SISTEM REKOMENDASI PEMILIHAN FILM ANIMASI MENGGUNAKAN METODE *CONTENT BASED FILTERING*

SKRIPSI

Oleh :
NAZHIF MU'AFA ROZIQIIN
NIM. 200605110160

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM
MALANG
2024

SISTEM REKOMENDASI PEMILIHAN FILM ANIMASI MENGGUNAKAN METODE *CONTENT BASED FILTERING*

SKRIPSI

Diajukan Kepada:
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang
Untuk memenuhi Salah Satu Persyaratan Dalam
Memperoleh Gelar Sarjana Komputer (S.Kom)

Oleh :
NAZHIF MU'AFA ROZIQIIN
NIM. 200605110160

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS SAINS DAN TEKNOLOGI
UNIVERSITAS ISLAM NEGERI MAULANA MALIK IBRAHIM MALANG
2024

HALAMAN PERSETUJUAN

SISTEM REKOMENDASI PEMILIHAN FILM ANIMASI MENGGUNAKAN METODE *CONTENT BASED FILTERING*

SKRIPSI

Oleh :
NAZHIIF MU'AFA ROZIQIIN
NIM. 200605110160

Telah Diperiksa dan Disetujui untuk Diuji:
Tanggal: 30 April 2024

Pembimbing I,	Pembimbing II,
Dr. Muhammad Faisal, M.T	Hani Nurhayati, M.T
NIP. 19740510 200501 1 007	NIP. 19780625 200801 2 006

Mengetahui,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan, M.MT, IPM
NIP. 19771020 200912 1 001

HALAMAN PENGESAHAN

SISTEM REKOMENDASI PEMILIHAN FILM ANIMASI MENGGUNAKAN METODE *CONTENT BASED FILTERING*

SKRIPSI

Oleh :
NAZHIF MU'AFA ROZIQIIN
NIM. 200605110160

Telah Dipertahankan di Depan Dewan Penguji Skripsi
dan Dinyatakan Diterima Sebagai Salah Satu Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer (S.Kom)
Tanggal: 21 Mei 2024

Susunan Dewan Penguji

Ketua Penguji	: Dr. Ririen Kusumawati, S.Si., M.Kom NIP. 19720309 200501 2 002	( )
Anggota Penguji I	: Dr. Yunifa Miftachul Arif, M.T NIP. 19830616 201101 1 004	( )
Anggota Penguji II	: Dr. Muhammad Faisal, M.T NIP. 19740510 200501 1 007	( )
Anggota Penguji III	: Hani Nurhayati, M.T NIP. 19780625 200801 2 006	( )

Mengetahui dan Mengesahkan,
Ketua Program Studi Teknik Informatika
Fakultas Sains dan Teknologi
Universitas Islam Negeri Maulana Malik Ibrahim Malang

Dr. Fachrul Kurniawan, M.MT, IPM
NIP. 19771020 200912 1 001

PERNYATAAN KEASLIAN TULISAN

Saya yang bertanda tangan di bawah ini:

Nama	: Nazhif Mu'afa Roziqiin
NIM	: 200605110160
Fakultas / Program Studi	: Sains dan Teknologi / Teknik Informatika
Judul Skripsi	: Sistem Rekomendasi Pemilihan Film Animasi Menggunakan Metode *Content Based Filtering*

Menyatakan dengan sebenarnya bahwa Skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri, bukan merupakan pengambil alihan data, tulisan, atau pikiran orang lain yang saya akui sebagai hasil tulisan atau pikiran saya sendiri, kecuali dengan mencantumkan sumber cuplikan pada daftar pustaka.

Apabila dikemudian hari terbukti atau dapat dibuktikan skripsi ini merupakan hasil jiplakan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Malang, 30 April 2024
Yang membuat pernyataan,

Nazhif Mu'afa Roziqiin
NIM.200605110160

MOTTO

وَلَا تَهِنُواْ وَلَا تَحۡزَنُواْ وَأَنتُمُ ٱلۡأَعۡلَوۡنَ إِن كُنتُم مُّؤۡمِنِينَ ١٣٩

"Dan janganlah kamu (merasa) lemah, dan jangan (pula) bersedih hati, sebab kamu paling tinggi (derajatnya), jika kamu orang beriman."

— Surat Ali Imran 139

,,

"لا تلومن الا نفسك، فكر في كيف يمكنك تحسين نفسك وتطوير سلوكياتك "

“Janganlah engkau menyalahkan siapa pun kecuali dirimu sendiri, pikirkan bagaimana engkau mengubah dan mengembangkan diri menjadi lebih baik”

,,

HALAMAN PERSEMBAHAN

Alhamdulillah, dengan Rahmat Allah yang Maha Pengasih Lagi Maha Penyayang.

Kami persembahkan karya ini terutama untuk Bapak Arif Afandi dan Ibu Indriasari sebagai tanda hormat dan terima kasih yang tidak terhingga atas doa yang selalu terpanjat. Kepada kakekku, (.alm) Aliudin yang telah memberikan dukungan untuk melanjutkan kuliah. Tak lupa juga kepada adik-adik ku yang baik hati.

Kepada dosen pembimbing I, Dr. Muhammad Faisal, M.T dan Pembimbing II, Hani Nurhayati M,T yang telah memberikan banyak saran dan pelajaran dalam pembuatan skripsi ini.

Kepada seluruh dosen dan jajaran sivitas akademika program studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang yang telah mendampingi dan memberi kelancaran ketika masa studi.

Terakhir, kepada diri sendiri yang telah berjuang dan bertahan.

KATA PENGANTAR

Assalamu'alaikum Warahmatullah Wabarakatuh

Bismillah, puji dan syukur penulis panjatkan kepada Allah *Subhanahu Wa Ta'ala* Tuhan semesta alam, karena dengan seluruh rahmat dan nikmat-Nya penulis dapat menyelesaikan skripsi dengan judul "Sistem Rekomendasi Pemilihan Film Animasi Menggunakan Metode *Content Based Filtering*" dengan baik. Shalawat serta salam selalu tercurahkan kepada Nabi Agung Muhammad *shallallahu 'alaihi wasallam* yang telah membimbing manusia ke jalan yang benar.

Alhamdulillah pada akhirnya skripsi yang digunakan untuk menyelesaikan jenjang sarjana ini dapat diselesaikan dengan sebaik mungkin. Selanjutnya penulis mengapresiasi diri sendiri karena sudah bisa berjuang hingga detik ini. Tak lupa penulis ucapkan terima kasih kepada pihak yang turut memberikan bantuan baik secara moril dan materil. Atas segala bantuannya, penulis mengucapkan terima kasih sebesar-besarnya kepada :

1. Prof. Dr. H.M. Zainuddin, MA selaku Rektor Universitas Islam Negeri Maulana Malik Ibrahim Malang.
2. Dr. Sri Harini, M.Si selaku Dekan Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang
3. Dr. Fachrul Kurniawan ST., M.MT., IPM selaku Ketua Program Studi Teknik Informatika Universitas Islam Negeri Maulana Malik Ibrahim Malang.

4. Dr. Muhammad Faisal, M.T selaku pembimbing I yang telah meluangkan waktu untuk membimbing, mengarahkan, dan memberikan banyak motivasi ketika proses pengerjaan skripsi sehingga bisa tuntas.
5. Hani Nurhayati, M.T selaku pembimbing II yang telah memberikan arahan dan dengan cermat meluruskan kesalahan skripsi ini.
6. Segenap Dosen Teknik Informatika yang telah memberikan ilmu yang telah dimiliki kepada penulis selama kurang lebih 4 tahun.
7. Kedua orang tua tercinta, Pak Arif Afandi dan Ibu Indriasari yang telah memberikan banyak dukungan moral.
8. Semua sivitas akademika yang telah membantu proses administrasi.
9. Semua teman seperjuangan terutama Integer, Befakkariem, dan Ghuroba yang selalu memberikan semangat untuk terus berjuang
10. Semua pihak lain yang tidak bisa disebutkan satu per-satu, tanpa bantuan kalian, skripsi ini tidak akan selengkap ini.

Penulis menyadari masih banyak yang kurang sempurna pada skripsi ini, oleh karena itu kritik dan saran yang membangun sangat diharapkan untuk kemajuan penelitian selanjutnya. Terakhir, penulis berharap penelitian ini dapat memberikan manfaat bagi pembaca dan peneliti selanjutnya. Aamiin

Assalamu'alaikum Warahmatullah Wabarakatuh

Malang, 30 April 2024

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

ABSTRAK

Roziqiin, Nazhif Mu'afa. 2024. **Sistem Rekomendasi Pemilihan Film Animasi Menggunakan Metode *Content Based Filtering*.** Skripsi. Program Study Teknik Informatika Fakultas Sains dan Teknologi Universitas Islam Negeri Maulana Malik Ibrahim Malang. Pembimbing: (I) Dr. Muhammad Faisal, M.T (II) Hani Nurhayati, M.T.

Kata kunci: Sistem Rekomendasi, *Content Based Filtering*, Animasi

Sistem rekomendasi merupakan sebuah platform yang memberikan saran kepada pengguna berdasarkan data yang telah ada. Sistem rekomendasi pemilihan film animasi bertujuan untuk memberikan rekomendasi film animasi berdasarkan preferensi pengguna. Tujuan penelitian ini adalah mengembangkan metode *content based filtering* yang dikolaborasikan dengan *singular value decomposition* dan *cosine similarity*. Data yang digunakan dalam penelitian ini adalah data "52000 *detail animation movies*" yang diambil dari *The Movies Databases*. Uji coba dilakukan dengan menginputkan beberapa *query* berupa judul film, kemudian dicari nilai *Mean Average Precision* (MAP) berdasarkan relevansinya. Relevansi film animasi diambil dari kesamaan genre pada film animasi. Hasil uji coba menunjukkan metode *content-based filtering* dengan menggunakan *singular value decomposition* mendapatkan nilai rata-rata MAP 0.865. Hasil rekomendasi juga menunjukkan metode *content based filtering* dengan algoritma SVD memberikan hasil yang lebih bervariasi daripada metode *content based filtering* karena SVD juga memperhatikan hubungan *semantic* antar kata.

ABSTRACT

Roziqiin, Nazhif Mu'afa. 2024. **Recommendation System for Animation Movies Using *Content Based Filtering*.** Undergraduate Thesis. Department of Informatics Engineering Faculty of Science and Technology Islamic State University of Maulana Malik Ibrahim Malang. Supervisors: (I) Dr. Muhammad Faisal, M.T (II) Hani Nurhayati, M.T.

Keywords: Recommendation System, *Content Based Filtering*, Animation.

Recommendation system is a platform that provides advice to users based on existing data. The animation movie selection recommendation system aims to provide animation movie recommendations based on user preferences. The purpose of this research is to develop a content-based filtering method collaborated with singular value decomposition and cosine similarity. The data used in this research is "52000 detailed animation movies" data taken from The Movies Databases. Tests were conducted by inputting several queries in the form of movie titles, then looking for MAP on their relevance. The relevance of animated films is taken from the similarity of genres in animated films. The test results show that the content-based filtering method using singular value decomposition gets obtained an average MAP value of 0.865. The recommendation results also show that the content based filtering method with the SVD algorithm provides more varied results than the content based filtering method because SVD also pays attention to the semantic relationships between words.

البحث مستخلص

رازقين، نظيف معافا. ٢٠٢٤. نظام التوصية باختيار أفلام الرسوم المتحركة باستخدام طريقة **Content Based Filtering**. البحث الجامعي. قسم هندسة المعلوماتية، كلية العلوم والتكنولوجيا، الجامعة الإسلامية الحكومية مولانا مالك إبراهيم مالانج. المشرف : (١) الدكتور محمد فيصل الماجستير (٢) هاني نورحياتي الماجستير

الكلمات الرئيسية: نظام التوصية، Content Based Filtering، الرسوم المتحركة

نظام التوصية هو عبارة عن منصة تقدم المشورة للمستخدمين بناءً على البيانات الموجودة. يهدف نظام التوصية باختيار أفلام الرسوم المتحركة إلى تقديم توصيات لأفلام الرسوم المتحركة بناءً على تفضيلات المستخدم. والغرض من هذا البحث هو تطوير طريقة Content Based Filtering مع تحلل Singular Value Decomposition و Cosine Similarity. والبيانات المستخدمة في هذا البحث هي بيانات "52000 فيلم رسوم متحركة مفصلة" مأخوذة من قواعد بيانات الأفلام. تم إجراء الاختبارات عن طريق إدخال عدة استعلامات عشوائية في شكل عناوين أفلام، ثم البحث عن MAP على مدى ملاءمتها. أهمية أفلام الرسوم المتحركة مأخوذة من الأنواع في أفلام الرسوم المتحركة. تُظهر نتائج الاختبار أن Content Based Filtering المستندة إلى المحتوى باستخدام SVD واحد لها متوسط قيمة MAP يبلغ 0.865. تظهر نتائج التوصية أيضًا أن طريقة التصفية المعتمدة على المحتوى باستخدام خوارزمية SVD توفر نتائج أكثر تنوعًا من طريقة التصفية المعتمدة على المحتوى لأن SVD تهتم أيضًا بالعلاقات الدلالية بين الكلمات

BAB I

PENDAHULUAN

1.1 Latar Belakang

Akses ke informasi mengenai berbagai jenis film animasi telah tersebar di internet pada era digital, memungkinkan seseorang untuk dengan mudah mencari film yang diinginkannya. Berbagai negara maju telah memperluas industri film animasi baik di dalam negeri maupun di mancanegara. Contohnya Cina memiliki *Animation Park Tianjin City*, Jepang memiliki berbagai studio animenya, Amerika memiliki *Walt Disney, Cartoon Network, Dream Work,* dan *Illumination Studio,* sementara Korea Selatan memiliki animasi seperti *Pororo si Penguin Kecil* yang diproduksi oleh *Iconix Entertainment Co*. Indonesia juga mulai produktif dalam memproduksi animasi. Berdasarkan riset yang dilakukan oleh Asosiasi Industri Animasi Indonesia (AINAKI) pada tahun 2020 terhadap 120 studio animasi di Indonesia, nilai jasa yang dihasilkan diperkirakan mencapai 0,6-0,8 triliun rupiah. Angka ini masih hanya seperlima dari total jumlah studio animasi di Indonesia. Dalam kurun waktu 2015-2019, industri animasi di Indonesia telah tumbuh sebesar 153% dengan kenaikan rata-rata 26%. Pertumbuhan industri animasi di Indonesia ini diperkirakan akan terus berkembang seiring dengan meningkatnya permintaan di kanal-kanal populer seperti *Youtube*, *Instagram, Netflix*, dan sebagainya (AINAKI, 2020).

Meningkat secara drastisnya tingkat kepemirsaan TV pada penonton anak, yang mencapai *rating* tertinggi sebesar 16.2% (AINAKI, 2020). Perhatian orang

tua dalam mendampingi anak saat menonton film animasi menjadi semakin penting. Namun, masalah lain yang muncul adalah kebingungan penonton dalam mencari film animasi yang serupa dengan yang disukai. Dengan banyaknya judul film animasi, mencari judul yang cocok bisa menjadi tantangan, karena memerlukan waktu untuk menelusuri satu per satu informasi dari setiap film. Oleh karena itu, dibutuhkan sistem rekomendasi yang mampu memberikan rekomendasi dan informasi yang sesuai dengan preferensi pengguna. Sistem ini sering disebut dengan sistem rekomendasi. Sebuah sistem rekomendasi di desain dapat memudahkan pengguna dalam mencari suatu data yang mungkin sesuai dengan yang disukai pengguna secara cepat dan mengurangi informasi yang terlalu banyak. Beberapa metode populer pada sistem rekomendasi adalah *content based filtering*, *collaborative filtering*, dan *hybrid content based filtering* adalah rekomendasi dengan mempelajari profil pengguna, deskripsi produk atau perspektif-perspektif lain dari pengguna terhadap suatu item. Metode ini banyak digunakan untuk merekomendasikan berita, artikel, dan situs *web* (Ikhsani Suwandy Futri & Nuraini Siti Fathonah, 2023).

Memberikan rekomendasi kepada orang lain merupakan salah satu ajaran Islam yang sangat dianjurkan. Hal ini dijelaskan dalam Q.S Al-Anbiya ayat 7 :

وَمَآ اَرْسَلْنَا قَبْلَكَ اِلَّا رِجَالًا نُّوْحِيْٓ اِلَيْهِمْ فَسْـَٔلُوْٓا اَهْلَ الذِّكْرِ اِنْ كُنْتُمْ لَا تَعْلَمُوْنَ

“Kami tidak mengutus sebelum engkau (Nabi Muhammad) melainkan beberapa orang laki-laki yang Kami beri wahyu kepada mereka. Maka, bertanyalah kepada orang yang berilmu jika kamu tidak mengetahui” (Al-Anbiyā' [21]:7).

Dalam Tafsir Ibnu Katsir dijelaskan, Allah *Subhanahu Wa Ta'ala* memerintahkan manusia meminta tolong atau bertanya kepada orang yang lebih berilmu, jika tidak mengetahui akan sesuatu. Dijelaskan bahwa yang dimaksud dengan *ahlu dzikrí* di sini ialah seorang manusia yang lebih memiliki ilmu, yang memiliki kepahaman tentang suatu hal. Hal ini bukan termasuk suatu kesombongan, ini adalah nikmat yang Allah *Subhanahu Wa Ta'ala* berikan kepada seluruh makhluknya untuk saling menasehati dan mengingatkan kepada kebaikan. Hal ini serupa juga dengan salah satu hadis Nabi Muhammad *shalallahu 'alaihi wasallam* yang memerintahkan kepada umatnya untuk menyampaikan ilmu walaupun sedikit yang berbunyi :

عن عبد الله ابن عمرو ان النبي صلى الله عليه و سلم قال : بلّغوا عنّي ولو آيةً وحدّثوا عن بني إسرائيلَ ولا حرجَ ومن كذبَ عليَّ متعمّدًا فليتبوَّأ مقعدَهُ من النَّارِ

"*Dari Abdullah bin 'Amr (dia berkata) bahwa Nabi shalallahu 'alaihi wasallam telah bersabda: Sampaikanlah dariku walaupun hanya satu ayat, dan ceritakanlah dari Bani Israil, dan tidak ada dosa, barang siapa berdusta atas namaku secara sengaja, maka hendaklah dia menempati tempat duduknya dari neraka." (HR. Bukhari No. 3461)*

Hadis ini menjelaskan bagaimana Rasulullah *shalallahu 'alaihi wasallam* memerintahkan untuk aktif dalam mengajarkan ilmu yang telah diketahui ke orang lain walaupun itu hanya sedikit. Sebagai tanggung jawab orang muslim untuk selalu berbagi ilmu dan bertanya kepada orang yang lebih berilmu (Syamil, 2023).

Dalam penelitian ini, penulis menggunakan metode *Content Based Filtering* (CBF) dengan menggunakan *Term Frequency-Inverse Document* Frequency (TF-IDF) untuk pembobotan kata, dekomposisi matriks dengan *Singular Value*

Decompostion (SVD), dan *cosine similarity* untuk menghitung kesamaan dokumen. Dalam metode *content based filtering*, sistem akan mencari film animasi yang memiliki kesamaan dengan film lainnya berdasarkan profil film. Dengan demikian, diharapkan sistem rekomendasi dapat memberikan rekomendasi yang akurat kepada pengguna.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan, rumusan masalah dari penelitian ini adalah bagaimana kinerja metode *Content Based Filtering* dengan menggunakan algoritma *Singular Value Decomposition* dalam sistem rekomendasi pemilihan film animasi?

1.3 Tujuan Penelitian

Tujuan dari penelitian ini adalah untuk mengetahui kinerja metode *Content Based Filtering* jika menggunakan algoritma *Singular Value Decomposition* pada sistem rekomendasi pemilihan film animasi.

1.4 Batasan Masalah

1. Data diambil diambil dari *The Movie Database* (TMDB) hingga tahun 2023.
2. Menggunakan *content-based filtering* dengan menggunakan *term frequency – inverse document frequent* (TF-IDF), *singular value decomposition* (SVD) untuk melakukan ekstraksi fitur, dan *cosine similarity*
3. Bahasa pemrograman menggunakan Python dan *framework* Flask.

