personal dan relevan dibandingkan metode frekuensi sederhana [38], [44]. Rumus *popularity baseline* MostPop dapat dinyatakan sebagai berikut:

$$Score(i) = \sum_{u \in U_{train}} I(u,i) \quad \text{.....(2.13)}$$

Dimana:

Score(i) = skor popularitas item i

U_{train} = kumpulan pengguna dalam data latih

$I(u,i)$ = total jumlah pengguna dalam sistem

2.7 Recall@K

Recall@K merupakan metrik evaluasi yang digunakan untuk mengukur cakupan (coverage) atau proporsi item relevan yang berhasil ditemukan oleh sistem di dalam daftar K rekomendasi teratas [11]. Dalam konteks sistem rekomendasi, metrik ini berfokus pada kemampuan model untuk menangkap sebanyak mungkin item yang diminati pengguna dari keseluruhan kumpulan item yang dianggap relevan dalam data uji [9], [7]. Berbeda dengan presisi yang menitikberatkan pada ketepatan setiap item yang disarankan, recall memberikan gambaran mengenai seberapa efektif sistem dalam menyajikan informasi yang benar-benar diinginkan tanpa melewatkan item-item penting yang ada dalam preferensi pengguna [32], [38].

Secara matematis, Recall@K dihitung dengan membagi jumlah item relevan yang muncul dalam daftar Top−K dengan total jumlah item relevan yang seharusnya direkomendasikan kepada pengguna tersebut [7], [38]. Persamaan metrik ini merepresentasikan rasio antara irisan dari himpunan item yang direkomendasikan pada posisi K dengan himpunan item ground truth terhadap total jumlah item pada ground truth [7], [46]. Nilai recall berada dalam rentang antara 0 hingga 1, di mana skor yang semakin mendekati nilai 1 mengindikasikan bahwa model rekomendasi tersebut semakin efektif dalam mengidentifikasi pola preferensi pengguna secara menyeluruh [7], [39].

Dalam berbagai eksperimen sistem rekomendasi yang mengadopsi protokol evaluasi leave-one-out, metrik Recall@K sering kali memiliki karakteristik yang serupa atau identik dengan Hit Ratio (HR) [10], [40], [47]. Pada skema evaluasi ini, hanya terdapat satu item relevan (item uji) untuk setiap pengguna yang harus diprediksi oleh sistem [40]. Sebuah "hit" terjadi apabila item uji tunggal tersebut berhasil menempati posisi di dalam daftar Top−K yang dihasilkan model [40], [47]. Karena jumlah total item relevan pada penyebut rumus recall bernilai satu, maka nilai persentase recall secara matematis akan sama dengan nilai hit tersebut, yang menjadikan HR sebagai metrik yang bersifat recall-based [10], [47].

Penggunaan Recall@K sebagai parameter keberhasilan sangat krusial karena membantu peneliti memahami sejauh mana sistem mampu menembus pola preferensi pengguna untuk menemukan item-item yang tersembunyi [38]. Meskipun demikian, peningkatan nilai recall sering kali diiringi dengan tantangan berupa risiko penurunan presisi, sehingga metrik ini umumnya dianalisis secara bersamaan dengan Precision@K dan F1-score untuk mendapatkan gambaran performa model yang lebih seimbang [32], [38], [7]. Dengan mengoptimalkan Recall@K, sistem diharapkan dapat memberikan rujukan yang lebih personal dan komprehensif, sehingga membantu pengguna menavigasi informasi di tengah fenomena kelebihan data yang masif [9], [7], [38]. Rumus Recall@K dapat dinyatakan sebagai berikut:

$$Recall@K = \frac{|R_u \cap \hat{R}_u^k|}{R_u} \quad \text{.....(2.14)}$$

Dimana:

R_u = himpunan item yang relevan untuk user u

$\hat{R}_u^k$ = himpunan Top-K item yang direkomendasikan sistem

$|R_u \cap \hat{R}_u^k|$= jumlah item relevan yang berhasil muncul di Top-K

Metrik ini sangat berguna untuk menunjukkan kemampuan sistem dalam menemukan sebanyak mungkin item relevan dalam jendela rekomendasi yang terbatas bagi pengguna [46].