1.5 Manfaat Penelitian

Penulis berharap sistem rekomendasi dengan menggunakan *content based filtering* ini dapat memberikan rekomendasi yang berguna bagi penonton film terutama yang bergenre animasi sehingga memudahkan pengguna menemukan animasi yang cocok dengan kesukaannya berdasarkan profil film.

BAB II

STUDI PUSTAKA

2.1 Kajian Empiris

Penelitian Wibowo *et. al.* (2022) memfokuskan pada pengembangan sistem rekomendasi film yang menggabungkan metode *hybrid collaborative filtering* dan *content-based filtering*. *Hybrid collaborative filtering* pada penelitian tersebut lebih berorientasi pada penggunaan *item-based collaborative filtering* dengan mengukur kemiripan antar item menggunakan korelasi Pearson. Sementara itu, *content based filtering* memanfaatkan profil pengguna yang mencerminkan minat pengguna terhadap konten dalam item, serta profil item yang mencerminkan relevansi konten terhadap item tersebut. Pembangunan matriks profil dilakukan dengan menggunakan metode TF-IDF pada *dataset* MovieLens yang telah diproses sebelumnya. Hasil penelitian menunjukkan bahwa pendekatan *hybrid collaborative filtering-content based filtering* memberikan rekomendasi film yang lebih baik daripada menggunakan kedua metode secara terpisah.

Fitrianti *et. al.* (2020) dalam penelitiannya tentang pembuatan sistem rekomendasi film menggunakan pendekatan *content based filtering*. Penelitian ini menggunakan algoritma *k-nearest neighbors* (KNN) untuk melakukan klasifikasi terhadap data *training* dan *testing*. Data film diambil dari *Kaggle* memiliki atribut: *title*, *genres*, *plot*, *popularity*, poster, *length*, dan *rate*. Hasilnya sistem rekomendasi yang menggunakan dua variabel (*genres* dan *rating*) ini menghasilkan nilai presisi 45% dan nilai *recall* 60%.

Muhadzdzib Ramadhan dan Erwin Setiawan (2022) pada penelitiannya membandingkan kinerja kombinasi antara *k-means clustering* dengan *collaborative filtering* dan hanya menggunakan *collaborative filtering*. Data penelitian diambil dari Twitter dan ditambahkan *rating* dari IMDb, Rotten Tomatoes, dan Metacritic. Kemudian diproses dengan menggunakan *text processing*, *polarity*, dan *labeling*. Sistem diuji dengan menggunakan MAE (*Mean Absolute Error*) dengan hasil 0.5029 dan RMSE (*Root Mean Square Error*) dengan hasil 0.6354. Hal ini menunjukkan gabungan *k-means clustering* dan *collaborative filtering* memiliki nilai akurasi yang lebih baik daripada hanya menggunakan *collaborative filtering* saja.

Penelitian Singla *et al.* (2020) tentang aplikasi FLEX, yaitu sistem rekomendasi pemilihan film yang datanya diambil dari IMDB. Penelitian ini menggunakan fitur *plot*, *genre*, *title*, *votes*, *avgRating*, *year*, dan *productionInfo*. Hasil evaluasi penelitian ini dinilai dengan *precision*, dengan nilai k=5 menggunakan tiga data *dummy* dengan sampel berjumlah 500 data. Percobaan *user dummy* 1 mendapatkan nilai *precision* 0,64, untuk *user dummy* 2 mendapatkan nilai 0,65, dan untuk *user dummy* 3 mendapatkan nilai 0,66.

Penelitian oleh Armadhani Hiro dan Agung Toto Wibowo (2023) sistem rekomendasi film berdasarkan sinopsis film menggunakan metode *content based filtering* dengan TF-IDF dan *cosine similarity*. Pengumpulan data dilakukan dengan menggunakan data set dari MovieLens dengan 45.466 film. Data *keyword* didapatkan melalui proses ekstraksi ketika *preprocessing*. Hasil pengujian dengan menggunakan kombinasi antara *genre* dan *keyword* pada *top-N* 20 tanpa

menggunakan preferensi pengguna mendapatkan nilai *recall*, *precission*, dan *F1-Score* masing-masing 0.01, 0.11, dan 0.02. Hasil pengujian menggunakan *genre* dengan preferensi pengguna pada *top-N* 20 mendapatkan nilai *recall*, *precission*, dan *F1-Score* masing-masing 0.15, 0.25, dan 0.15. Hasil pengujian menggunakan *keyword* dengan preferensi pengguna pada *top-N* 20 mendapatkan nilai *recall*, *precission*, dan *F1-Score* masing-masing 0.01, 0.08, dan 0.01. Sedangkan hasil pengujian kombinasi *genre* dan *keyword* dengan menggunakan preferensi pengguna pada *top-N* 20 mendapatkan nilai *recall*, *precission*, dan *F1-Score* masing-masing 0.03, 0.08, dan 0.04.

Penelitian oleh Mu'tashim Billah *et al.* (2021) tentang penerapan *Collaborative Filtering*, PCA, dan *K-Means* dalam sistem rekomendasi film. Metode PCA pada penelitian tersebut digunakan untuk mempercepat proses *clustering*. Kemudian metode yang digunakan untuk mengukur hasil *clustering* adalah *Silhouette Coefficient*. Penelitian ini juga menggunakan metode *Elbow* yang berfungsi untuk mengatasi masalah lemahnya *K-Means Clustering* dalam menentukan jumlah *k* terbaik dari percobaan *n*. Pengujian ini mendapatkan nilai *Mean Reciprocal Rank* (MRR) 0,4453. Perbandingan penelitian terdahulu dapat dilihat pada Tabel 2.1

Tabel 2.1 Tabel perbandingan dengan penelitian terdahulu

No	Peneliti	Objek	Metode	Hasil	Perbandingan Penelitian ini
1.	Arfisko dan Wibowo (2022)	Film	*Collaborative Filtering*, *Hybrid* CF-CBF, *Content Based Filtering*, *Hybrid* CBF-CF	Menggunakan MAP pada sepuluh percobaan pertama didapatkan metode *Collaborative Filtering* mendapatkan skor 0,3054, *Hybrid* CF-CBF mendapatkan skor 0,3159, *Content Based Filtering* 0,2219, dan *Hybrid* CBF-CF mendapat skor 0,2572	1. Penelitian sekarang menggunakan algoritma SVD 2. Peneliti sekarang mengambil data dari TMDb 3. Penelitian sekarang lebih fokus kepada film animasi
2.	Fitrianti *et al.* (2020)	Film	*Content Based Filtering* dan *K-Nearest Neighbors*	Sistem rekomendasi yang menggunakan dua variabel yaitu *genre* dan *rating* mendapatkan nilai presisi 45% dan *recall* 60%	1. Penelitian sekarang menggunakan algoritma SVD dan Cosine Similarity 2. Penelitian sekarang menggunakan empat fitur atau variabel, sedangkan penelitian oleh Fitrianti et al. (2020) hanya menggunakan dua Fitur. 3. Penelitian sekarang lebih fokus kepada film animasi
3.	Muhadzdzib Ramadhan & Setiawan (2022)	Film	*Collaborative Filtering*, *K-Means Clustering*	Gabungan *Collaborative Filtering* dan *K-Means Clustering* mendapatkan hasil rekomendasi yang lebih baik daripada *Collaborative Filtering* saja. Dengan nilai MAE 0,5029 dan RMSE 0,6354	1. Peneliti sekarang menggunakan metode Content Based Filtering dengan SVD 2. Dataset yang digunakan Muhadzdzib Ramadhan & Setiawan (2022) diambil dari Platform Twitter, sedangkan penelitian sekarang dari TMDb 3. Penelitian sekarang lebih fokus kepada film animasi
4.	Singla *et al.* (2020)	Film	*Content Based FIltering*	Dengan *Content Based Filtering* dengan algoritma TF-IDF mampu menghasilkan nilai presisi 0,66 jika menggunakan *similarity threshold* dan 0,47 jika menggunakan *average*	1. Peneliti sekarang menggunakan metode Content Based Filtering dengan SVD 2. Dataset yang digunakan oleh Singla et

No	Peneliti	Objek	Metode	Hasil	Perbandingan Penelitian ini
				rating.	al. (2020) diambil dari IMDb, sedangkan penelitian sekarang dari TMDb 3. Penelitian sekarang lebih fokus kepada film animasi
5.	Hiro Juni Permana & Toto Wibowo (2023)	Film	*Content Based Filtering* dengan TF-IDF dan *Cosine SImilarity*	Pada semua skenario, *precision* cenderung lebih tinggi dibandingkan *recall*, akan tetapi masih pada peringkat yang rendah. Skenario dengan menggunakan preferensi pengguna memberikan peningkatan dibandingkan tanpa preferensi pengguna. Hasil tertinggi didapatkan pada skenario pengujian *genre* dengan menggunakan preferensi pengguna yaitu dengan nilai F1-Score 0.15.	1. Peneliti sekarang menggunakan metode Content Based Filtering dengan SVD 2. Penelitian sekarang lebih fokus kepada film animasi
6.	Mu'tashim Billah *et al.* (2021)	Film	*Collaborative Filtering*, PCA, dan *K-Means*	Pengujian dengan menggunakan MRR dengan 10 *user* mendapatkan nilai rata-rata sebesar 0,4453.	1. Peneliti sekarang menggunakan metode Content Based Filtering dengan SVD 2. Penelitian sekarang lebih fokus kepada film animasi

2.2 Animasi

Animasi merupakan gambar bergerak yang terbentuk dari sekumpulan objek yang disusun atau diatur secara berurutan mengikuti pergerakan alur tertentu pada setiap hitungan waktu yang terjadi. Animasi berasal dari bahasa latin yaitu *anima* yang berarti jiwa, hidup, atau semangat. Animasi juga berasal dari bahasa inggris yaitu *animation* yang berarti kehidupan. Secara definisi umum animasi dapat dikatakan sebagai suatu gambar yang ditampilkan pada tenggang waktu tertentu sehingga tercipta ilusi gambar. Animasi sebenarnya merupakan sebuah rangkaian gambar yang disusun secara berurutan. Objek dalam gambar pada definisi di atas bisa berupa foto, gambar, tulisan, warna, maupun efek spesial (Firmansyah & Kurniawan, 2013).

Animasi merupakan metode untuk memanipulasi gambar-gambar untuk menghasilkan bentuk-bentuk bergerak yang disusun secara berurutan mengikuti lintasan yang telah ditentukan intervalnya. Secara umum animasi dapat diartikan sebagai serangkaian gambar yang ditampilkan dalam waktu tertentu sehingga dihasilkan sebuah ilusi bergerak. Ada dua objek penting dalam animasi yaitu bingkai (*frame*) dan lintasan gerak (*motion path*) (Kurnia, 2020).

Dari beberapa referensi yang telah disebutkan, diambil kesimpulan bahwa animasi merupakan teknik menampilkan gambar-gambar secara berkala sesuai dengan lintasan gerak sehingga dihasilkan sebuah ilusi bergerak.

2.2.1 Jenis-jenis Animasi

Ada beberapa jenis dari animasi :

1. Animasi Tradisional

Merupakan jenis animasi yang paling tua. Pada awalnya para animator membuat animasi dengan menggambar di atas meja dengan cahaya yang menyinari kertas. Kemudian animator melihat *frame-frame* yang telah mereka gambar. Pada perkembangannya, animator mulai menggambar objek di seluloid transparan sehingga animasi tradisional ini disebut juga dengan animasi target. Animator harus menggambar secara individual *frame-frame* yang akan disusun dan biasanya animasi ini memiliki format 2 dimensi. Contohnya adalah *Tarzan*, *Pinocchio*, dan *Little Mermaid* (Kurnia, 2020).

2. Animasi 2D

Merupakan animasi yang objeknya memiliki ukuran panjang(*x-axis*) dan lebar(*y-axis*) saja (Munir, 2012). Animasi tradisional umumnya hanya terbatas dalam format 2D, namun animasi 2D tidak sebatas animasi tradisional. Animasi 2D dapat dibentuk dengan menggunakan gambar vektor. Hal ini memungkinkan pembuatan animasi lebih mudah dikendalikan dibanding dengan animasi berbasis *pixel*. Lebih menarik, dengan menggunakan vektor ini animator bisa menggunakan karakter yang sama tanpa harus menggambar ulang karakter di *setting* yang berbeda. Contohnya adalah *Tom and Jerry*, *SpongeBob Squarepants*, dan *Scooby Doo* (Kurnia, 2020).

3. Animasi 3D

Perkembangan teknologi dan komputer membuat teknik pembuatan animasi berkembang sehingga muncullah animasi 3D. Animasi 3D berbasis dimensi yang mempunyai (*x-axis*), lebar (*y-axis*), dan tinggi (*z-axis*). Pergerakan dari animasi 3D ini hampir mendekati kenyataan asli sehingga animasi terlihat lebih hidup

(Munir, 2012). Animator cukup menggerakkan bagian-bagian tubuh tertentu karakter dengan mengatur posenya. Meskipun terlihat lebih mudah, proses ini sebenarnya membutuhkan proses pengaturan *frame* yang lebih lama (Kurnia, 2020). Contohnya adalah *Kung Fu Panda*, *Up*, *Upin dan Ipin*, dan *Boboiboy*.

4. *Stop Motion Animation*

Jenis animasi ini dikenal juga dengan *Claymation* karena animasi ini dibuat menggunakan *clay* (tanah liat) sebagai objek. Animasi jenis ini pertama kali di perkenalkan oleh Stuart Blakton pada 1906. Tokoh-tokoh pada jenis animasi ini dibuat dengan menggunakan kerangka khusus untuk kerangka tubuhnya. Setelah tokohnya siap, lalu di foto pergerakannya sehingga menjadi *frame*. Kemudian tokoh diubah lagi posturnya dan di foto lagi, mirip seperti animasi tradisional atau *flip book* (Munir, 2012). Jenis animasi ini kurang popular karena sukar untuk membuatnya dan membutuhkan biaya yang lumayan malah. Namun, industri besar seperti *Disney* dan *Pixar* masih sering membuat animasi menggunakan metode ini. Contoh dari animasi *stop motion* adalah *The Sheep Movie*, *The Nightmare Before Christmas*, dan *Kubo and The Two Strings* (Kurnia, 2020).

5. Animasi Jepang

Jenis animasi ini adalah sebutan tersendiri untuk film animasi di Jepang. Animasi yang biasa disebut *Anime* ini tidak kalah popular dengan animasi buatan Eropa lainnya. Animasi ini memiliki bentuk yang berbeda dengan animasi buatan Eropa dan biasanya digambar manual dengan tangan dan sedikit bantuan komputer. Cerita anime sering kali diambil dan dibukukan dalam bentuk komik yang disebut *manga*. Dr. Osamu Tezuka adalah salah satu tokoh legendaris yang

menciptakan *anime Tetsuwan Atom* atau lebih di kenal dengan *Astro Boy* (Munir, 2012). Contoh dari *anime* populer adalah *Naruto, One Piece*, dan *Doraemon*.

6. Jenis Animasi *File* GIF

Graphic Interfaces Format atau GIF merupakan teknik animasi sederhana yang menggunakan prinsip animasi gambar berupa gambar-gambar yang saling dihubungkan. Pada dasarnya animasi jenis ini adalah beberapa gambar yang digabungkan dan dimainkan secara bergantian seperti halnya *slideshow* (Munir, 2012). GIF juga merupakan salah satu format gambar *bitmap* yang mendukung 8 bit per *pixel* di setiap gambarnya. Karena keterbatasan tersebut, format ini kurang mendukung jika digunakan dalam dunia fotografi atau percetakan. Namun, format ini masih bisa digunakan dalam desain grafis. GIF di kompres dengan teknik *lossless compression* yang mana teknik ini memungkinkan melakukan kompres tanpa mengurangi kualitas gambar. Jenis animasi ini banyak di temui di internet dan sering dipakai pada stiker *WhatsApp*.

2.3 Sistem Rekomendasi

Sistem rekomendasi adalah sebuah platform yang memberikan informasi serta saran kepada pengguna berdasarkan data yang telah ada sebelumnya. Tujuannya adalah membantu pengguna dalam membuat keputusan dengan menyajikan item-item yang mungkin bermanfaat. Sistem ini merupakan gabungan dari perangkat lunak dan teknik-teknik yang secara otomatis menyediakan rekomendasi berdasarkan preferensi dan aktivitas pengguna sebelumnya. Biasanya, sistem rekomendasi ditujukan kepada individu yang mungkin tidak

memiliki pengalaman atau pengetahuan yang cukup untuk menilai jumlah besar item alternatif yang tersedia (Ricci *et al.*, 2011).

Rekomendasi berbeda dari prediksi. Untuk membuat prediksi diperlukan informasi tentang item yang jarang dinilai, sementara pada rekomendasi hanya diperlukan sebagian informasi tertentu. Oleh karena itu, sistem rekomendasi dapat mengurangi masalah kebutuhan memori besar dan waktu komputasi. Sebagian data yang telah didapat kemudian dianalisis dan dipelajari menggunakan algoritma *machine learning* seperti *clustering* dan kategorisasi (Aamir & Bhusry, 2015).

Sistem rekomendasi memiliki tiga pendekatan utama. Pertama, sistem rekomendasi berbasis konten (*content based filtering*) yang mempertimbangkan karakteristik item konten. Kedua, sistem rekomendasi kolaboratif (*collaborative filtering*) yang memiliki karakteristik dari lingkungan pengguna, dan terakhir sistem rekomendasi hibrid (*hybrid filtering*) yang menggabungkan dari kedua pendekatan sebelumnya (Hanifah, 2023).

2.3.1 *Content Based Filtering*

Sistem rekomendasi berbasis konten atau *content based filtering* (CBF) merupakan metode yang melakukan penilaian berdasarkan deskripsi pada item agar dapat menghasilkan rekomendasi berdasarkan preferensi profil pengguna dan hubungan antar deskripsi item. Sehingga metode ini hanya dapat memilih item dengan konten yang hampir sama dalam rekomendasi (Ikhsani Suwandy Futri & Nuraini Siti Fathonah, 2023).

Sistem rekomendasi *content based filtering* membandingkan profil item pengguna dengan karakteristik item saat ini dan berusaha merekomendasikan item serupa yang kemungkinan disukai oleh pengguna. Profil pengguna terdiri dari kata kunci yang berbeda, sehingga teknik mencocokkan kata kunci dari profil pengguna dengan peringkat yang tinggi. Untuk membuat profil pengguna diperlukan data tentang preferensi item pengguna dan informasi pengguna yang dikumpulkan baik secara eksplisit maupun implisit (Hanifah, 2023).

Proses rekomendasi dimulai dengan membangun hubungan antar item dan fitur-fiturnya dalam bentuk matriks. Kemudian teknik memilih item yang paling mirip dengan item yang diinginkan pengguna dengan menghitung kemiripan berdasarkan fitur-fitur yang terkait dengan item tersebut. Perhitungan kemiripan ini menggunakan fungsi matematika seperti *adjustine cosine*, *cosine similarity*, atau *pearson coefficient*. Semakin tinggi nilai kemiripan, semakin besar nilai peluang item tersebut disarankan kepada pengguna (Kadam & Kumar, 2016).

2.3.2 *Collaborative Filtering*

Sistem rekomendasi kolaboratif atau *Collaborative Filtering* (CB) merupakan suatu metode untuk rekomendasi item dengan menggunakan persepsi atau opini orang lain. *Collaborative filtering* melakukan proses penyaringan terhadap semua pengguna untuk mendapatkan informasi pengguna dalam memberikan suatu rekomendasi. Pada proses penyaringan data, *collaborative filtering* bekerja berdasarkan kemiripan karakteristik pengguna yang nantinya akan memberikan suatu informasi baru kepada pengguna. Hal ini dikarenakan teknik akan memberikan suatu informasi berdasarkan pada pola suatu kelompok

pengguna yang hampir mirip (Herny Februariyanti, Aryo Dwi Laksono, Jati Sasongko Wibowo, 2021).

Proses paling umum dari *collaborative filtering* melibatkan perhitungan kesamaan pada Kumpulan preferensi pengguna yang biasanya di dikumpulkan oleh web dari penilaian yang diberikan terhadap item. Perhitungan yang digunakan sama dengan *content based filtering*, namun lebih fokus ke pendapat pengguna lain yang memiliki minat serupa (Kadam & Kumar, 2016).