2.8 NDCG@K

Normalized Discounted Cumulative Gain (NDCG@K) adalah metrik evaluasi yang digunakan untuk mengukur kualitas urutan atau peringkat dari daftar rekomendasi yang dihasilkan oleh sistem [37], [7], [48]. Berbeda dengan metrik seperti Recall atau Hit Ratio yang hanya memeriksa keberadaan item relevan tanpa memperhatikan urutannya, NDCG bersifat position-sensitive [10], [11], [40]. Artinya, sistem akan memberikan skor yang lebih tinggi jika item yang sangat relevan diletakkan pada posisi yang lebih atas dalam daftar Top-K [37], [7]. Metrik ini sangat krusial dalam sistem pemeringkatan karena mencerminkan perilaku nyata pengguna yang cenderung lebih memperhatikan item-item di posisi puncak dibandingkan posisi di bawahnya [39].

Mekanisme perhitungan NDCG didasarkan pada konsep *Discounted Cumulative Gain* (DCG), yang menjumlahkan nilai relevansi item dengan menerapkan penalti berupa pembagi logaritma berdasarkan posisi item tersebut [37], [39], [46]. Penggunaan fungsi logaritma ini bertujuan untuk secara sistematis mengurangi bobot nilai relevansi seiring dengan semakin rendahnya posisi item dalam daftar peringkat [37], [39]. Dengan demikian, keberhasilan model dalam menempatkan item yang benar-benar diminati pengguna pada peringkat pertama akan memberikan kontribusi skor yang jauh lebih besar dibandingkan jika item tersebut muncul pada peringkat yang lebih rendah, seperti peringkat kesepuluh [7], [40], [10].

Penurunan NDCG@K dimulai dari konsep *Discounted Cumulative Gain* (DCG), yang dirumuskan sebagai berikut [46], [12]:

$$DCG@K = \sum_{i=1}^{k} \frac{rel_i}{\log(i+1)} \quad \text{.....(2.15)}$$

Dimana:

i = urutan atau posisi item dalam daftar rekomendasi (dimulai dari 1 hingga K)

log(i+1) = Faktor pendiskon (*discounting factor*) yang berfungsi untuk mengurangi bobot skor item jika letaknya semakin jauh di bawah dalam daftar rekomendasi.

K = Jumlah total item yang direkomendasikan untuk dievaluasi.

Agar skor evaluasi dapat dibandingkan secara adil antar pengguna yang memiliki jumlah item relevan berbeda, nilai DCG perlu dinormalisasi menggunakan *Ideal Discounted*

Cumulative Gain (IDCG) [39], [46], [10], [7]. IDCG merepresentasikan skor DCG maksimum yang mungkin dicapai jika seluruh item relevan disusun dalam urutan terbaik atau ideal [37], [39], [7]. Hasil akhir metrik NDCG diperoleh dari rasio antara DCG dan IDCG, yang menghasilkan nilai dalam rentang antara 0 hingga 1 [46], [10], [7], [39]. Standarisasi ini memastikan bahwa performa model dapat diukur secara konsisten di seluruh populasi pengguna dalam dataset meskipun terdapat perbedaan jumlah *ground truth* [39], [7]. Rumus IDCG@K dapat dinyatakan sebagai berikut:

$$IDCG@K = \sum_{i=1}^{\min(k,|R_u|)} \frac{1}{\log(i+1)} \quad \text{.....(2.16)}$$

Dimana:

Ru = Jumlah item relevan yang dimiliki user dalam data uji.

Min(k,|Ru|) = Batas maksimal perhitungan ideal, diambil darimana yang lebih kecil antara jumlah rekomendasi (K) atau jumlah item relevan asli.

log(i+1) = Faktor pendiskon (*discounting factor*) yang berfungsi untuk mengurangi bobot skor item jika letaknya semakin jauh di bawah dalam daftar rekomendasi.

Setelah nilai IDCG didapati, nilai NDCG@K didapatkan dengan membagi skor aktual (DCG) dengan skor ideal (IDCG) [46], [39].

$$NDCG@K = \frac{DCG@K}{IDCG@K} \quad \text{.....(2.17)}$$

Dalam penelitian sistem rekomendasi, NDCG@K dianggap sebagai metrik yang lebih komprehensif untuk mengevaluasi efektivitas pemeringkatan dan kegunaan daftar rekomendasi bagi pengguna secara praktis [11], [7], [42]. Nilai NDCG yang mendekati 1 menunjukkan bahwa model telah berhasil menangkap pola preferensi pengguna dengan sangat baik dan mampu menyajikannya dalam urutan yang optimal [48], [10], [46]. Optimalisasi terhadap metrik ini membuktikan bahwa sistem tidak hanya mampu menemukan item yang relevan, tetapi juga memiliki kemampuan untuk menyusun informasi secara berkualitas di tengah fenomena kelebihan data yang masif [7], [9], [3]. Penggunaan NDCG bersama metrik akurasi lainnya memberikan gambaran yang lebih utuh mengenai keunggulan kompetitif suatu algoritma dalam konteks personalisasi [7].

2.9 Black Box Testing

Black-box testing merupakan metode pengujian perangkat lunak yang dilakukan dengan mengamati fungsi sistem tanpa melihat struktur kode program yang ada di dalamnya. Pengujian ini berfokus pada input yang diberikan kepada sistem dan output yang dihasilkan untuk memastikan bahwa sistem telah berjalan sesuai dengan spesifikasi yang telah

ditentukan. Metode ini digunakan untuk menemukan kesalahan pada fungsi sistem, kesalahan pada antarmuka, kesalahan struktur data, serta kesalahan dalam pemrosesan data [49].

Black-box testing memiliki beberapa teknik pengujian yang dapat digunakan untuk memastikan sistem berjalan dengan baik. Adapun beberapa teknik yang digunakan dalam pengujian *black-box testing* adalah sebagai berikut:

1. *Equivalence Partitioning*

   Equivalence Partitioning merupakan teknik pengujian yang dilakukan dengan membagi data *input* ke dalam beberapa kelompok atau partisi yang memiliki karakteristik yang sama. Setiap kelompok data tersebut kemudian diuji untuk memastikan bahwa sistem dapat menerima *input* yang valid maupun menolak *input* yang tidak valid sesuai dengan aturan yang telah ditentukan. Dengan menggunakan teknik ini, proses pengujian dapat dilakukan secara lebih efisien karena tidak semua kemungkinan *input* perlu diuji secara satu per satu, melainkan cukup menggunakan perwakilan dari setiap kelompok data [50].

2. *Boundary Value Analysis*

   Boundary Value Analysis merupakan teknik pengujian yang dilakukan dengan menguji nilai batas dari suatu input yang dimasukkan ke dalam sistem. Pengujian dilakukan pada nilai minimum, nilai maksimum, serta nilai yang berada di sekitar batas tersebut. Teknik ini bertujuan untuk mengetahui apakah sistem mampu menangani nilai-nilai batas dengan benar sehingga dapat meminimalisir kesalahan yang terjadi akibat input yang berada di luar batas yang telah ditentukan [51].

3. *Robustness Testing*

   Robustness Testing merupakan teknik pengujian yang digunakan untuk mengetahui sejauh mana sistem mampu menangani berbagai kondisi *input* yang tidak valid atau tidak terduga. Pengujian ini dilakukan dengan memberikan data *input* secara acak atau data yang tidak sesuai dengan format yang telah ditentukan untuk memastikan bahwa sistem tetap dapat berjalan dengan baik tanpa mengalami kegagalan atau kesalahan yang dapat menyebabkan sistem berhenti berfungsi [52].

4. *Cause-Effect Relationship Testing*

   Cause-Effect Relationship Testing merupakan teknik pengujian yang digunakan untuk menganalisis hubungan antara kondisi atau penyebab (cause) dengan hasil atau efek (effect) yang dihasilkan oleh sistem. Teknik ini dilakukan dengan mengidentifikasi berbagai kondisi input yang dapat mempengaruhi keluaran sistem, kemudian

menentukan hubungan logis antara kondisi tersebut dengan output yang dihasilkan sehingga dapat diketahui apakah sistem telah memberikan respon yang sesuai [53].

5. *State Transition Testing*

   State Transition Testing merupakan teknik pengujian yang dilakukan dengan mengamati perubahan kondisi atau status pada suatu sistem ketika menerima *input* tertentu. Teknik ini digunakan untuk memastikan bahwa setiap perubahan status yang terjadi pada sistem telah berjalan dengan benar sesuai dengan alur yang telah dirancang. Pengujian ini biasanya digunakan pada sistem yang memiliki proses perpindahan status seperti proses *login*, proses transaksi, maupun perubahan kondisi pada antarmuka pengguna [54].

Selain teknik pengujian, pelaksanaan *Black box Testing* juga memerlukan langkah-langkah yang sistematis agar hasil pengujian dapat berjalan secara optimal [53]. Adapun langkah-langkah dalam melakukan pengujian *Black Box Testing* adalah sebagai berikut:

1. Perancangan Skenario Pengujian

   Tahap ini dilakukan dengan merancang skenario atau perintah pada *form* yang akan dilakukan pengujian. Skenario pengujian dibuat untuk menentukan bagaimana proses pengujian akan dilakukan pada sistem.