Umumnya metode *collaborative filtering* terbagi menjadi dua teknik, yaitu *Item Based Collaborative Filtering* dan *User Based Filtering*.

2.3.2.1 *Item-Based Collaborative Filtering*

Pendekatan *item-based collaborative filtering* ini diusulkan oleh salah satu peneliti di Universitas Minnesota pada tahun 2001. Ketika jumlah pengguna dalam suatu sistem naik semakin tinggi. Ketika metode *neighbor-based collaborative filtering* atau *user-based collaborative filtering* tidak dapat menangani dengan baik karena kompleksitas dalam menemukan pengguna serupa menjadi sangat tinggi, daripada mencari pengguna serupa, sistem mencari item-item yang serupa untuk memberikan rekomendasi (Shah *et al.*, 2016).

2.3.2.2 *User-Based Collaborative Filtering*

Pendekatan *user-based collaborative filtering* ini diusulkan oleh seorang profesor dari Universitas Minnesota yaitu Jonathan L. Herlocker pada akhir tahun 1990-an. Metode ini menggunakan basis data pengguna untuk menghasilkan rekomendasi untuk pengguna. Sistem ini mengamati dan mencocokkan perilaku

pengguna dan menggunakan data tersebut untuk memprediksi perilaku pengguna di masa mendatang, atau untuk memprediksi bagaimana perilaku pengguna lain (Nugraha *et al.*, 2021).

Logika metode ini muncul karena adanya keyakinan bahwa kebanyakan orang memiliki ciri-ciri yang mirip dan memiliki kesenangan yang sama. Sebagai contoh teman bermain saat masih kecil, seseorang bisa akrab karena memiliki kesenangan yang sama, mungkin dari film atau music yang sama. Hal ini tidak menutup kemungkinan dimasa mendatang seseorang tersebut dan teman akrabnya terus menyukai film yang sama. Sehingga metode *user-based collaborative filtering* ini dapat merekomendasikan item dengan menemukan pengguna yang mirip dengan pengguna lain (Nugraha *et al.*, 2021).

2.3.2.3 *Hybrid Filtering*

Metode pendekatan hibrid atau *hybrid filtering* ini menggabungkan antar *content-based filtering* dan *collaborative filtering* untuk mendapatkan mencapai hasil rekomendasi yang lebih baik. Kedua teknik tersebut digabungkan untuk mengambil keuntungan dan membuang kelemahan kedua metode (Sorde & Deshmukh, 2015).

2.4 *Term Frequency-Inverse Document Frequency* (TF-IDF)

Term Frequency-Inverse Document Frequency (TF-IDF) merupakan ukuran seberapa pentingnya kata dalam dokumen. *Term Frequency* (TF) menyatakan banyaknya suatu kata dalam dokumen. Menurut buku *An Introduction to*

Information Retrieval edisi tahun 2009, setidaknya ada lima variasi untuk mencari nilai TF (Manning *et al.*, 2009).

1. *Natural* (*Raw*) TF

Tidak dilakukan normalisasi pada pendekatan ini, untuk memperoleh nilai TF dihitung jumlah *term* berdasarkan jumlah kemunculan *term* dalam suatu dokumen. Ketika *term i* muncul sebanyak satu kali dalam dokumen *j*, maka nilai TF nya adalah 1. Pendekatan ini yang digunakan sebagai formula dasar pada fungsi *TiIdfVectorizer()* pada Scikit-learn ketika tidak digunakan normalisasi. Rumus *natural* TF dapat dilihat pada persamaan 2.1 (Jurafsky & Martin, 2023).

$$tf_{t,d} = count(t, d) \tag{2.1}$$

Keterangan :
$tf_{t,d}$ = Jumlah dari *term t* pada dokumen *d*

2. *Logarithm* TF

Nilai TF didapatkan dengan menggunakan fungsi logaritma dalam matematika. Perhitungan ini menghindari dominasi dokumen yang mengandung sedikit *term* dalam *query* namun mempunyai frekuensi yang tinggi (Nisrina, 2020). Nilai *logarithm* TF dapat diambil dari persamaan 2.2 (Manning *et al.*, 2009).

$$ltf = 1 + \log\left(tf_{t,d}\right) \tag{2.2}$$

Keterangan :
ltf = *Logarithm* TF
$tf_{t,d}$ = Jumlah dari *term t* pada dokumen *d*
$\log\left(tf_{t,d}\right)$ = logaritma dari jumlah *term t* pada dokumen *d*

3. *Augmented* TF,

Pada dasarnya *augmented* TF digunakan untuk menghilangkan bias terhadap dokumen yang lebih panjang. Nilai *augmented* TF dapat diambil dari persamaan 2.3 (Manning *et al.*, 2009).

$$atf = 0.5 + \frac{0.5 \times tf_{t,d}}{max_t(tf_{t,d})} \quad (2.3)$$

Keterangan :
atf = *Augmented* TF
$tf_{t,d}$ = Jumlah dari *term t* pada dokumen *d*
$max_t(tf_{t,d})$ = Jumlah munculnya *term t* terbanyak dari dokumen *d*

4. *Boolean* (*Binnary*) TF

Nilai TF akan bernilai 0 apabila tidak ada *term t* pada dokumen *d* dan bernilai 1 apabila ada *term t* pada dokumen *d*.

5. *Log ave* (*Log-Average*) TF

Nilai dari *log ave* TF digunakan untuk memberikan bobot yang lebih seimbang antara dokumen dengan panjang dan distribusi istilah yang berbeda. Nilai TF pada pendekatan ini didapatkan dari persamaan 2.4 (Manning *et al.*, 2009).

$$Ltf = \frac{1 + \log(tf_{t,d})}{1 + \log(\text{ave}_{t \epsilon d}(tf_{t,d}))} \quad (2.4)$$

Keterangan :
Ltf = *Log ave* TF
$tf_{t,d}$ = Jumlah dari *term t* pada dokumen *d*
$\text{ave}_{t \epsilon d}(tf_{t,d})$ = Rata-rata jumlah kemunculan *term t* dalam dokumen *d*

Inverse Document Frequency (IDF) merupakan kebalikan dari TF. IDF menunjukkan hubungan ketersediaan *term* dalam sebuah dokumen. Nilai IDF akan semakin besar apabila jumlah dokumen yang mengandung *term t* semakin

sedikit (Nisrina, 2020). Nilai dari IDF didapatkan dari persamaan 2.5 (Manning *et al.*, 2009).

$$idf_t = \log\left(\frac{N}{df_t}\right) \tag{2.5}$$

Keterangan :
idf_t = Nilai IDF dari *term t*
N = Total semua dokumen
df_t = Jumlah dokumen yang mengandung *term t*

Nilai dari TF-IDF merupakan perkalian antara TF dan IDF, sehingga didapatkan persamaan 2.6

$$tf - idf_t = tf_{t,d} \times idf_t \tag{2.6}$$

Keterangan :
$tf - idf_t$ = Nilai TF-IDF dari *term t*
$tf_{t,d}$ = Nilai TF *term t* pada dokumen *d*
idf_t = Nilai IDF *term t*

2.5 *Singular Value Decomposition* (SVD)

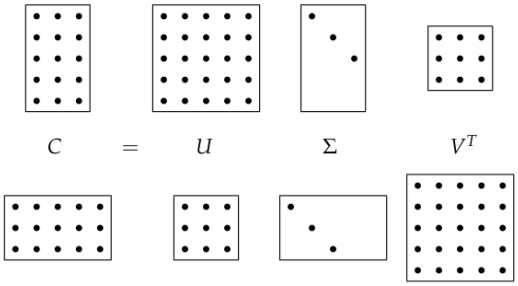
Singular Value Decomposition (SVD) merupakan algoritma transformasi matriks *orthogonal* yang memiliki nilai *singular* dengan stabilitas numerik yang baik ketika matriks berubah (Pratondo, 2023). SVD sering dibandingkan dengan dekomposisi diagonal simetris. SVD dapat diterapkan pada matriks tidak persegi. SVD memfaktorkan matriks *C* menjadi tiga matriks seperti pada persamaan 2.7 (Manning *et al.*, 2009).

$$C = U\Sigma V^T \tag{2.7}$$

Keterangan :
C = Matriks *Term Document*
U = Matriks berukuran *M x M* yang kolom-kolomnya merupakan *eigenvector* ortogonal dari CC^T.
Σ = Matriks diagonal berukuran *M x N* yang dengan nilai-nilai *singular* σ_i pada diagonalnya dari posisi pertama hingga posisi ke-*r* dan elemen elemen di luar diagonal bernilai 0.

V^T = *Transpose* dari matriks ortogonal V berukuran *N x N* yang kolomnya adalah *eigenvector* ortogonal dari $C^T C$.

Matriks $C^T C$ dan CC^T memiliki *eigenvalue* $\lambda_1, \lambda_2, \ldots, \lambda_n$ yang sama dan merupakan dasar dari dekomposisi ini. Nilai σ_i disebut dengan nilai *singular* dan diperoleh dengan mengambil akar kuadrat dari *eigenvalue* λ_i. *Eigenvalues* ini diurutkan sedemikian rupa sehingga $\lambda_1 \geq \lambda_{i+1}$ dan *eigenvalue* terbesar berada di urutan pertama. Matriks Σ memberikan informasi tentang kontribusi masing-masing komponen terhadap struktur data. Ilustrasi skema dari SVD dapat dilihat pada gambar 2.1 (Manning *et al.*, 2009).



Gambar 2.1 Skema SVD
Sumber : Manning *et al.*, 2009

Truncated SVD merupakan metode untuk menghasilkan fitur yang memfaktorkan matriks *C* dalam tiga matriks U, Σ, dan V^T berdasarkan nilai komponen *r*. *Truncated* SVD dapat menangani masalah kerenggangan matriks untuk menghasilkan matriks fitur. Berbeda dengan SVD biasa, t*runcated SVD* menghasilkan faktorisasi di mana jumlah kolom dapat ditentukan untuk sejumlah pemotongan. Pada *truncated* SVD tentunya ada kolom dari ketiga matriks yang dihilangkan. Pada matriks Σ jumlah nilai *singular* akan disederhanakan sesuai dengan jumlah *r* sehingga didapatkan matriks Σ berukuran *r* x *r* . Kolom *M* - *r*

paling kanan dari U yang berhubungan dengan baris-baris Σ yang dihilangkan tersebut; begitu juga kolom *N* - *r* paling kanan dari *V* dihilangkan karena kolom-kolom tersebut berhubungan dengan baris-baris yang akan dikalikan dengan kolom-kolom nol di Σ (Manning *et al.*, 2009). Dengan menghilangkan kolom-kolom tersebut akan terbentuk matriks yang lebih ringkas. Proses ini akan membantu memahami esensi dari informasi dalam dokumen-dokumen tersebut secara lebih sederhana dan efisien, karena mengurangi dimensi data sekaligus menyajikan pola-pola yang penting dalam sistem rekomendasi (Pratondo, 2023).

2.6 Cosine Similarity

Untuk menemukan item yang sesuai dengan preferensi pengguna dalam sistem rekomendasi, metode pengukuran kesamaan item sangat penting. Beberapa metode yang umum digunakan termasuk *cosine similarity*, *pearson coefficient*, *euclidean distance*, dan lain-lain. *Cosine similarity* sering dipilih karena memiliki tingkat perhitungan kesamaan yang baik dan tingkat presisi yang tinggi. Metode ini menggunakan sudut antara dokumen-dokumen untuk mengukur kesamaan. Persamaan *cosine similarity* antara dua item (vektor skor d_1 dan d_2) dapat didefinisikan dengan persamaan 2.8 (Manning *et al.*, 2009).

$$sim(d_1, d_2) = \frac{\vec{V}(d_1).\mathrm{V}(d_2)}{\left|\vec{V}(d_1)\right| \left|\vec{V}(d_1)\right|} \tag{2.8}$$

Keterangan :

$sim(d_1, d_2)$ = *Similarity* dari dokumen d_1 dan dokumen d_2

$\vec{V}(d_1).\mathrm{V}(d_2)$ = Dengan *denominator* merupakan *dot product* (atau biasa juga disebut *inner product*) dari vektor $\vec{V}(d_1)$ dan $\vec{V}(d_2)$. *Dot product* dari vektor $\vec{V}(d_1)$ dan $\vec{V}(d_2)$ dapat didefinisikan sebagai $\sum_{i=1}^{M} \vec{V}(d_1).\vec{V}(d_2)$ dengan *M* merupakan komponen atau jumlah *term* unik dalam dokumen $\overrightarrow{V_1}(d)$ ... $\overrightarrow{V_M}(d)$.

$|\vec{V}(d)|$ = Panjang *Euclidian* dari vektor dokumen *d* dan bisa didefinisikan dengan $\sqrt{\sum_{i=1}^{M} \vec{V}_i^2\,(d)}$.

Nilai *dot product* berbanding lurus dengan vektor dokumen d_1 dan d_2, Namun berbanding terbalik dengan perkalian dari akar jumlah vektor dokumen d_1 dan akar jumlah kuadrat vektor d_2 (Wahyuni *et al.*, 2017). Nilai kemiripan pada *cosine* similarity berkisar antara -1 hingga 1, nilai mendekati 1 mengindikasikan nilai kemiripan yang lebih tinggi (Miesle, 2023).

2.7 *Mean Average Precission* (MAP)

Mean Average Precisiion (MAP) merupakan ukuran umum yang digunakan dalam komunitas *TREC* untuk mengevaluasi kualitas sistem pencarian informasi. MAP termasuk salah satu metode yang sering digunakan untuk mengevaluasi model. *Precision* merupakan persentase dokumen relevan yang berhasil diambil oleh sistem. Dalam banyak aplikasi seperti pencarian web, pengguna biasanya lebih memperhatikan hasil pencarian yang muncul di urutan teratas. Oleh karena itu, penting untuk mengukur seberapa relevan hasil-hasil teratas tersebut. Pengukuran ini dikenal dengan istilah *precision at k*, di mana '*k*' adalah jumlah hasil rekomendasi teratas yang diberikan oleh sistem. Rumus dari *precision* adalah

$$Precision@k = \frac{Jumlah\ dari\ dokumen\ relevan\ di\ Top\ k}{k} \qquad (2.9)$$

Keterangan :
k = Jumlah item teratas yang direkomendasikan sistem

Average precission (AP) merupakan nilai rata-rata dari *precision* setiap item yang relevan yang dihasilkan. MAP merupakan nilai rata-rata dari *average*

precision tersebut (Manning *et al.*, 2009). *Precision* dalam AP dihitung dengan mempertimbangkan urutan item yang dihasilkan oleh sistem, sehingga *precision* diberikan untuk setiap item yang dihasilkan berdasarkan urutan tersebut (Hasan, 2018). Misalkan himpunan dokumen relevan untuk sebuah rekomendasi q_j adalah $\{d_1, d_2, \dots, d_{m_j}\}$. Jika R_{jk} adalah himpunan hasil rekomendasi terurut dari yang teratas hingga dokumen ke-*k*, maka (Manning *et al.*, 2009):

$$\text{MAP(Q)} = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{m_j}\sum_{k=1}^{m_j} Precision(R_{jk}) \qquad (2.10)$$

Keterangan :
Q = Banyaknya *query* yang dievaluasi.
R_{jk} = Hasil rekomendasi diurutkan berdasarkan relevansi dari urutan pertama sampai urutan ke- k untuk *query* ke-*j.*
m_j = banyaknya jumlah dokumen yang relevan untuk *query* ke-*j*.

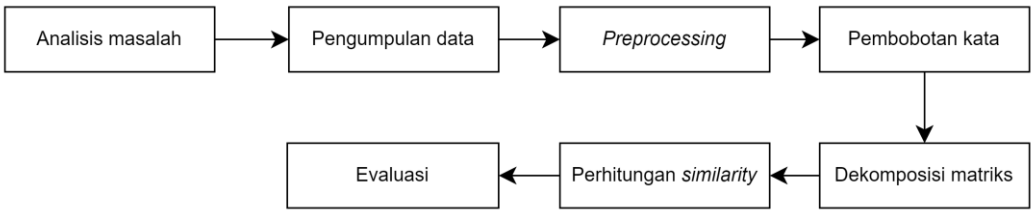
Jika dokumen di peringkat k relevan maka bernilai 1, jika dokumen tidak relevan maka bernilai 0. Nilai MAP antara 0 hingga 1. Semakin mendekati 1 maka suatu sistem dikatakan baik.

BAB III

METODOLOGI PENELITIAN

3.1 Desain Sistem

Desain sistem merupakan tahap yang akan digunakan oleh peneliti untuk melakukan penelitian. Dalam proses pembuatan sistem rekomendasi film animasi pada penelitian ini diperlukan beberapa tahapan dan proses, seperti analisis masalah, pengumpulan data, *preprocessing*, pembobotan kata, dekomposisi matriks dan evaluasi. Gambar 3.1 menunjukkan langkah-langkah penelitian.



Gambar 3.1 Desain sistem penelitian

Pada proses pembobotan kata, teks diubah menjadi representasi numerik menggunakan TF-IDF untuk menentukan bobot dalam dokumen. Kombinasi TF dan IDF memberikan bobot yang lebih tinggi kepada kata tang sering muncul dalam satu dokumen tapi jarang muncul di dokumen lainnya. Setelah proses pembobotan kata, dilakukan teknik dekomposisi matriks dengan menggunakan SVD. Proses ini membantu mengidentifikasi struktur laten dalam teks, mengurangi kompleksitas komputasi, dan meningkatkan kinerja model dengan mereduksi matriks renggang. Setelah matriks TF-IDF di dekomposisi oleh SVD, *cosine similarity* menghitung kesamaan antara vektor-vektor teks untuk menentukan dokumen yang paling relevan atau mirip dengan dokumen atau *query*

tertentu. Kombinasi teknik-teknik ini memungkinkan sistem untuk lebih efektif dalam mengelola, menganalisis, dan merekomendasikan konten teks yang relevan. Pada tahap terakhir, dilakukan evaluasi atau uji coba dengan menggunakan MAP.

3.2 Analisis Masalah

Penelitian ini bertujuan untuk membangun sistem rekomendasi pemilihan film animasi yang sesuai dengan minat pengguna. Sistem ini menggunakan metode *content based filtering* dengan algoritma TF-IDF kemudian didekomposisi menggunakan SVD. Selain metode *content based filtering* memberikan hasil rekomendasi yang tinggi, metode ini tidak terpengaruh oleh masalah *cold start*, yaitu ketika pengguna item atau pengguna baru belum memiliki cukup data untuk memberikan rekomendasi yang akurat karena metode ini merekomendasikan item baru berdasarkan fiturnya tanpa memerlukan *rating* atau *feedback* dari pengguna lain.

Penerapan pembobotan dengan metode TF-IDF dipilih karena memberikan hasil yang akurat dan mudah diinterpretasikan serta dapat menangkap makna yang relevan dengan konteks dan tujuan teks tersebut. Metode ini sangat sesuai untuk pengolahan teks yang tidak terstruktur dan memiliki dimensi tinggi, seperti fitur *overview* dalam penelitian ini. Penggunaan *singular value decomposition* (SVD) dalam penelitian bertujuan untuk mengatasi masalah *sparsity* pada matriks TF-IDF serta mengurangi beban komputasi. Terutama pada sistem rekomendasi denagn banyak fitur, SVD membantu mengurangi kebutuhan komputasi menjadi lebih efisien. Untuk menghitung kemiripan antara konten diukur dengan metode *cosine similarity*. Metode ini dipilih karena telah terbukti memiliki tingkat akurasi

yang tinggi, sesuai dengan referensi yang disebutkan dalam artikel-artikel terkait. Oleh karena itu, penulis memilih untuk mencoba penggunaan *cosine similarity* pada matriks hasil dekomposisi matriks TF-IDF dengan menggunakan SVD. Dengan demikian, pendekatan ini diharapkan dapat memberikan hasil yang optimal dalam memberikan rekomendasi film animasi kepada pengguna.