2. Pembuatan *Test Case*

   Pada tahap ini dilakukan pembuatan *test case* dengan mencatat bagian atau kolom yang akan diuji pada *form* penginputan data.

3. Melakukan Pengujian *Test Case*

   Tahap ini dilakukan dengan menjalankan test case yang telah dibuat sebelumnya dengan melakukan perintah sesuai dengan skenario pengujian.

4. Pencatatan Hasil Pengujian

   Pada tahap ini seluruh hasil pengujian dicatat dan disusun dalam bentuk laporan untuk mengetahui apakah sistem telah berjalan sesuai dengan yang diharapkan atau masih terdapat kesalahan pada sistem.

2.10 Penelitian Terdahulu (Sistem Rekomendasi Resep Makanan)

Bagian penelitian terdahulu ini merangkum penelitian mengenai sistem rekomendasi resep makanan dari berbagai sumber penelitian. Terdapat dua tabel yang dimana tabel pertama membahas pendekatan, model yang diapkai, dataset yang dipakai, metrik evaluasi yang digunakan, dan kesimpulan yang bisa diambil dari penelitian tersebut. Tabel pertama berfungsi untuk mempermudah analisa *insight* dan *gap* yang ingin diisi oleh penelitian

kami. Tabel kedua membahas salah satu penelitian yang melakukan komparasi beberapa model tertentu dalam melakukan sistem rekomendasi terhadap data resep makanan. Kami memaparkan tabel ke-2 karena model-model didalamnya melatih menggunakan dataset yang sama dengan yang kami pakai, dan terdapat metrik evaluasi yang sama yaitu NDCG@10. Dengan begitu, tabel ke-2 berfungsi sebagai salah satu tolak ukur/baseline selain popularity baseline, agar hasil sistem rekomendasi model EASE-R nantinya bisa dinilai atau diuji lebih objektif.

Berikut tabel pertama penelitian terdahulu

Tabel 2. 5 Tabel Penelitian Terdahulu

Penelitian	Pendekatan	Model	Objek	Metrik Evaluasi	Kesimpulan
Mulyawan & Lestari (2019)	Content Based Filtering	Cosine Similarity berdasarkan bahan baku resep makanan	TheMealDB (Resep masakan dunia	Indeks Kekuatan Skala Likert	Sistem rekomendasi dievaluasi menggunakan tingkat kekuatan kesetujuan responden mengenai kesesuaian resep yang ditampilkan dengan apa yang mereka inginkan adalah, nilai tersebut mencapai 71% yang menunjukkan pengguna sudah cukup puas. Namun metrik evaluasi belum cukup objektif, penelitian belum menerapkan collaborative filtering untuk rekomendasi yang lebih personal, dan beberapa fitur seperti penyimpanan resep tidak ada pada sistem [6].
Sari et al. (2024)	Content Based Filtering	TF-IDF & Cosine Similarity	masakapahariini.com (Resep masakan)	RMSE	Metode content-based filtering menggunakan TF-IDF dan cosine similarity menghasilkan rekomendasi yang cukup akurat berdasarkan nilai metrik RMSE yang mencapai 0.356359182. Namun penelitian ini hanya menggunakan sampel data resep makanan berjumlah 30

					items, serta meskipun penelitian ini menunjukkan performa yang baik berdasarkan nilai RMSE, metrik tersebut pada dasarnya hanya mengukur tingkat kesalahan prediksi rating dan belum sepenuhnya merepresentasikan kualitas sistem rekomendasi dalam skenario top-N recommendation, dan sistem rekomendasi masih berdasarkan kemiripan query dan konten resep makanan, belum bersifat personal yang didasari dari preferensi pengguna [5].
Widiantari et al. (2025)	Hybrid Recommendation System (Content Based filtering & Collaborative Filtering)	NCF (Collaborative), Cosine Similarity (Content-based)	Hasil scraping dari goodfood.com	MAP@k, NDCG@k	Berdasarkan hasil evaluasi, performa model mengalami peningkatan yang signifikan setelah lebih dari 50 sesi pelatihan, ditunjukkan dengan penurunan nilai loss dari 0,6728 menjadi 0,3702 serta peningkatan metrik NDCG@10 dari 0,7371 menjadi 0,8813 dan MAP@10 dari 0,6468 menjadi 0,8402. Hasil tersebut menunjukkan bahwa model tidak hanya mampu mengidentifikasi resep yang relevan, tetapi juga mampu menempatkan rekomendasi pada posisi teratas secara efektif, sehingga kualitas ranking rekomendasi menjadi lebih baik. Penggunaan metrik berbasis ranking seperti NDCG@10 dan MAP@10 dinilai lebih