3.3 Pengumpulan Data

Langkah awal dilakukan pengumpulan data yang akan dalam penelitian. Data yang digunakan dalam penelitian ini di dapat dari situs *kaggle.com* dengan judul *52000 Animation Movie Details*. Data set tersebut diunggah oleh pengguna bernama @ASANICZKA yang diambil melalui *scrapping* dari *website* TMDB (*The Movie Database*). Data set ini memiliki 52.000 total data film animasi yang diambil dari situs TMDB yang di*filter* sehingga tersisa film dengan genre animasi saja. Data set ini memiliki lisensi ODC *Attribution License*, sebuah lisensi yang digunakan dalam *Open Data Commons* (*ODC*), sebuah organisasi yang mempromosikan penggunaan data secara terbuka. Sampel data dalam penelitian ini ditunjukkan pada Tabel 3.1.

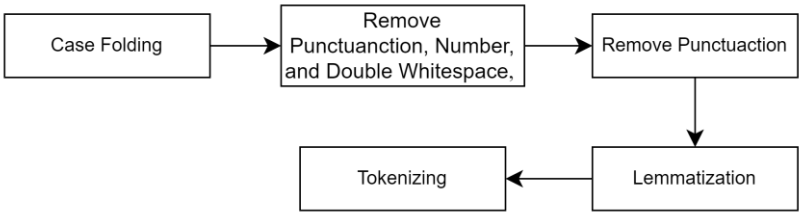
Tabel 3.1 Sampel deskripsi data film animasi

Fitur	Penjelasan	Contoh
Title	Judul film animasi	*The Incredibles*
Vote_average	Rata-rata *vote* pengguna TMDB	7.704
Vote_count	Banyaknya pengguna yang melakukan *vote*	16584
Status	Status film, seperti sudah rilis atau masih dalam masa produksi	*Released*
Release_date	Tanggal rilis film tersebut	2004-10-27
Runtime	Durasi film (jam)	115
Homepage	Halaman ke *website* resmi film	http://disney.go.com/disneyvideos/animatedfilms/incredibles/main.html
Overview	Ringkasan singkat film	*"Bob Parr has given up his superhero days to log in time as*

Fitur	Penjelasan	Contoh
		an insurance adjuster and raise his three children with his formerly heroic wife in suburbia."
Tagline	*Tagline* singkat film	*"No gut, no glory"*
Genres	Genre film	*Family, Animation, Drama*
Production_companies	Perusahaan pembuat film	Walt Disney
Production_countries	Negara asal film	United States of America
Spoken_language	Bahasa yang digunakan dalam film	*French, English*

3.4 *Preprocessing*

Preprocessing merupakan langkah awal untuk mengubah data mentah menjadi informasi yang lebih bersih dan siap diolah lebih lanjut. *Preprcessing* pada penelitian ini memanfaatkan *Natural Language Processing* (NLP), yang merupakan cabang dari kecerdasan buatan (*intelligent system*) berkaitan dengan kemampuan komputer untuk memahami kata-kata. Tahapan yang dilakukan dapat dilihat pada Gambar 3.2.



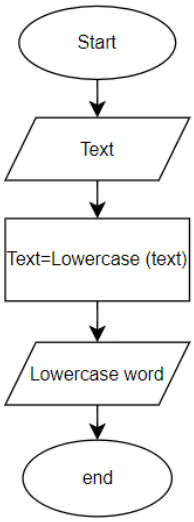
Gambar 3.2 Tahapan preprocessing

Tahapan *preprocessing* pada penelitian ini adalah *case folding*, *remove punctuaction*, *remove double whitespace*, *remove number*, *stopward removal*, *stemming*, dan *tokenizing*.

3.4.1 Case Folding

Case folding adalah proses mengubah semua huruf dalam teks menjadi huruf kecil (*lowercase*). Tujuan dari *case folding* adalah mengurangi kerumitan

dalam pembandingan teks. *Case folding* memastikan bahwa semua huruf dalam teks diubah menjadi huruf kecil. Algoritma dasar dari *case folding* dapat dilihat pada gambar 3.3.



Gambar 3.3 Flowcart *case folding*
Sumber : Nuha, 2020

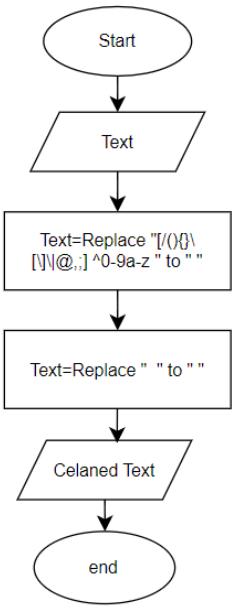
Proses dimulai dengan *input* teks. Kemudian semua huruf pada teks diubah menjadi *lowercase*. Dengan demikian, kata yang sama ditulis dengan huruf kapital (*uppercase*) atau huruf kecil (*lowercase*) akan dianggap identik. Contohnya kata "Adventure" dan "adventure" akan dianggap sama setelah proses *case folding*. Contoh proses *case folding* pada tabel 3.2.

Tabel 3.2 Proses *case folding*

Sebelum	Sesudah
A magic adventure of wonder and heroism.	a magic adventure of wonder and heroism.
Embark on a magical adventure and become an epic hero.	embark on a magical adventure and become an epic hero.
a fantasy worlds with brave warriors and mythical creatures.	a fantasy worlds with brave warriors and mythical creatures.
In a world of epic wonders and brave warriors, fantasy and magic.	in a world of epic wonders and brave warriors, fantasy and magic.

3.4.2 *Remove Punctuanction, Number, and Double Whitespace*

Remove punctuaction merupakan proses dalam NLP untuk menghapus tanda baca karena tidak memberikan kontribusi yang signifikan terhadap makna teks. Angka (digit) yang tidak bermanfaat terhadap proses rekomendasi juga dihapus dengan proses *remove number*. Terakhir untuk mengatasi masalah spasi ganda (*double whitespace*) dilakukan proses *remove whitespace*. Ketiga proses tersebut dilakukan untuk membersihkan teks dari karakter-karakter yang dianggap tidak relevan dalam proses sistem rekomendasi. *Flowchart remove punctuaction, numer,* dan *double whitespace* dapat dilihat pada gambar 3.4.



Gambar 3.4 *Flowcart remove punctuaction, number,* dan *double whitespace*

Proses pertama dilakukan dengan *input* teks. Kemudian diperiksa apakah karakter berupa tanda baca atau huruf yang tidak digunakan, jika tidak maka karakter akan dihilangkan. Terakhir, dicek apakah ada *double whitespace*

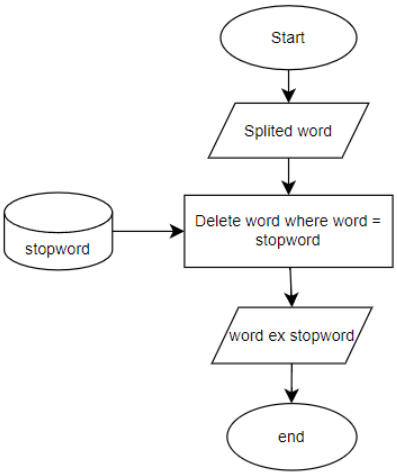
berlebih, jika ada maka dikurangi menjadi *single whitespace*. Contoh dari proses penghapusan angka, simbol dan spasi berlebih dapat dilihat pada Tabel 3.3

Tabel 3.3 Proses *remove punctuanction, number, and double whitespace*

Sebelum	Sesudah
a magic adventure of wonder and heroism.	a magic adventure of wonder and heroism
embark on a magical adventure and become an epic hero.	embark on a magical adventure and become an epic hero
a fantasy worlds with brave warriors and mythical creatures.	a fantasy worlds with brave warriors and mythical creatures
in a world of epic wonders and brave warriors, fantasy and magic.	in a world of epic wonders and brave warriors fantasy and magic

3.4.3 *Stopward Removal*

Stopword Removal adalah proses menghapus kata-kata umum yang tidak memiliki makna atau kontribusi signifikan dalam analisis teks. Kata-kata semacam ini, yang sering disebut "*stopwords,*" termasuk kata-kata seperti "*the*" "*is*", "*in*", "*and,*" "*of,*" dan sebagainya. Dalam analisis teks, kata-kata ini sering kali dianggap tidak penting karena mereka muncul begitu sering dalam banyak dokumen dan tidak memberikan wawasan yang berarti tentang konten teks yang sedang diolah. Dengan menghapus *stopwords,* teks yang diolah menjadi lebih bersih dan fokus pada kata-kata yang lebih penting. Proses algoritma *flowchart* dari *stopword removal* dapat dilihat pada gambar 3.5.



Gambar 3.5 *Flowcart stopward removal*
Sumber : Nuha, 2020

Pertama, melakukan *split* terhadap teks, kemudian daftar kata-kata *stopwords* diinisialisasi. Setiap kata diperiksa apakah termasuk dalam *stopwords* atau tidak. Jika kata adalah *stopword*, kata tersebut dihapus dari teks. Hasil akhirnya adalah teks tanpa *stopwords*. Contoh hasil *stopword removal* adalah pada tabel 3.4.

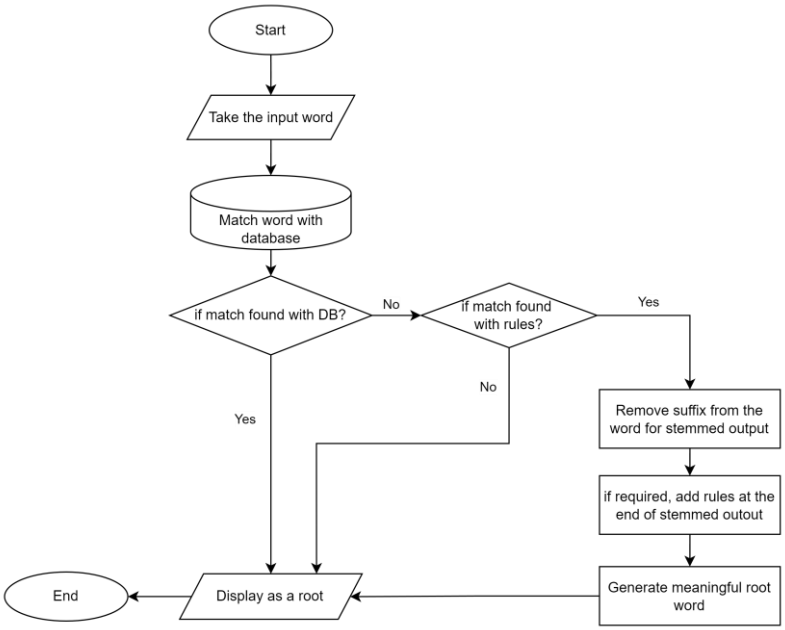
Tabel 3.4 Proses *stopword removal*

Sebelum	Sesudah
a magic adventure of wonder and heroism	magic adventure wonder heroism
embark on a magical adventure and become an epic hero	embark magical adventure become epic hero
a fantasy worlds with brave warriors and mythical creatures	fantasy worlds brave warriors mythical creatures
in a world of epic wonders and brave warriors fantasy and magic	world epic wonders brave warriors fantasy magic

3.4.4 *Lemmatization*

Lemmatization merupakan proses dalam *Natural Language Processing* (NLP) yang bertujuan untuk mengubah kata-kata yang berbeda namun memiliki akar kata sama ke dalam bentuk dasarnya. Misalnya *running*, *runner*, dan *ran*

akan diubah menjadi kata dasarnya yaitu *run*. *Lemmatization* bekerja dengan menemukan akar dasar dari sebuah kata (*lemma*) dengan memperhatikan analisis morfologi dan kamus (Cheryl, 2022). Algoritma sederhana dari proses *lemmatization* adalah pada gambar 3.6



Gambar 3.6 *Flowcart lemmatization*
Sumber : Gupta *et al.* (2015)

Proses dimulai dengan mengambil *input* teks yang telah melalui proses penghapusan stopwords sebelumnya. Kata masukan ini kemudian diperiksa terhadap database yang sudah ada, yang berisi kata-kata dan lemmanya. Jika kata tersebut ditemukan dalam database, maka kata tersebut tidak diubah karena sudah berada dalam bentuk akar kata.

Jika kata tersebut tidak ditemukan dalam *database*, proses dilanjutkan dengan memeriksa apakah kata tersebut sesuai dengan aturan linguistik yang telah ditentukan untuk mengidentifikasi bentuk akarnya. Jika sesuai dengan aturan, sufiks akan dihilangkan, dan jika diperlukan, tambahan aturan akan diterapkan

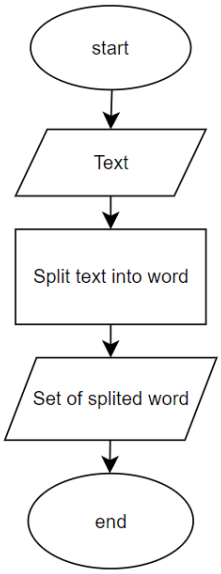
untuk membentuk kata dasar yang benar. Namun, jika kata tersebut tidak sesuai dengan aturan yang telah ditetapkan, kata tersebut tidak dapat direduksi lebih lanjut dan langsung ditampilkan. Contoh hasil *lemmatization* dapat dilihat pada tabel 3.5.

Tabel 3.5 Proses *lemmatization*

Sebelum	Sesudah
magic adventure wonder heroism	magic adventure wonder heroism
embark magical adventure become epic hero	embark magical adventure become epic hero
fantasy worlds brave warriors mythical creatures	fantasy world brave warrior mythical creature
world epic wonders brave warriors fantasy magic	world epic wonder brave warrior fantasy magic

3.4.5 *Tokenizing*

Tokenizing (tokenisasi) merupakan sebuah metode untuk memisahkan input data tekstual menjadi *token* terpisah yang dapat dipahami dan di proses lebih lanjut oleh mesin. Pada penelitian ini akan menggunakan *word tokenizing* (tokenisasi kata), yaitu tokenisasi yang paling umum. Tokenisasi ini dilakukan dengan memisahkan kata dengan kalimatnya. Gambar 3.7 menggambarkan *flowcart* dari proses *tokenizing.*



Gambar 3.7 *Flowcart tokenizing*

Proses *tokenizing* dimulai dengan memasukkan teks. Kemudian teks tersebut dipisah menjadi per-kata berdasarkan dari spasi yang ada. Lalu teks yang sudah terpisah akan menjadi potongan kecil kata-kata. Contoh dari dokumen film yang telah di *tokenizing* dapat dilihat pada Tabel 3.6

Tabel 3.6 Proses *tokenizing*

Sebelum	Sesudah
magic adventure wonder heroism	[magic, adventure, wonder, heroism]
embark magical adventure become epic hero	[embark, magical, adventure, become epic, hero]
fantasy world brave warrior mythical creature	[fantasy, world, brave, warrior, mythical, creature]
world epic wonder brave warrior fantasy magic	[world, epic, wonder, brave, warrior, fantasy, magic]

3.5 Pembobotan Kata

Pembobotan kata dilakukan dengan menggunakan algoritma TF-IDF. Pembobotan kata merupakan ukuran penting kata tersebut dalam mempresentasikan data film animasi. Bobot makin tinggi apabila frekuensi kemunculan kata semakin tinggi, akan tetapi bobot akan berkurang apabila kata

tersebut sering muncul pada data film animasi lain. Misal terdapat 4 dokumen yang telah melewati *preprocessing* berikut :

a. Dokumen 1 : “magic adventure wonder heroism”
b. Dokumen 2 : “embark magical adventure become epic hero”
c. Dokumen 3 : “fantasy world brave warrior mythical creature”
d. Dokumen 4 : “world epic wonder brave warrior fantasy magic”

Dari keempat dokumen di atas kemudian dilakukan perhitungan dengan menggunakan rumus 2.6. Detail pembobotan dapat dilihat pada tabel 3.7

Tabel 3.7 Pembobotan TF-IDF

No	Kata	TF				df	IDF	TF-IDF			
		Doc 1	Doc 2	Doc 3	Doc 4			Doc 1	Doc 2	Doc 3	Doc 4
1	adventure	1	1	0	0	2	ln(4/2)=1, 693147	1, 693147	1, 693147	0	0
2	become	0	1	0	0	1	ln(4/1)=2, 386294	0	2, 386294	0	0
3	brave	0	0	1	1	2	ln(4/2)=1, 693147	0	0	1, 693147	1, 693147
4	creature	0	0	1	0	1	ln(4/1)=2, 386294	0	0	2, 386294	0
5	embark	0	1	0	0	1	ln(4/1)=2, 386294	0	2, 386294	0	0
6	epic	0	1	0	1	2	ln(4/2)=1, 693147	0	1, 693147	0	1, 693147
7	fantasy	0	0	1	1	2	ln(4/2)=1, 693147	0	0	1, 693147	1, 693147
8	hero	0	1	0	0	1	ln(4/1)=2, 386294	0	2, 386294	0	0
9	heroism	1	0	0	0	1	ln(4/1)=2, 386294	2, 386294	0	0	0
10	magic	1	0	0	1	2	ln(4/2)=1, 693147	1, 693147	0	0	1, 693147
11	magical	0	1	0	0	1	ln(4/1)=2, 386294	0	2, 386294	0	0
12	mythical	0	0	1	0	1	ln(4/1)=2, 386294	0	0	2, 386294	0
13	warrior	0	0	1	1	2	ln(4/2)=1, 693147	0	0	1, 693147	1, 693147
14	wonder	1	0	0	1	2	ln(4/2)=1, 693147	1, 693147	0	0	1, 693147
15	world	0	0	1	1	2	ln(4/2)=1, 693147	0	0	1, 693147	1, 693147

Ada sedikit perbedaan antara persamaan IDF yang digunakan di *sklearn* dan yang tertulis pada persamaan (2.5). Sesuai dokumentasi *online sklearn*, persamaan IDF pada *sklearn* dapat ditulis sebagai berikut apabila nilai *smooth_idf* adalah "True":

$$idf(t) = \log\left[\frac{(1+n)}{1+df(t)}\right] + 1 \quad (3.1)$$

Ketika nilai *smooth_idf* adalah "False", maka persamaan IDF :

$$idf(t) = log\left[\frac{n}{df(t)}\right] + 1 \quad (3.2)$$

Keterangan :
$idf(d,t)$ = Frekuensi dokumen di mana banyaknya dokumen *d* yang mengandung *term t*
n = Jumlah seluruh dokumen

Logaritma yang digunakan pada persamaan (3.1) dan (3.2) merupakan logaritma natural (ln) (Unnikrishnan, 2021). Penulisan kedua rumus di atas juga sedikit berbeda dengan beberapa literatur yang menyebutkan penambahan +1 berada di bagian penyebut bukan di akhir.

3.6 Dekomposisi Matriks

Pada tahap ini matriks yang didapatkan dari pembobotan TF-IDF akan mengalami proses dekomposisi matriks, lalu didapatkan pengurangan dimensi. Pengurangan dimensi memiliki tujuan untuk melakukan seleksi pada matriks fitur. Dalam proses ini fitur yang kurang informatif akan dibuang. Proses ini mempermudah sistem dalam melakukan pencarian informasi pada setiap profil konten serta dapat meningkatkan akurasi pada sistem (Pratondo, 2023).

Sistem rekomendasi dengan menggunakan SVD dapat mengatasi masalah *sparse matrix* yang sering terjadi pada sistem rekomendasi. *Sparse matrix*

merupakan kondisi di mana banyak atau Sebagian elemen dari matriks bernilai nol. Tentunya dengan semakin padatnya nilai matriks akan memberikan hasil rekomendasi yang lebih baik (Siringoringo *et al.*, 2021). Proses SVD menghasilkan 3 matriks yaitu U, S, dan V^T . Matriks U dan V merupakan matriks orthogonal, sedangkan S adalah matriks diagonal di mana eleman lain selain dari diagonalnya bernilai 0.

Dari hasil perhitungan TF-IDF pada Tabel 3.7 dapat dibentuk sebuah matriks :

$$A = \begin{pmatrix} 1.693147 & 1.693147 & 0 & 0 \\ 0 & 0 & 1.693147 & 1.693147 \\ 0 & 0 & 2.386294 & 0 \\ 0 & 2.386294 & 0 & 0 \\ 0 & 1.693147 & 0 & 1.693147 \\ 0 & 0 & 1.693147 & 1.693147 \\ 0 & 2 & 0 & 0 \\ 2 & 0 & 0 & 0 \\ 1.693147 & 0 & 0 & 1.693147 \\ 0 & 2.386294 & 0 & 0 \\ 0 & 0 & 2.386294 & 0 \\ 0 & 0 & 1.693147 & 1.693147 \\ 1.693147 & 0 & 0 & 1.693147 \\ 0 & 0 & 1.693147 & 1.693147 \end{pmatrix}$$

Dengan asumsi kolom pada matriks A mempresentasikan dokumen, sedangkan baris pada matriks A melambangkan *term*. Selanjutnya matriks A ingin dekomposisi dengan nilai *n_component* = 3. Matriks didekomposisi menjadi tiga matriks, U, Σ, V^T. Langkah pertama adalah mencari nilai matriks dari AA^T dan A^TA.

$$AA^T = \begin{pmatrix} 5.73349475 & 0 & \dots & 2.86674738 & 0 \\ 0 & 5.73349475 & \dots & 2.86674738 & 5.73349475 \\ \dots & \dots & \dots & \dots & \dots \\ 2.86674738 & 2.86674738 & \dots & 5.73349475 & 2.86674738 \\ 0 & 5.73349475 & \dots & 2.86674738 & 5.73349475 \end{pmatrix}$$

$$A^TA = \begin{pmatrix} 14.2946429 & 2.86674738 & 0 & 5.73349475 \\ 2.86674738 & 22.81669708 & 0 & 2.86674738 \\ 0 & 0 & 22.85579106 & 11.4669895 \\ 5.73349475 & 2.86674738 & 11.4669895 & 20.06723163 \end{pmatrix}$$

Kemudian setelah didapatkan nilai matriks AA^T dan A^TA, dicari nilai *eigenvalues* dan *eigenvectors* dari kedua matriks tersebut. Diasumsikan nilai *n_component* = 3, maka matriks Σ didapatkan dengan nilai-nilai *singular*nya merupakan akar masing-masing *eigenvalues* matriks A^TA dan diurutkan dari nilai terbesar hingga terkecil.

$$\Sigma = \begin{pmatrix} \sqrt{34.35803827} & 0 & 0 \\ 0 & \sqrt{23.58269759} & 0 \\ 0 & 0 & \sqrt{14.97997001} \end{pmatrix}$$

$$\Sigma = \begin{pmatrix} 5.86157302 & 0 & 0 \\ 0 & 4.85620197 & 0 \\ 0 & 0 & 3.87039662 \end{pmatrix}$$

Kemudian nilai matriks *U* didapatkan dengan mengambil nilai masing-masing *eigenvectors* dari matriks AA^T. Karena *n_component* (*r*) = 3, maka diambil tiga nilai terbesar setiap *eigenvector*nya. Berikut adalah nilai matriks *U*

$$U = \begin{pmatrix} 0.12892824 & 0.40319186 & 0.16841531 \\ 0.38762343 & -0.13407151 & -0.0569851 \\ 0.27273582 & -0.17769399 & -0.25645091 \\ 0.09058687 & 0.43954152 & -0.23851139 \\ 0.25838331 & 0.30387553 & -0.04425647 \\ 0.38762343 & 0.13407151 & -0.0569851 \\ 0.09058687 & -0.43954152 & -0.23851139 \\ 0.09112252 & -0.12871059 & 0.47587322 \\ 0.25876337 & -0.0833317 & 0.46262076 \\ 0.09058687 & -0.43954152 & -0.23851139 \\ 0.27273582 & 0.17769399 & -0.25645091 \\ 0.38762343 & 0.13407151 & -0.0569851 \\ 0.25876337 & -0.0833317 & 0.46262076 \\ 0.38762343 & 0.13407151 & -0.0569851 \end{pmatrix}$$

Pada contoh ini, kolom pada matriks *U* mempresentasikan komponen *r*, sementara baris matriks *U* mempresentasikan *term*.

Kemudian matriks V^T didapatkan dengan mengambil nilai masing-masing *eigenvectors* dari matriks $A^T A$. Karena *n_component* (*r*) = 3, maka diambil tiga nilai terbesar setiap *eigenvector*nya. Berikut adalah nilai matriks V^T

$$V^T = \begin{pmatrix} 0.22382875 & 0.22251301 & 0.6699345 & 0.67199436 \\ 0.26193107 & 0.89448411 & -0.36161418 & -0.02292317 \\ 0.7718319 & -0.3868482 & -0.4159448 & 0.2856815 \end{pmatrix}$$

Masing-masing kolom pada matriks V^T mempresentasikan tiap dokumen, sementara baris pada matriks V^Tmempresentasikan komponen *r*. Matriks V^T akan digunakan untuk perhitungan *similarity*.

3.7 Perhitungan *Similarity*

Tahap ini adalah tahap terakhir dalam proses pembelajaran. Pada tahap ini, data hasil *preprocessing* akan diukur tingkat kesamaannya dengan data dokumen lainnya. Proses perhituangan *similarity* menggunakan algoritma *cosine* similarity.

Dengan pengukuran ini akan dapat diketahui kemiripan antara satu konten dengan konten lainnya.

Pada perhitungan dekomposisi matriks sebelumnya, diketahui nilai matriks *U* merupakan representasi *term* pada dokumen dengan komponen *r* pada suang vektor. Selanjutnya matriks *A* akan ditranformasikan dengan matriks U agar bisa dihitung *similarity* nya. Dari perkalian tersebut didapatkan matriks A' berikut

$$A' = \begin{pmatrix} 1.311198856 & 1.27199019 & 2.98729557 \\ 1.30427624 & 4.34379551 & -1.49725597 \\ 3.92686999 & -1.7560715 & -1.7560715 \\ 3.938944 & -0.11131953 & 1.10570072 \end{pmatrix}$$

Kolom pada matriks A' mempresentasikan komponen *r* dan barisnya merepresentasikan dokumen. Kemudian semua nilai vektor yang didapat dimasukkan pada persamaan (2.8). Hasil dari rumus persamaan ini nantinya akan digunakan sebagai dasar untuk memberikan rekomendasi kepada pengguna. Tabel 3.8 menunjukkan nilai vektor *d1*, *d2*, *d3*, dan *d4*.

Tabel 3.8 Nilai vektor dokumen *d1*, *d2*, *d3*, dan *d4*

Tid	***Component_1***	***Component_2***	***Component_3***
d1	1.311198856	1.2719901956	2.98729557
d2	1.30427624	4.34379551	-1.49725597
d3	3.92686999	-1.7560715	-1.60987133
d4	3.938944	-0.11131953	1.10570072

Misalkan ingin mencari nilai persamaan antara dokumen 3 (*d3*) dan dokumen 4 (*d4*) digunakan persamaan (2.8)

$$sim(T1, T2) = \frac{(3.927 \times 3.939) + (-1.756 \times -0.111) + (-1.610 \times 1.106)}{\sqrt{3.927^2 + (-1.756)^2 + (-1.610)^2}\sqrt{3.939^2 + (-0.111)^2 + 1.106^2}}$$

$$sim(T1, T2) = \frac{15{,}468453 + 0{,}194916 + (-1{,}78066)}{\sqrt{15.421329 + 3.083536 + 2{,}5921}\sqrt{15{,}515721 + 0.012321 + 1.223236}}$$

$$sim(T1, T2) = \frac{13.882709}{\sqrt{21.096965}\sqrt{16{,}751278}}$$

$$sim(T1, T2) = \frac{13.882709}{18,798964} = 0,7384827$$

Diperoleh *d3* dan *d4* memiliki tingkat kemiripan 0,7384827 yang menunjukkan kemiripan yang bagus, karena semakin mendekati nilai 1 maka *similarity* akan semakin tinggi dan jika mendekati -1 maka *similarity* semakin rendah. Hasil dari rekomendasi berupa judul dan skor *similarity* dari film dengan *input* dari pengguna.

3.8 Skenario Uji Coba

Uji coba atau evaluasi digunakan untuk mengetahui seberapa bagus hasil rekomendasi yang diberikan oleh pengguna. Metode evaluasi yang digunakan adalah *Mean Average Precission* (MAP). Hasil dari penelitian ini adalah berupa nilai *similarity* dan judul dari tiap-tiap dokumen. Nilai tersebut kemudian di rangking dari terbesar hingga terkecil. Kemudian dilakukan evaluasi dengan MAP berdasarkan rangking (*rank-k*) hasil setiap *query* yang dimasukkan.

Pada tahap awal, *precision* dihitung dengan membagi jumlah dokumen relevan dengan semua dokumen yang ditampilkan oleh sistem berdasarkan *rank-k*. Relevansi hasil rekomendasi dengan *query* adalah berdasarkan kesamaan genre animasi yang diambil dari *dataset movie*. Kemudian *Average Precision* (AP) didapat dengan rata-rata *precision* dari *query* pada *rank-k*. Pada tahap terakhir, MAP diperoleh dengan membagi AP masing-masing *query* dengan *rank-k* dokumen yang terlibat. Terdapat 15 judul film animasi sebagai *query*. Perhitungan MAP dilakukan dengan *rank-k* = 10 dan *rank-k* = 15.

3.9 Implementasi GUI

Sistem rekomendasi yang telah berhasil dibuat selanjutnya akan diimplementasikan dalam bentuk *Graphical User Interface* (GUI) pada sebuah *website*. Pembuatan *website* dilakukan dengan menggunakan Flask sebagai *framework* untuk sistemnya.

3.10 Komponen Perencanaan Sistem

Perancangan sistem ini bertujuan untuk menciptakan sebuah sistem rekomendasi yang akan diimplementasikan pada sebuah situs web. Komponen-komponen utama dari perancangan ini mencakup perangkat keras (hardware) dan perangkat lunak (software). Dalam penelitian ini, digunakan beberapa perangkat yang memiliki peran kunci. Berikut adalah daftar perangkat yang digunakan :

1. Perangkat keras
    a) Laptop Dell Vostro 3405
    b) Prosesor AMD Ryzen 7
    c) RAM 8 GB
    d) SSD 512 GB
2. Perangkat lunak
    a) Sistem Operasi Windows 11
    b) Visual Studio Code
    c) Google Colab
    d) Web Browser
    e) Bahasa pemrograman Python

BAB IV

HASIL DAN PEMBAHASAN

4.1 Pengumpulan Data

Data yang digunakan dalam penelitian ini diambil dari situs kaggle.com dengan judul *52000 Animation Movie Details*. Data set tersebut diunggah oleh pengguna bernama @ASANICZKA yang diambil melalui *scrapping* dari *website* TMDB (*The Movie Database*). Data berjumlah 52.000 film animasi. Karena data terlalu banyak dan belum bersih, penulis melakukan *filter* dengan mengambil film yang fitur *overview* yang tidak bernilai *Nan*. Selanjutnya, data yang diambil adalah film dengan tahun rilis lebih dari 1995 dan memiliki popularitas yang tinggi. Sehingga didapatkan 3.856 film animasi. Fitur yang digunakan dalam sistem rekomendasi *overview*, *tagline*, *production_companies*, dan *genres.*

4.2 Preprocessing

Tahapan *preprocessing* dilakukan untuk membersihkan kalimat (*cleaning*) pada fitur yang tidak perlu seperti angka, simbol, dan kata yang tidak memiliki arti sehingga proses berjalan lebih maksimal. Sebelum dilakukan *preprocessing*, semua fitur yang digunakan dalam proses rekomendasi akan digabungkan menjadi satu kolom bernama ‘features’. Ada beberapa tahapan pada proses *preprocessing* ini yaitu *case folding*, *remove punctuaction*, *remove number*, *remove whitespace*, *stopword removal*, dan *tokenizing*.

4.2.1 Case Folding

Case folding dilakukan dengan mengubah semua teks menjadi huruf kecil (*lowercase*). Pada penelitian ini, proses *case folding* dilakukan dengan menggunakan fungsi *lower()* dari *library* pandas. *Pseudocode* proses *case folding* ditunjukkan pada Gambar 4.1.

```
1.  FUNCTION convert_to_lowercase(features):
2.      lowercase_features = []  // inisialisasi list kosong
3.      FOR feature IN features:
4.          lowercase_feature = feature.lower()  // Konversi ke lowercase
5.          lowercase_features.append({"case_folding": lowercase_feature})
6.      RETURN lowercase_features
```

Gambar 4.1 *Pseudocode* proses *case folding*

4.2.2 Remove Punctuaction, Number and Whitespace

Untuk menghilangkan karakter tersebut diperlukan tahapan *remove punctuaction*. Selain itu untuk mengatasi teks dalam dokumen memiliki spasi berlebih dan angka yang tidak dibutuhkan dilakukan tahapan *remove whitespace* dan *remove number*. Gambar 4.2 dan gambar 4.3 merupakan *pseudocode* untuk proses *remove punctuaction* dan *remove whitespace*.

```
1.  FUNCTION remove_punctuation(features):
2.      cleaned_features = []
3.      FOR feature IN features:
4.          clean_feature = remove_punct(feature)
5.          cleaned_features.append(clean_feature)
6.      RETURN cleaned_features
7.
8.  FUNCTION remove_punct(text):
9.      special_chars_regex = re.compile('[/(){}\[\]\|@,;]')
10.     symbols_regex = re.compile('[^0-9a-z ]')
11.     digits_regex = re.compile(r'\b\d+\b')
12.     alphanumeric_digits_regex = re.compile(r'\b\d+\w+\b')
13.
14.     text = special_chars_regex.sub('', text)
15.     text = symbols_regex.sub(' ', text)
16.     text = digits_regex.sub('', text)
17.     text = alphanumeric_digits_regex.sub('', text)
18.
19.     return text
```

Gambar 4.2 *Pseudocode* proses *remove punctuaction*

Proses *remove punctuaction* dan *remove number* menggunakan beberapa pola *Regular Expression* (Regex) untuk mengenali beberapa pola pada teks. Setelah pola didapatkan, langkah selanjutnya adalah mengubah karakter-karakter yang telah didapatkan menjadi spasi.

```
1.  FUNCTION remove_double_whitespace(features):
2.      cleaned_features = []
3.      FOR feature IN features:
4.          clean_feature = _normalize_whitespace(feature)
5.          cleaned_features.append(clean_feature)
6.      RETURN cleaned_features
7.
8.  FUNCTION _normalize_whitespace(text):
9.      # Regular expressions untuk menghapus double whitespace
10.     tab_replacement_regex = re.compile(r"\t\t", re.DOTALL)
11.     whitespace_regex = re.compile(r"( )\1+", re.DOTALL)
12.     newline_regex = re.compile(r"(\n)\1+", re.DOTALL)
13.     carriage_return_regex = re.compile(r"(\r)\1+", re.DOTALL)
14.     return_text = text.strip(" ")
15.
16.     # Mengaplikasikan RegEx secara berurutan
17.     return_text = tab_replacement_regex.sub(r"\t", return_text)
18.     return_text = whitespace_regex.sub(r"\1", return_text)
19.     return_text = newline_regex.sub(r"\1", return_text)
20.     return_text = carriage_return_regex.sub(r"\1", return_text)
21.
22.     return return_text
```

Gambar 4.3 *Pseudocode* proses *remove double whitespace*

Proses *remove double whitespace* juga menggunakan *Regex* untuk membersihkan teks dengan mengahapus *double whitespace* (spasi ganda), tab ganda, *newline* (baris baru) ganda, dan *carriage return* (kembali ke baris) ganda.

4.2.3 Stopward Removal

Stopward removal adalah proses untuk mengambil kata-kata penting dalam suatu dokumen dan membuang kata-kata tidak penting (*stopword*). *Stopword* tersebut diambil *library* nltk pada Python ditambah kata yang menurut penulis tidak memiliki makna penting seperti *ii*, *also*, *another*, *please*, *may*, dan lain

sebagainya. *Pseudocode* untuk proses *stopward removal* ditampilkan pada gambar 4.4.

```
1.  FUNCTION remove_stopwords(features):
2.      cleaned_features = []
3.      FOR feature IN features:
4.          clean_feature = remove_stopwords(feature)
5.          cleaned_features.append(clean_feature)
6.      RETURN cleaned_features
7.
8.  FUNCTION remove_stopwords(text):
9.      words = text.split()
10.
11.     filtered_words = [word for word in words if word.lower() not in
    stopword]
12.
13.     return ' '.join(filtered_words)
```

Gambar 4.4 *Pseudocode* proses *stopword removal*

Proses *remove stopwords* dimulai dengan memisahkan teks menjadi sebuah daftar kata, setiap spasi akan menjadi batas pemisah. Kemudian iterasi dilakukan pada individu kata, jika kata termasuk *stopwords* maka akan dihilangkan. Namun, apabila kata bukan termasuk *stopwords* maka akan ditambahkan ke dalam daftar *filtered_words*.

4.2.4 Lemmatization

Setelah proses *stopword removal* teks akan menjadi lebih bersih berisi kata-kata penting yang digunakan dalam proses rekomendasi. Namun, dari kata-kata tersebut terkadang masih banyak kata yang sebetulnya memiliki arti sama namun dianggap berbeda oleh sistem karena memiliki kata imbuhan atau *affix*. Fungsi dari *lemmatization* adalah mengubah kata-kata ke dalam bentuk akarnya berdasarkan morfologi dan kamus sehingga kata-kata dianggap sama oleh sistem. Proses dari *lemmatization* dapat dilihat pada gambar 4.5.

```
# Inisialisasi WordNetLemmatizer (initialization)
lemmatizer = WordNetLemmatizer()

FUNCTION lemmatize_text(features):
    cleaned_features = []
    FOR feature IN features:
        clean_feature = lemmatize_text(feature)
        cleaned_features.append(clean_feature)
    RETURN cleaned_features

FUNCTION lemmatize_text(text):
    words = text.split()
    lemmatized_words = [lemmatizer.lemmatize(word) for word in words]
    return ' '.join(lemmatized_words)
```

Gambar 4.5 *Pseudocode* proses *lemmatization*

Proses lematisasi dilakukan dengan menggunakan modul *WordNetLemmatier* dari *library* nltk untuk mengubah kata menjadi bentuk dasarnya. Kemudian juga digunakan *WordNet* yang merupakan basis data lesikal dalam bahasa Inggris yang sering digunakan untuk pemrosesan bahasa alami. *WordNet* menyediakan informasi morfologis tentang kata-kata, termasuk bentuk dasar (*lemma*), jenis kata, dan hubungan antar kata. Selanjutnya teks di *split* menjadi kata-kata individual dan menerapkan *lemmatization* dengan iterasi terhadap kata individual. Setelah semua proses iterasi selesai, semua kata digabungkan kembali.

4.2.5 Tokenizing

Proses *tokenizing* digunakan untuk memisah kata pada sebuah teks berdasarkan kata penghubung atau spasi menjadi token-token atau bagian-bagian tertentu. Proses *tokenizing* menggunakan fungsi *word_tokenize* yang telah tersedia di *library* nltk. Fungsi tersebut kemudian dimasukkan ke dalam parameter saat mendefinisikan vektor TF-IDF.

4.3 Pembobotan Kata

Pada tahap pembobotan kata, kumpulan fitur yang telah melalui proses *preprocessing* akan diubah menjadi sebuah matriks dengan algoritma TF-IDF. Untuk melakukan *transformasi* data dari kumpulan kata menjadi sebuah matriks menggunakan fungsi *TfidfVectorizer* yang telah disediakan dalam *library* Scikit-learn. Gambar 4.6 menunjukkan kode yang digunakan untuk proses TF-IDF.

```
1.  # Dengan asumsi TfidfVectorizer telah diimpor dari
    sklearn.feature_extraction.text
2.
3.  FUNCTION tfidf_vectorize(features):
4.      # Buat objek vektorizer TF-IDF dengan parameter yang diinginkan
5.      vectorizer = TfidfVectorizer(
6.          encoding='latin-1',
7.          ngram_range=(1, 1),
8.          tokenizer=word_tokenize,
9.          Norm=None,
10.         use_idf=True,
11.         analyzer='word',
12.         stop_words=None,
13.         smooth_idf=False,
14.         Sublinear_tf=False
15.     )
16.
17.     tfidf_matrix = vectorizer.fit_transform(features)
18.
19.     return tfidf_matrix
```

Gambar 4.6 Pseudocode proses pembobotan kata dengan *TF-IDF*

Proses perhitungan kata dengan *term frequency inverse document frequency* dengan berbagai parameter. Parameter *ngram_range=(1,1)* digunakan untuk mengatur vektorisasi hanya mempertimbangkan *unigram* atau kata tunggal. Parameter *tokenizer* yang diisi dengan modul *word_tokenize* yang telah di *import* sebelumnya digunakan untuk melakukan tokenisasi. Pada penelitian ini juga tidak digunakan *smoothing*, *sublinear_idf*, dan *normalisasi*. Dari proses tersebut didapatkan hasil dari TF-IDF. Tabel 4.1 adalah sampel hasil perhitungan *Term Frequency Inverse Document Frequency*.