					representatif dalam mengevaluasi sistem rekomendasi dibandingkan metrik error prediksi seperti RMSE, khususnya pada pendekatan recommendation berbasis implicit feedback, karena metrik tersebut lebih mampu menggambarkan keberhasilan sistem dalam menyajikan item relevan pada daftar rekomendasi pengguna. Namun demikian, model berbasis Neural Collaborative Filtering (NCF) memiliki kelemahan berupa kebutuhan komputasi yang relatif tinggi akibat penggunaan arsitektur deep learning dan proses tuning hyperparameter yang kompleks, sehingga waktu pelatihan menjadi lebih lama dibandingkan metode yang lebih sederhana seperti TF-IDF. Kondisi ini dapat menjadi tantangan dalam implementasi pada aplikasi web atau sistem real-time karena membutuhkan sumber daya komputasi yang lebih besar agar proses pelatihan maupun inferensi dapat berjalan secara efisien [3].

Berikut tabel ke-2 Penelitian terdahulu

Tabel 2. 6 Tabel Penelitian Terdahulu Yang Menggunakan Dataset Yang Sama

Model	Dataset	HR@10	NDCG@10	Deskripsi Model

GRU4Rec	FoodSysRecV1	0.0485	0.0242	GRU4Rec merupakan model rekomendasi *Recurrent Neural* Network (RNN). GRU4Rec berbasis sesi yang menggunakan unit saraf berulang (*Gated Recurrent Units / GRU*) untuk menangkap ketergantungan sekuensial (urutan) dalam interaksi pengguna. Model ini mengandalkan urutan ID item tanpa akses langsung ke konten semantik atau nutrisi dari makanan [55].
MoRec	FoodSysRecV1	0.0605	0.0325	MoRec merupakan model rekomendasi sekuensial berbasis modalitas. MoRec. Model ini menggantikan lapisan *embedding* ID pada model SASRec dengan *encoder* modalitas yang dipelajari secara *end-to-end* (seperti BERT untuk teks atau ResNet untuk gambar). Model ini digunakan untuk memvalidasi efektivitas representasi berbasis modalitas murni dalam mengatasi masalah *cold-start* dan meningkatkan pemahaman semantik dibandingkan model berbasis ID murni. Namun identik dengan model deep learning lainnya MoRec memiliki biaya komputasi tinggi akibat pemrosesan fitur konten melalui jaringan saraf dalam, waktu pelatihan yang lama, dan waktu inferensi dan latensi yang tinggi [55].
TALLRec	FoodSysRecV1	0.0598	0.0322	TALLRec merupakan model rekomendasi berbasis LLM. Model ini menggunakan kerangka kerja *instruction-tuning* yang merumuskan tugas

				rekomendasi sebagai masalah generasi berbasis *prompt*. Model ini melakukan *fine-tuning* pada LLaMA menggunakan instruksi riwayat interaksi pengguna. Deng & Tu mencatat bahwa TALLRec cenderung sedikit di bawah performa karena model ini mengandalkan *tuning* instruksi tanpa penyelarasan kolaboratif eksplisit, sehingga terdapat "celah semantik" antara ruang pra-latih dan ruang perilaku pengguna yang spesifik [55].
LlaRA	FoodSysRecV1	0.0655	0.0365	*Large Language and Recommendation Assistant* (LlaRA) merupakan model rekomendasi berbasis LLM. Model ini memiliki kerangka kerja yang mengadaptasi LLM untuk rekomendasi sekuensial menggunakan pendekatan hibrida**.** Model ini berupaya menjembatani kesenjangan antara *embedding* ID tradisional dengan kemampuan tekstual LLM untuk meningkatkan akurasi prediksi urutan interaksi [55].
A-LLMRec	FoodSysRecV1	0.0668	0.0375	*All-round LLM-based Recommender* (A-LLMRec) merupakan model rekomendasi berbasis LLM. A-LLMRec adalah model yang menyelaraskan sinyal kolaboratif dengan ruang semantik LLM. Penyelarasan ini memungkinkan model untuk memanfaatkan pengetahuan dari *collaborative filtering* sekaligus penalaran tekstual

				dari LLM secara bersamaan. Dalam penelitian ini, A-LLMRec diakui sebagai pesaing LLM terkuat, meskipun NutriRec tetap mengunggulinya karena NutriRec memiliki komponen khusus untuk menangkap atribut kesehatan struktural makanan [55].