Tabel 4.1 Sampel hasil perhitungan TF-IDF

Dokumen ke-	aa	…	across	act	actas	...	acting	action	…	zzzax
1	0	…	0	0	0	...	0	0	…	0
2	0	…	0	0	0	...	0	0	…	…
3	0	…	0	0	0	...	0	0	…	0
4	0	…	0	0	0	...	0	0	…	0
5	0	…	4,966926	0	0	...	0	0	…	0
6	0	…	0	0	0	...	0	0	…	0
7	0	…	0	0	0	...	0	0	…	0
8	0	…	0	0	0	...	0	0	…	0
9	0	…	0	0	0	...	0	0	…	0
10	0	…	0	0	0	...	0	0	…	0
11	0	…	0	0	0	...	0	0	…	0
12	0	…	0	6,03851	0	...	0	0	…	0
13	0	…	0	0	0	...	0	0	…	0
14	0	…	0	0	0	...	0	0	…	0
15	0	…	0	0	0	...	0	2,795917	…	0
16	0	…	0	0	0	...	0	0	…	0
17	0	…	4,966926	0	0	...	0	0	…	0
18	0	…	0	0	0	...	0	0	…	0
19	0	…	4,966926	0	0	...	0	0	…	0
20	0	…	0	0	0	...	0	0	…	0
…	…	…	…	…	…	...	…	…	…	…
3856	0	…	0	0	0	...	0	0	…	0
df	1	…	73	63	4	...	7	640	…	1
idf	9.2574	…	4.9669	6.1365	7.8711	...	7.3115	2.7959	…	9.2574

Hasil perhitungan TF-IDF tersebut kemudian dibuat menjadi sebuah matriks dengan dimensi 3856 baris x 20440 kolom. Matriks tersebut digunakan untuk melakukan dekomposisi matriks dengan menggunakan SVD.

4.4 Dekomposisi Matriks

Pada tahap ini, penulis akan melakukan ekstraksi fitur dengan melakukan dekomposisi matriks. Algoritma yang digunakan adalah *Singular Value Decomposition* (SVD). Ketika matriks TF-IDF mempunya dimensi yang besar pasti akan memakan banyak komputasi untuk perhitungannya, sehingga digunakan dekomposisi matriks seperti SVD. Pada penelitian ini, Gambar 4.7 menunjukkan proses dekomposisi matriks dengan SVD.

```
1.  # Dengan asumsi TruncatedSVD telah di import dari sklearn.decomposition
2.
3.  FUNCTION perform_tsvd(tfidf_matrix, num_topics=1000):
4.      # Set random seed
5.      np.random.seed(100)
6.
7.      # Buat objek TruncatedSVD dengan jumlah topik yang diinginkan
8.  svd = TruncatedSVD(n_components= num_topics, random_state=100)
9.
10.     # Terapkan SVD ke TF-IDF matrix
11.     svd_matrix = svd.fit_transform(tfidf_matrix)
12.
13.     return svd_matrix
```

Gambar 4.7 *Pseudocode* proses Perhitungan *Singular Value Decomposition*

Untuk melakukan dekomposisi matriks dengan menggunakan SVD, penulis menggunakan fungsi *TruncatedSVD* dari *library* scikit-learn pada python. Kemudian Langkah selanjutnya adalah mengatur jumlah *component* yang diinginkan, pada penelitian ini menggunakan jumlah komponen sebanyak 1000 dan menggunakan *random_state* bernilai 100 untuk menghitung SVD. Maka dihasilkan matriks sebesar 1000 x *N*, di mana *N* adalah banyaknya dokumen. Hasil proses dekomposisi ditunjukkan pada tabel 4.2.

Tabel 4.2 Sampel hasil perhitungan SVD

Dokumen ke-	Comp_1	Comp_2	Comp_3	Comp_4	Comp_5	...	Comp_996	Comp_997	Comp_998	Comp_999	Comp_999
0	7.289732	-1.059515	2.751886	0.824903	-2.734397	...	2.777972	2.378788	0.703867	-0.944729	0.418383
1	5.466489	-0.872366	1.539651	-0.217145	-1.386795	...	0.122491	-1.558913	0.158642	0.438734	-0.687196
2	3.386052	-0.693112	0.722330	0.201111	-1.449699	...	1.799696	-0.443422	0.338365	-2.093481	0.192631
3	7.686792	-1.399050	2.419467	0.195621	-3.304336	...	1.337610	0.396602	0.597665	0.446921	-0.002673
4	9.257360	-1.384215	1.808940	-1.275493	0.130231	...	0.002747	-0.380871	0.080304	0.263472	-0.494440
...	...	...	...	...	...	...	...	...	...	...	...
3851	5.699544	0.624816	1.244716	-2.047858	3.134197	...	-0.378108	0.643637	0.628790	0.350049	0.744491
3852	2.627090	-0.473858	1.590995	-0.415269	-0.233422	...	0.132878	0.048270	0.105585	-0.263733	-0.647648
3853	6.160484	-0.683684	1.601215	-0.258735	0.132496	...	1.359085	0.403987	0.574033	-1.587951	-0.186721
3854	6.039937	-1.195232	-0.865103	0.702644	-0.220732	...	0.082499	-0.661204	-0.329778	-0.755322	-0.741227
3855	4.457490	-1.452891	0.026108	0.556881	-1.233644	...	0.949820	0.222183	0.070936	0.306594	1.414654

4.5 Perhitungan *Similarity*

Matriks hasil SVD sebelumnya akan dihitung kemiripan antar dokumennya. Nilai hasil dari *cosine similarity* ini yang akan dijadikan acuan dalam menentukan tingkat kemiripan antar dokumen. Gambar 4.8 merupakan proses perhitungan *similarity* dengan menggunakan *cosine similarity*.

```
# Dengan asumsi cosine_similarity adalah fungsi dari perpustakaan yang sesuai

FUNCTION calculate_cosine_similarity(svd_matrix):
    // Hitung cosine similarity antara setiap pasangan baris
    cos_sim = cosine_similarity(svd_matrix_float)

    return cos_sim
```

Gambar 4.8 *Pseudocode* proses Perhitungan *Cosine Similarity*

Perhitungan dilakukan dengan menggunakan fungsi *cosine_similarity* dari *library* scikit-learn untuk menghitung *similarity* kosinus antara tiap pasang dokumen yang sebelumnya telah direduksi dimensinya dengan menggunakan SVD. Hasil dari perhitungan ini adalah matriks simetris yang menunjukkan nilai *cosine_similarity* antara setiap pasang dokumen dalam data set. Nilai-nilai dalam matriks tersebut berkisar dari -1 hingga 1, di mana nilai yang lebih tinggi menunjukkan tingkat kesamaan yang lebih tinggi antara dokumen-dokumen tersebut, sedangkan nilai negatif menunjukkan arah yang berlawanan. Tabel 4.3 menunjukkan sampel hasil dari *cosine similarity*.

Tabel 4.3 Sampel hasil perhitungan *cosine similarity*

	0	**1**	**2**	**3**	**4**	**5**	**...**	**3850**	**3851**	**3852**	**3853**	**3854**
0	1.000000	0.113177	0.027443	0.041506	0.084101	0.061031	...	0.012600	0.001302	0.043139	-0.002133	-0.013246
1	0.113177	1.000000	0.100201	0.031560	0.104585	0.127221	...	0.000307	0.124667	0.147266	-0.010949	0.007351
2	0.027443	0.100201	1.000000	0.031949	0.011205	0.072881	...	-0.009318	-0.008083	0.022047	0.130112	0.026350
3	0.041506	0.031560	0.031949	1.000000	0.037958	0.009839	...	-0.017013	0.035163	0.073840	0.012459	0.009643
4	0.084101	0.104585	0.011205	0.037958	1.000000	0.018804	...	-0.006000	0.047144	0.095373	0.008380	0.001254
...	...	...	...	...	...	...	...	...	...	...	...	...
3851	0.001302	0.124667	-0.008083	0.035163	0.047144	0.043732	...	0.013355	1.000000	0.123282	0.018293	0.028741
3852	0.043139	0.147266	0.022047	0.073840	0.095373	0.011288	...	-0.002458	0.123282	1.000000	0.083435	0.031689
3853	-0.002133	-0.010949	0.130112	0.012459	0.008380	0.002232	...	0.007847	0.018293	0.083435	1.000000	0.002621
3854	-0.013246	0.007351	0.026350	0.009643	0.001254	0.108182	...	0.007541	0.028741	0.031689	0.002621	1.000000
3855	0.042418	0.051271	0.058039	0.001166	0.014439	0.082670	...	0.009294	0.012907	0.035110	0.005344	0.019751

4.6 Hasil Rekomendasi Berdasarkan Judul

Setelah semua dokumen dihitung *similarity* nya menggunakan *cosine similarity*, selanjutnya dokumen akan di rangking dari dokumen yang memiliki nilai *similarity* terbesar hingga terkecil, lalu diambil 20 dokumen teratas sebagai hasil rekomendasi. Sebagai sampel, penulis mencoba memasukkan judul film "Toy Story". Tabel 4.4 menunjukkan 15 film teratas yang dihasilkan sistem dengan *query* "Toy Story".

Tabel 4.4 Hasil rekomendasi berdasarkan judul

No	Dokumen ke-	Judul Film Animasi	Skor Similarity
1	10193	*Toy Story 3*	0.66610
2	863	*Toy Story 2*	0.66012
3	301528	*Toy Story 4*	0.55867
4	6187	*Buzz Lightyear of Star Command: The Adventure Begins*	0.48758
5	82424	*Small Fry*	0.43993
6	718789	*Lightyear*	0.37650
7	77887	*Hawaiian Vacation*	0.24884
8	462883	*Woody Woodpecker*	0.23599
9	660859	*Forky Asks a Question: What Is Love?*	0.20557
10	852719	*Twenty Something*	0.18948
11	594530	*Lamp Life*	0.18506
12	175574	*Free Birds*	0.16532
13	660862	*Forky Asks a Questions. What is Leader*	0.15859
14	615960	*Toy Guardian*	0.14923
15	265923	*VeggieTales: The Toy That Saved Christmas*	0.14905

4.7 Hasil Rekomendasi Berdasarkan Kata Kunci

Berbeda dengan sub bab 4.4 sebelumnya yang hanya dapat memberikan rekomendasi untuk judul film animasi yang tersedia di dalam *database*, ketika judul atau teks *input* yang diberikan tidak ada di dalam *database* maka sistem tidak bisa memberikan rekomendasi. Untuk itu ditambahkan fitur rekomendasi berdasarkan kata kunci (*keyword)*. Gambar 4.9 adalah *pseudocode* untuk proses film animasi berdasarkan kata kunci.

```
1.  FUNCTION get_user_recommendations(user_input, num_recommendations=15):
2.
3.    # Proses Preprocessing
4.
5.    # Mendapatkan judul film
6.    titles = movies.index  # Assuming 'movies' is a dictionary with titles
    as the index
7.
8.    # Menghitung User Input TF-IDF
9.    user_tfidf_weights = tfidf_vectorizer.transform([user_input])
10.
11.   # Proyeksikan TF-IDF user input ke SVD yang telah ada
12.   user_svd_weights = svd.transform(user_tfidf_weights)
13.
14.   columns=tfidf_vectorizer.get_feature_names_out())
15.
16.   # Menghitung Cosine Similarity
17.   user_cosine_sim = cosine_similarity(user_svd_weights, svd_matrix)
18.
19.   # Mencari rekomendasi film
20.   user_sim_scores = list(enumerate(user_cosine_sim[0]))
21.   user_sim_scores = sorted(user_sim_scores, key=lambda x: x[1],
    reverse=True)
22.   user_sim_scores = user_sim_scores[:num_recommendations
23.   user_movies_indices = [i[0] for i in user_sim_scores]
24.
25.   # Membuat data frame rekomendasi
26.   user_recommendations = pd.DataFrame(columns=['Id', 'Title', 'Score'])
27.   for i in range(num_recommendations):
28.     movie_idx = user_movies_indices[i]
29.     user_recommendations.at[i, 'Id'] = movie_idx
30.     user_recommendations.at[i, 'Title'] = titles[movie_idx]
31.     user_recommendations.at[i, 'Score'] = user_sim_scores[i][1]
32.
33.   return user_recommendations
```

Gambar 4.9 *Pseudocode* proses rekomendasi berdasarkan kata kunci

Proses dimulai dengan mengimpor modul 'string', yang menyediakan konstanta dan fungsi untuk manipulasi *string*. Kemudian, dilakukan preprocessing pada *input* pengguna, seperti mengonversi huruf menjadi *lowercase*, menghapus angka dan tanda baca, serta melakukan lematisasi kata-kata untuk mengubahnya menjadi bentuk dasarnya. Selanjutnya, *input* pengguna diubah menjadi vektor bobot TF-IDF menggunakan TF-IDF Vectorizer yang telah di-*training* sebelumnya, memungkinkan representasi *input* untuk perhitungan *similarity*. Proyeksi vektor bobot TF-IDF *input* pengguna dilakukan ke dalam ruang SVD

yang telah direduksi sebelumnya, sehingga *input* memperoleh representasi dalam dimensi yang sesuai dengan ruang SVD yang telah di-*training*. Akhirnya, dilakukan perhitungan *similarity* antara *input* pengguna dan dokumen dalam *database* menggunakan *cosine similarity*. Hasil *similarity* diurutkan dari tertinggi ke terendah, dan 15 film animasi dengan skor *similarity* tertinggi ditampilkan kepada pengguna. Tabel 4.5 menunjukkan hasil rekomendasi berdasarkan kata kunci dengan sampel *input* "*Andy heads off to Cowboy Camp, leaving his toys to their own devices*".

Tabel 4.5 Hasil rekomendasi berdasarkan *keyword*

No	Dokumen ke-	Judul Film Animasi	Skor Similarity
1	19	Toy Story 2	0.81879
2	17	Toy Story 3	0.63042
3	6	Toy Story	0.53343
4	38	Toy Story 4	0.49237
5	2309	Toy Guardians	0.44242
6	781	Lamp Life	0.40107
7	369	Toy Story of Terror!	0.38943
8	1417	How the Toys Saved Christmas	0.33367
9	3479	VeggieTales: The Toy That Saved Christmas	0.32409
10	471	Hawaiian Vacation	0.31056
11	1772	Forky Asks a Question: What Is Love?	0.30022
12	2903	The Forgotten Toys	0.26908
13	1541	Scooby-Doo! Haunted Holidays	0.24828
14	1632	Rudolph the Red-Nosed Reindeer & the Island of Misfit Toys	0.23123
15	572	Partysaurus Rex	0.22589

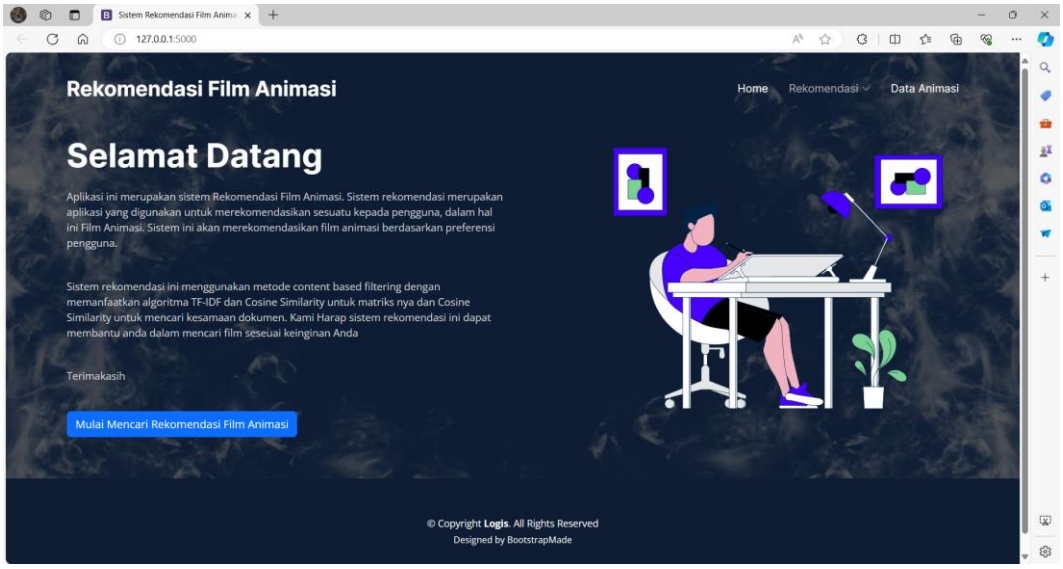
4.8 Perancangan Antarmuka Sistem

Sistem rekomendasi yang telah dibuat dalam penelitian ini telah diimplementasikan dalam antarmuka sistem, ini memungkinkan pengguna untuk berinteraksi dengan sistem secara langsung. Melalui antarmuka, pengguna dalam memasukkan preferensi berupa judul ataupun kata kunci lalu menerima hasil

rekomendasi. Implementasi ke dalam antarmuka sistem meningkatkan kegunaan dan aksebilitas.

4.8.1 Tampilan Awal

Tampilan awal sistem rekomendasi film animasi berisi sapaan singkat penulis dan menjelaskan secara singkat apa yang dimaksud dengan sistem rekomendasi serta metode apa yang digunakan seperti pada Gambar 4.10

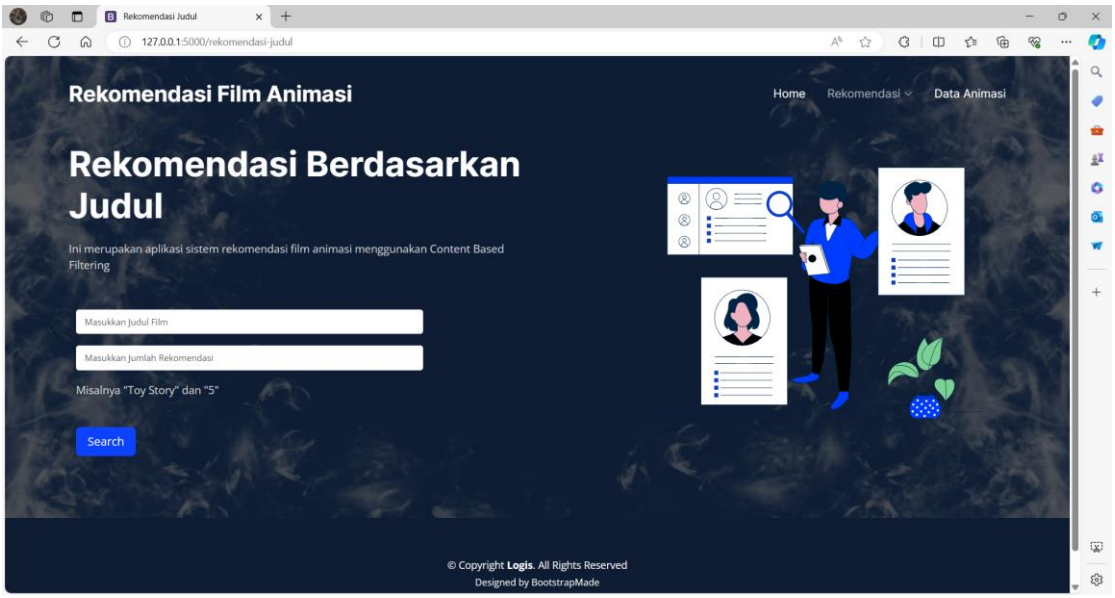


Gambar 4.10 Halaman Awal Sistem Rekomendasi

4.8.2 Halaman Rekomendasi Berdasarkan Judul

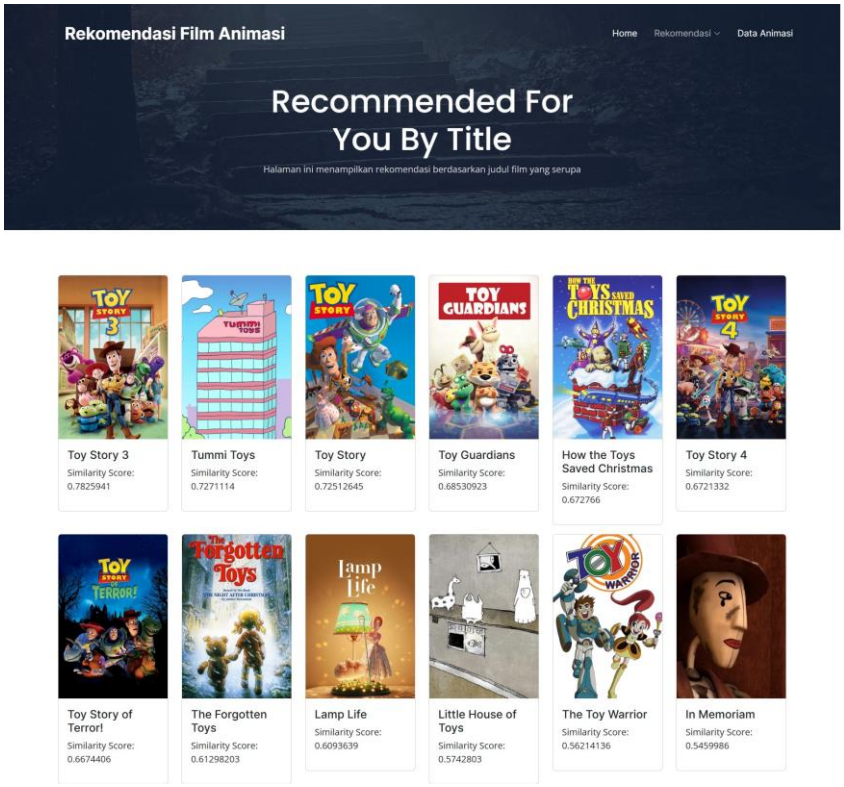
Tombol menuju halaman kedua ada di tombol “Mulai Mencari Rekomendasi Film Animasi” pada halaman awal atau menu “Rekomendasi” pada *header*. Pada halaman ini, pengguna dapat memasukkan judul film animasi yang diminati dan dapat menentukan jumlah rekomendasi animasi yang diberikan oleh sistem. Setelah menekan tombol “*Search*”, sistem akan menampilkan jumlah rekomendasi film animasi berdasarkan judul dan jumlah rekomendasi yang di-

input pengguna. Gambar 4.11 menunjukkan halaman rekomendasi berdasarkan judul.



Gambar 4.11 Halaman *Form* Rekomendasi Berdasarkan Judul

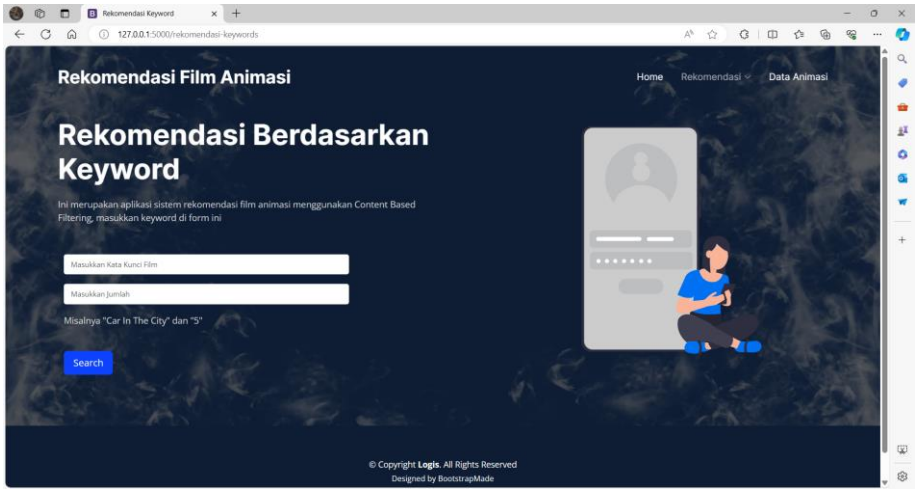
Sistem akan memberikan rekomendasi berdasarkan tingkat kemiripan yang dihitung menggunakan *cosine similarity*. Gambar 4.12 menunjukkan hasil rekomendasi berdasarkan judul.



Gambar 4.12 Halaman hasil rekomendasi berdasarkan judul

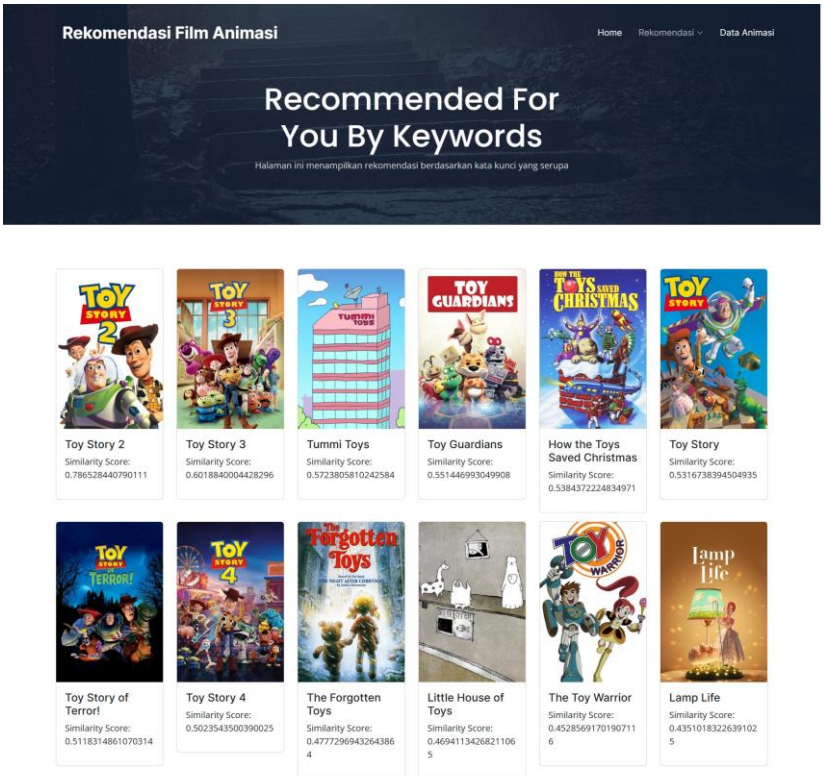
4.8.3 Halaman Rekomendasi Berdasarkan *Keyword*

Tombol menuju halaman rekomendasi berdasarkan *keyword* berada di menu "Rekomendasi" pada *header*. Pada halaman ini, pengguna dapat memasukkan *keyword* berdasarkan referensi pengguna, pengguna juga dapat menentukan jumlah rekomendasi animasi yang diberikan oleh sistem. Setelah menekan tombol "*Search*", sistem akan menampilkan jumlah rekomendasi film animasi berdasarkan judul dan jumlah rekomendasi yang di-*input* pengguna. Gambar 4.13 menunjukkan halaman rekomendasi berdasarkan *keyword*.



Gambar 4.13 Halaman *form* rekomendasi berdasarkan *keywords*

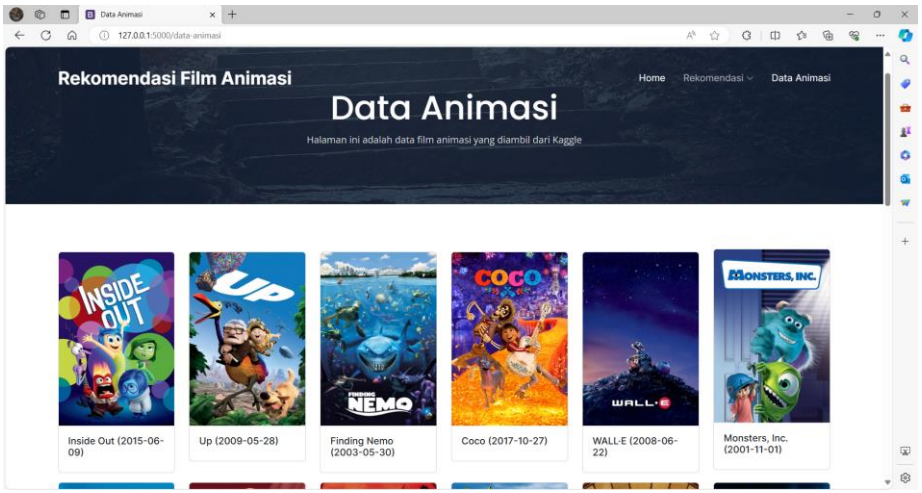
Sistem akan memberikan rekomendasi berdasarkan Tingkat kemiripan yang dihitung menggunakan *cosine similarity*. Gambar 4.14 menunjukkan hasil rekomendasi berdasarkan *keyword*.



Gambar 4.14 Halaman hasil rekomendasi berdasarkan *keyword*

4.8.4 Halaman Data Film Animasi

Halaman terakhir digunakan untuk mengecek dan memberikan data film animasi yang tersedia. Gambar 4.15 menunjukkan halaman data film animasi



Gambar 4.15 Halaman data film animasi

4.9 Analisa Uji Coba

Pengujian MAP akan dilakukan dengan mengambil 15 sampel judul animasi sebagai *query* untuk mengukur seberapa baik kinerja model sistem rekomendasi pemilihan film animasi. Pengujian dilakukan dengan melakukan analisis hasil rekomendasi teratas yang diberikan oleh sistem. Relevansi hasil rekomendasi dengan judul *query* adalah berdasarkan kesamaan genre dan tokoh film animasi yang diambil dari *dataset movie*. Pengujian ini juga akan membandingkan sistem rekomendasi yang telah ada sebelumnya, yaitu menggunakan metode TF-IDF tanpa SVD dengan metode yang digunakan dalam penelitian ini, yaitu TF-IDF dengan SVD. Tabel 4.6 menunjukkan detail perhitungan MAP metode TF-IDF dengan SVD pada *rank-k* = 15 dan tabel 4.7 pada *rank* = 10.

Tabel 4.6 Detail perhitungan MAP TF-IDF.SVD pada *rank-k* = 15

Query	**Nilai presisi pada tiap *rank-k***															**AP**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	
Kubo and The Two Strings	1	1	1				0,57			0,5				0,43	0,47	0.710
Cars	1	1	1	1	1	1	1	1	1	1		0,83	0,85	0,86	0,87	0.957
Toy Story	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
The Boss Baby	1		0,67	0,75		0,67	0,71	0,75	0,78	0,8	0,82	0,83	0,85	0,86	0,87	0.796
Ratatouille	1	1	1	1	1	1	1	1	1		0,91	0,92	0,92	0,93	0,93	0.972
Spirited Away	1		0,67	0,75	0,8	0,83		0,75		0,7	0,73		0,69	0,71	0,73	0.760
Your Name	1	1	1	1	1	1	1		0,89		0,82	0,83	0,85	0,86	0,87	0.932
Puss in Boots: The Last Wish	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Cloudy with a chance	1		0,67	0,75		0,67		0,625	0,67		0,64	0,67			0,6	0.697
SpongeBob Movie: Sponge on the Run	1	1	1	1	1	1	1	1		0,9	0,91	0,92	0,92	0,93	0,93	0.965
The Secret Life of Pets	1	1	1	1		0,83	0,86	0,875		0,8	0,82		0,77	0,79	0,8	0.878
The Super Mario Bros Movie	1	1	1	1	1	1		0,875	0,89	0,9	0,91	0,92	0,92	0,93	0,93	0.948
Happy Feet	1	1		0,75		0,67	0,71		0,67		0,64		0,62	0,64	0,67	0.736
Rise of the Guardians	1	1	1	1		0,83	0,86	0,875		0,8	0,82	0,83	0,85		0,8	0.889
Tangled		0,5		0,5	0,6				0,44	0,5		0,5	0,54		0,53	0.515
MAP																**0.850**

Tabel 4.7 Detail perhitungan MAP TF-IDF.SVD pada *rank-k* = 10

Query	**Nilai presisi pada tiap *rank-k***										**AP**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
Kubo and The Two Strings	1	1	1				0.57			0.5	0.814
Cars	1	1	1	1	1	1	1	1	1	1	1
Toy Story	1	1	1	1	1	1	1	1	1	1	1
The Boss Baby	1		0.67	0.75		0.67	0.71	0,75	0.78	0.8	0.766
Ratatouille	1	1	1	1	1	1	1	1	1		1
Spirited Away	1		0.67	0.75	0.8	0.83		0,75		0.7	0.786
Your Name	1	1	1	1	1	1	1		0.89		0.986
Puss in Boots: The Last Wish	1	1	1	1	1	1	1	1	1	1	1
Cloudy with a chance	1		0.67	0.75		0.67		0,625	0.67		0.729
SpongeBob Movie: Sponge on the Run	1	1	1	1	1	1	1	1		0.9	0.989
The Secret Life of Pets	1	1	1	1		0.83	0.85	0,875		0.8	0.921
The Super Mario Bros Movie	1	1	1	1	1	1		0,875	0.89	0.9	0.963
Happy Feet	1	1		0.75		0.67	0.71		0.67		0.800
Rise of the Guardians	1	1	1	1		0.83	0.85	0,875		0.8	0.921
Tangled		0.5		0.5	0.6				0.44	0.5	0.509
MAP											**0.88**

Pada tabel 4.6, dengan rank-k = 15, nilai MAP keseluruhan adalah 0.850. Mayoritas query memiliki nilai *Average Precision* (AP) yang tinggi, dengan dua *query* "Toy Story," dan "Puss in Boots: The Last Wish" mencapai nilai AP sempurna (1). Pada tabel 4.7, dengan rank-k = 10, nilai MAP keseluruhan sedikit lebih tinggi, yaitu 0.88. Nilai AP untuk *query* individu tetap tinggi, dengan beberapa *query* mencapai nilai AP sempurna (1). Nilai presisi pada *rank-k* 1 hingga 10 cenderung lebih stabil dan tinggi dibandingkan dengan *rank-k* 1 hingga 15. Konsistensi nilai presisi juga lebih baik pada *rank-k* = 10, menunjukkan peningkatan nilai AP dan stabilitas yang lebih tinggi.

Secara keseluruhan, metode TF-IDF dan SVD menunjukkan performa yang sangat baik dalam menghasilkan hasil pencarian yang relevan, dengan nilai MAP yang tinggi pada kedua *rank-k*. Namun, *rank-k* = 10 menunjukkan performa yang lebih stabil dan konsisten dibandingkan dengan rank-k = 15. Nilai MAP pada *rank-k* = 10 yang lebih baik daripada pada *rank*=15 menandakan bahwa metode ini lebih efektif dalam menempatkan hasil yang paling relevan di posisi teratas.

Kemudian metode TF-IDF.SVD yang telah dibuat di atas akan dibandingkan dengan metode TF-IDF tanpa menggunakan SVD yang telah banyak digunakan sebelumnya dengan cara yang sama yaitu berdasarkan kesamaan genre. Tabel 4.8 dan tabel 4.9 menunjukkan detail perhitungan MAP metode TF-IDF tanpa menggunakan SVD.

Tabel 4.8 Detail perhitungan MAP TF-IDF pada *rank-k* = 15

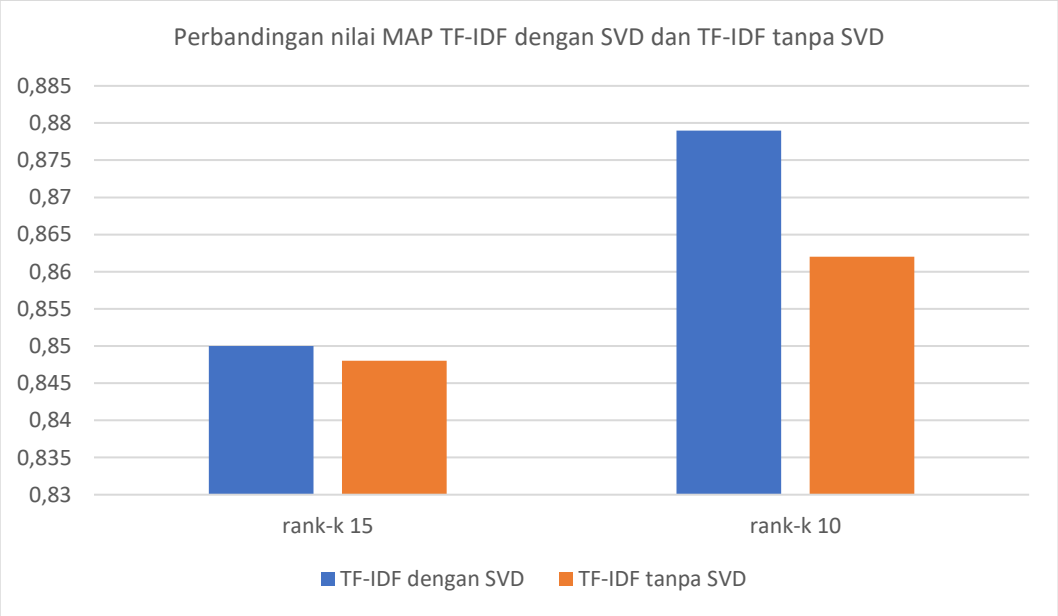
Query	**Nilai presisi pada tiap *rank-k***															**AP**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	
Kubo and The Two Strings	1		0,67						0,33	0,4		0,42	0,46	0,4	0,53	0,53
Cars	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Toy Story	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
The Boss Baby	1	1		0,75	0,8	0,83	0,86		0,78	0,8	0,82	0,83		0,79		0,84
Ratatouille	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Spirited Away		0,5	0,67		0,6	0,67	0,71		0,67	0,7	0,73	0,75	0,77	0,79	0,8	0,70
Your Name	1	1	1	1	1		0,86	0,88	0,89		0,82	0,83			0,73	0,91
Puss in Boots: The Last Wish	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
Cloudy with a chance	1	1			0,6	0,67	0,71	0,75	0,78	0,8		0,75	0,77	0,79	0,8	0,78
SpongeBob Movie: Sponge on the Run	1	1	1	1	1	1	1	1	1		0,91	0,92	0,92	0,93	0,93	0,97
The Secret Life of Pets	1	1	1	1	1	1	1	1	1	1	1		0,92	0,93	0,93	0,98
The Super Mario Bros Movie	1	1	1	1	1	1			0,78	0,8	0,82		0,77	0,79	0,8	0,90
Happy Feet	1		0,67	0,75			0,57		0,56		0,55	0,58	0,62			0,66
Rise of the Guardians	1			0,5	0,6	0,67	0,71	0,75	0,78	0,8	0,82	0,83		0,79	0,8	0,75
Tangled	1			0,5	0,6	0,67	0,71	0,75		0,7		0,67	0,69	0,71		0,70
MAP																0,848

Tabel 4.9 Detail perhitungan MAP TF-IDF pada *rank-k* = 10

Query	**Nilai presisi pada tiap *rank-k***										**AP**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	
Kubo and The Two Strings	1		0,67						0,33	0,4	0,60
Cars	1	1	1	1	1	1	1	1	1	1	1
Toy Story	1	1	1	1	1	1	1	1	1	1	1
The Boss Baby	1	1		0,75	0,8	0,83	0,86		0,78	0,8	0,85
Ratatouille	1	1	1	1	1	1	1	1	1	1	1
Spirited Away		0,5	0,67		0,6	0,67	0,71		0,67	0,7	0,64
Your Name	1	1	1	1	1		0,86	0,86	0,89		0,95
Puss in Boots: The Last Wish	1	1	1	1	1	1	1	1	1	1	1
Cloudy with a chance	1	1			0,6	0,67	0,71	0,75	0,78	0,8	0,79
SpongeBob Movie: Sponge on the Run	1	1	1	1	1	1	1	1	1		1
The Secret Life of Pets	1	1	1	1	1	1	1	1	1	1	1
The Super Mario Bros Movie	1	1	1	1	1	1			0,78	0,8	0,95
Happy Feet	1		0,67	0,75			0,57		0,56		0,71
Rise of the Guardians	1			0,5	0,6	0,67	0,71	0,75	0,78	0,8	0,73
Tangled	1			0,5	0,6	0,67	0,71	0,75		0,7	0,70
											0,862

Pada tabel 4.8, dengan *rank-k* = 15 nilai MAP keseluruhan metode TF-IDF tanpa SVD adalah 0.848. Mayoritas *query* memiliki nilai AP yang tinggi dengan empat *query* yang memiliki nilai sempurna (1). Akan tetapi nilai AP metode TF-IDF pada *rank-k*=15 cenderung tidak stabil, di mana *query* "Kubo and The Two Strings" memiliki nilai yang rendah dibandingkan dengan *query* lainnya. Sehingga membuat nilai MAP nya sedikit lebih turun. Pada tabel 4.9, dengan *rank-k*=10 nilai MAP keseluruhan metode TF-IDF tanpa SVD adalah 0.862. Nilai ini sedikit lebih tinggi daripada pada *rank-k* 10. Hasil pada *rank-k* 10 juga menunjukkan konsistensi yang lebih baik daripada *rank-k* 15, karena nilai dari *query* yang sebelumnya rendah sedikit naik.

Perbandingan nilai MAP berdasarkan *genre* antara metode TF-IDF dengan SVD dan metode TF-IDF tanpa SVD dapat dilihat pada gambar 4.16.



Gambar 4.16 Grafik perbandingan nilai MAP metode TF-IDF dengan SVD dan TF-IDF tanpa SVD

Setelah melakukan uji coba dengan mencari nilai MAP berdasarkan *genre*, dapat disimpulkan metode TF-IDF yang dikombinasikan dengan SVD secara konsisten memiliki kinerja yang lebih baik di kedua peringkat *rank-k* 15 dan *rank-k* 10. Pada *rank-k* 15 TF-IDF dengan SVD mendapatkan nilai 0.850. Nilai tersebut lebih besar 0.02 dibandingkan dengan TF-IDF tanpa SVD dengan nilai 0.848. Hal ini juga berbanding lurus pada *rank-k* 10, di mana metode TF-IDF dengan SVD mendapatkan nilai 0.879, lebih besar 0,017 dibandingkan TF-IDF tanpa SVD yang mendapatkan nilai 0.862. Jika dirata-rata antara *rank-k* 10 dan *rank-k* 15 pada kedua metode, didapatkan nilai MAP TF-IDF dengan SVD adalah 0.865. Sementara TF-IDF tanpa SVD mendapatkan nilai rata-rata 0.855. Rata-rata nilai tersebut menunjukkan adanya kenaikan kinerja oleh TF-IDF dengan SVD dibandingkan TF-IDF tanpa SVD walaupun tidak signifikan. Dari perhitungan uji coba metode TF-IDF dengan SVD juga didapatkan nilai MAP pada *rank-k* 10 lebih besar daripada nilai MAP pada *rank-k* 15 yang menunjukkan metode TF-IDF dengan SVD lebih efektif dalam menempatkan hasil yang paling relevan di posisi teratas.

Kemudian untuk hasil rekomendasi berdasarkan kata kunci, metode *content-based filtering* menggunakan kombinasi TF-IDF dan SVD bisa memberikan hasil yang lebih bervariasi daripada tanpa menggunakan SVD seperti pada Tabel 4.10.

Tabel 4.10 Perbandingan hasil rekomendasi berdasarkan kata kunci.

Rekomendasi ke-	**Nilai Similarity**	
	TF-IDF.SVD	**TF-IDF**
1	0.925498	0.2497
2	0.567964	0
3	0.219316	0
4	0.207145	0
5	0.19336	0
6	0.15195	0

Rekomendasi ke-	Nilai Similarity	
	TF-IDF.SVD	TF-IDF
7	0.147417	0
8	0.138522	0
9	0.131947	0
10	0.131827	0

Pada Tabel 4.10, terlihat bahwa metode *content-based filtering* dengan menggunakan algoritma SVD menghasilkan rekomendasi yang lebih bervariasi dibandingkan dengan TF-IDF tanpa SVD. Menggunakan kata kunci "tacodiles and shrimpanzees", TF-IDF tanpa SVD hanya dapat memberikan rekomendasi yang spesifik pada film animasi "Cloudy with a Chance of Meatballs 2", karena kedua kata tersebut secara eksplisit disebutkan dalam korpus film tersebut. Sebaliknya, metode TF-IDF dengan SVD mampu mengenali hubungan semantik antara kata-kata dan memberikan berbagai rekomendasi berdasarkan kesamaan semantik. Hal ini memungkinkan metode TF-IDF dengan SVD untuk merekomendasikan film animasi seperti "Cloudy with a Chance of Meatballs" (film yang direkomendasikan metode TF-IDF dengan SVD pada rekomendasi kedua) yang memiliki hubungan semantik dengan "Cloudy with a Chance of Meatballs 2" dan mencakup konten yang mungkin tidak secara eksplisit menyebutkan kata kunci tetapi memiliki relevansi konseptual.

4.10 Integrasi dengan Islam

Al-Quran tidak secara langsung menjelaskan tentang sistem rekomendasi, Akan tetapi Al-Qur'an menjelaskan tentang konsep saling menolong dalam kebaikan yang merupakan suatu prinsip berkembangnya sistem rekomendasi. Al-Qur'an dengan jelas menyebutkan prinsip-prinsip pada sistem rekomendasi, salah

satu ayat yang dapat dihubungkan dengan sistem rekomendasi adalah ayat tentang tolong menolong dalam kebaikan. Allah *Ta'ala* berfirman dalam surat Al-Maidah ayat 2 yang berbunyi.

يَٰٓأَيُّهَا ٱلَّذِينَ ءَامَنُواْ لَا تُحِلُّواْ شَعَٰٓئِرَ ٱللَّهِ وَلَا ٱلشَّهۡرَ ٱلۡحَرَامَ وَلَا ٱلۡهَدۡيَ وَلَا ٱلۡقَلَٰٓئِدَ وَلَآ ءَآمِّينَ ٱلۡبَيۡتَ ٱلۡحَرَامَ يَبۡتَغُونَ فَضۡلٗا مِّن رَّبِّهِمۡ وَرِضۡوَٰنٗاۚ وَإِذَا حَلَلۡتُمۡ فَٱصۡطَادُواْۚ وَلَا يَجۡرِمَنَّكُمۡ شَنَـَٔانُ قَوۡمٍ أَن صَدُّوكُمۡ عَنِ ٱلۡمَسۡجِدِ ٱلۡحَرَامِ أَن تَعۡتَدُواْۘ وَتَعَاوَنُواْ عَلَى ٱلۡبِرِّ وَٱلتَّقۡوَىٰۖ وَلَا تَعَاوَنُواْ عَلَى ٱلۡإِثۡمِ وَٱلۡعُدۡوَٰنِۚ وَٱتَّقُواْ ٱللَّهَۖ إِنَّ ٱللَّهَ شَدِيدُ ٱلۡعِقَابِ

"Wahai orang-orang yang beriman! Janganlah kamu melanggar syi'ar-syi'ar kesucian Allah, dan jangan (melanggar kehormatan) bulan-bulan haram, jangan (mengganggu) hadyu (hewan-hewan kurban), dan Qalaid (hewan-hewan kurban yang diberi tanda), dan jangan (pula) mengganggu orang-orang yang mengunjungi Baitulharam; mereka mencari karunia dan keridaan Tuhannya. Tetapi apabila kamu telah menyelesaikan ihram, maka bolehlah kamu berburu. Jangan sampai kebencian(mu) kepada suatu kaum karena mereka menghalang-halangimu dari Masjidilharam mendorongmu berbuat melampaui batas (kepada mereka). Dan tolong-menolonglah kamu dalam (mengerjakan) kebajikan dan takwa, dan jangan tolong-menolong dalam berbuat dosa dan permusuhan. Bertakwalah kepada Allah, sungguh, Allah sangat berat siksa-Nya." (QS. Al-Maidah [5]:2)

Tafsir Al-Misbah karya Dr. Quraish Shihab menjelaskan bahwa ayat ini mengajarkan prinsip dasar dalam menjalin kerja sama. Umat Islam diajarkan untuk bekerja sama dengan siapa saja, asalkan tujuannya adalah kebajikan dan ketakwaan kepada Allah *Subhanahu Wa Ta'ala*. Ayat ini juga menekankan pentingnya keadilan bagi semua orang. Allah *Subhanahu Wa Ta'ala* memerintahkan umat Muslim untuk saling membantu dalam kebaikan, yang mencakup segala sesuatu yang bermanfaat bagi dunia dan akhirat, meskipun dengan orang yang tidak seiman. Sebaliknya, Allah *Subhanahu Wa Ta'ala* melarang untuk bekerja sama dalam perbuatan dosa dan pelanggaran. Allah *Subhanahu Wa Ta'ala* juga melarang menebar kebencian, termasuk kepada orang-orang pembenci (Shihab, 2005).

Ayat 02 surat Al-Maidah dalam Al-Quran menitikberatkan pada etika muamalah dalam Islam dengan menekankan prinsip-prinsip moral yang mendasari interaksi sosial. Umat Islam diajak untuk saling membantu dalam melakukan kebajikan dan menjaga ketakwaan, yang menggarisbawahi pentingnya kebaikan dan kesalehan sebagai dasar muamalah. Larangan terlibat dalam perbuatan dosa dan permusuhan mengarahkan umat untuk menjauhi perilaku negatif serta merawat keharmonisan hubungan antar individu. Puncaknya, muamalah Islam didorong oleh nasihat untuk bertakwa kepada Allah *Subhanahu Wa Ta'ala*, mengingatkan setiap tindakan dan interaksi dalam kerangka ketaatan kepada-Nya. Kesadaran akan beratnya siksaan Allah *Subhanahu Wa Ta'ala* memberikan dimensi moral yang kuat, mendorong individu untuk membangun muamalah yang adil, penuh kasih, dan berlandaskan pada nilai-nilai keislaman. Dengan demikian, ayat-ayat ini membentuk landasan etika sosial dalam membantu sesama umat manusia.

Imam Ibnu Qoyyim mendefinisikan bahwa kata *al-birru* mencakup semua jenis kebaikan dan kesempurnaan yang dituntut dari seorang hamba. Definisi tersebut dikuatkan oleh Syaikh as-Sa'di, yaitu seorang ahli bahasa arab yang mendefinisikan bahwa kata *al-birru* merupakan sebuah nama yang mencakup segala yang Allah *Subhanahu Wa Ta'ala* cintai dan ridhoi, baik bersifat *zhahir* dan *batin* yang berhubungan dengan hak Allah *Subhanahu Wa Ta'ala* dan hak sesama manusia. Sebagai salah satu contoh menolong dalam kebaikan, Rasulullah *shallalahu 'alaihi wasallam* bersabda :

انْصُرْ أَخَاكَ ظَالِمًا أَوْ مَظْلُومًا قَالُوا يَا رَسُولَ اللَّهِ هَذَا نَنْصُرُهُ مَظْلُومًا فَكَيْفَ نَنْصُرُهُ ظَالِمًا قَالَ تَأْخُذُ فَوْقَ يَدَيْهِ

"Bantulah saudaramu, dalam keadaan sedang berbuat zhalim ataupun di zhalimi. Ada yang mengatakan "Wahai Rasulullah, kami akan menolong orang yang terzhalimi ini. Bagaimana menolong orang yang berbut zhalim?" Beliau menjawab: "Dengam menghalanginya melakukan kezhaliman. Itulah bentuk bantuanmu kepadanya". (HR. Bukhari No. 2444)

Dalam hadis lain, Rasulullah *shallalahu 'alaihi wasallam* bersabda:

الدَّالُ عَلَى الْخَيْرِ كَفَاعِلِهِ

"Orang yang menunjukkan kepada kebaikan, ia seperti mengerjakannya" (HR. Tirmidzi No. 2670)

Tidak ada batasan kemampuan dalam menolong kepada kebaikan. Seorang muslim diminta untuk menolong orang lain sesuai dengan kemampuannya. Orang berilmu membantu dengan ilmu dan orang kaya membantu dengan hartanya. Selain itu, seorang muslim juga diminta untuk tolong menolong dengan ikhlas, cinta dan penuh perhatian kepada Allah *Subhanahu Wa Ta'ala* berdasarkan perintah-Nya "*Dan bertakwalah kepada Allah*" (As-Sa'di, 1997). Berhubungan dengan penjelasan Al-Maidah ayat 2 di atas, sistem rekomendasi berguna untuk merekomendasikan item kepada pengguna yang mana ini merupakan salah satu bentuk tolong menolong sesama manusia. Dengan ini diharapkan penelitian ini, sistem rekomendasi film animasi dapat memberikan rekomendasi film sesuai keinginan pengguna dan memberikan bantuan bagi orang yang membutuhkan.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan penelitian yang telah dijabarkan sebelumnya, penulis mendapatkan beberapa kesimpulan :

1. Setelah melakukan uji coba dengan mencari nilai MAP berdasarkan genre, dapat disimpulkan bahwa metode TF-IDF yang dikombinasikan dengan SVD menunjukkan kinerja yang lebih baik dibandingkan TF-IDF tanpa SVD pada kedua peringkat *rank-k* 15 dan *rank-k* 10. Pada *rank-k* 15, TF-IDF dengan SVD mendapatkan nilai 0.850, yang lebih tinggi 0.02 dibandingkan TF-IDF tanpa SVD dengan nilai 0.848. Demikian pula pada *rank-k* 10, TF-IDF dengan SVD meraih nilai 0.879, lebih besar 0.017 dibandingkan TF-IDF tanpa SVD yang mendapatkan nilai 0.862. Rata-rata nilai MAP antara *rank-k* 10 dan *rank-k* 15 untuk TF-IDF dengan SVD adalah 0.865, sementara TF-IDF tanpa SVD memperoleh rata-rata 0.855. Meskipun peningkatan kinerja TF-IDF dengan SVD tidak signifikan, hasil ini menunjukkan bahwa SVD dapat meningkatkan efektivitas dalam menempatkan hasil yang paling relevan di posisi teratas, terutama terlihat dari nilai MAP pada *rank-k* 10 yang lebih tinggi dibandingkan *rank-k* 15.
2. Hasil rekomendasi dari metode TF-IDF.SVD bisa jadi lebih bervariasi pada rekomendasi berdasarkan kata kunci dibandingkan metode TF-IDF tanpa SVD.

5.2 Saran

Berdasarkan penelitian yang telah dilakukan, berikut merupakan beberapa saran untuk penelitian yang akan datang.

1. Menggunakan metode lain untuk dekomposisi matriks seperti *Principal Content Analys* (PCA)
2. Mencoba nilai *n_component* lainnya pada proses *singular value decomposition*.
3. Mencoba menggunakan fitur yang lebih banya.
4. Penggunaan lebih banyak dokumen akan menyebabkan bertambahnya waktu komputasi sehingga diperlukan pengembangan waktu komputasi yang cepat

DAFTAR PUSTAKA

Aamir, M., & Bhusry, M. (2015). Recommendation System: State of the Art Approach. *International Journal of Computer Applications*, *120*(12), 25–32. https://doi.org/10.5120/21281-4200

AINAKI. (2020). *2020 Indonesia Animation Report*. https://ainaki.or.id/indonesia-animation-report-2020/

Arfisko, H. H., & Wibowo, A. T. (2022). Sistem Rekomendasi Film Menggunakan Metode Hybrid Collaborative Filtering Dan Content-Based Filtering. *E-Proceeding of Engineering*, *9*(3), 2149–2159.

As-Sa'di, 'Abdur Rahmân. (1997). *Taisîrul Karîmir Rahmân*. Muassasah Risâlah.

Billah, tashim, Aidil Zartesya, M., Sandya Prasvita, D., Komp, S., Kom, M., Komputer Jl Simatupang, I. T., Timur, C., & Selatan, J. (2021). Penerapan Collaborative Filtering, PCA dan K-Means dalam Pembangunan Sistem Rekomendasi Film. In *Seminar Nasional Mahasiswa Ilmu Komputer dan Aplikasinya (SENAMIKA) Jakarta-Indonesia*.

Cheryl, A. M. (2022). *Sistem Rekomendasi Hotel Dengan Pendekatan Content-Based Filtering*.

Firmansyah, A., & Kurniawan, M. P. (2013). Pembuatan Film Animasi 2D Menggunakan Metode Frame by Frame Berjudul "Kancil dan Siput." *Jurnal Ilmiah DASI*, *14*(04), 10–13.

Fitrianti, A. R., Rohmani, A., & Widjanarto, W. (2020). Sistem Rekomendasi Film Berbasis Website Dengan Metode Prototype Menggunakan Algoritma K-Nearest Neighbors (KNN). *JOINS (Journal of Information System)*, *5*(2), 278–287. https://doi.org/10.33633/joins.v5i2.4168

Gupta, V., Joshi, N., & Vidyapith, B. (2015). Design & Development of a Rule Based Urdu Lemmatizer. *1st International Conference on Futuristic Trend in Computational Analysis & Knowledge Management, IEEE, July*.

Hanifah, A. W. (2023). Sistem Rekomendasi Pemilihan Susu Formula Menggunakan User-Based Collaborative Filtering. In *Skripsi*. http://etheses.uin-malang.ac.id/53662/

Hasan, M. I. (2018). *Information Retrieval System artikel kesehatan menggunakan pembobotan tf. idf dan Latent Semantic Indexing*. http://etheses.uin-malang.ac.id/12546/

Herny Februariyanti, Aryo Dwi Laksono, Jati Sasongko Wibowo, M. S. U. (2021). Implementasi Metode Collaborative Filtering Untuk Sistem Rekomendasi Penjualan Pada Toko Mebel. *Jurnal Khatulistiwa Informatika*, *IX*(I), 43–50. www.unisbank.ac.id

Hiro Juni Permana, A., & Toto Wibowo, A. (2023). Movie Recommendation System Based on Synopsis Using Content-Based Filtering with TF-IDF and Cosine Similarity. *Intl. Journal on ICT*, *9*(2), 1–14. https://doi.org/10.21108/ijoict.v9i2.747

Ikhsani Suwandy Futri, N., & Nuraini Siti Fathonah, R. (2023). Penerapan Metode Content Based Filtering Dan Knn Pada Aplikasi Rekomendasi Laptop Berbasis Mobile. *JATI (Jurnal Mahasiswa Teknik Informatika)*, *7*(2), 1229–1236. https://doi.org/10.36040/jati.v7i2.6724

Jurafsky, D., & Martin, J. H. (2023). Speech and Laguage Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition. In *Stanford University* (Third Edit). Stanford University. https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf

Kadam, N., & Kumar, S. (2016). A review of Content and Collaborative filtering approaches on Movielens Data. *International Research Journal of Engineering and Technology*, 273–278. www.irjet.net

Kurnia, F. (2020). *Animation: Definition, Types and Examples, and Benefits*. https://en.dailysocial.id/post/animation-adalah

Manning, C. D., Raghavan, P., & Schütze, H. (2009). An Introduction to Information Retrieval. In *Cambridge University Press* (Online Edt).

Miesle, P. (2023). *What is Cosine Similarity: A Comprehensive Guide*. Datastax. https://www.datastax.com/guides/what-is-cosine-similarity

Muhadzdzib Ramadhan, M. T., & Setiawan, E. B. (2022). Netflix Movie Recommendation System Using Collaborative Filtering With K-Means Clustering Method on Twitter. *Jurnal Media Informatika Budidarma*, *6*(4), 2056. https://doi.org/10.30865/mib.v6i4.4571

Munir. (2012). Multimedia konsep dan aplikasi dalam pendidikan. In *Alfabeta* (Vol. 58, Issue 12).

Nisrina, F. (2020). *Implementasi Deteksi Topik Putusan Hakim Dengan Latent Dirichlet Allocation (Lda)* [Universitas Islam Indonesia]. https://dspace.uii.ac.id/handle/123456789/23847

Nugraha, D., Purboyo, T. W., & Nugrahaeni, R. A. (2021). Sistem Rekomendasi Film Menggunakan Metode User Based Collaborative Filtering. *E-Proceeding of Engineering*, *8*(5), 6765–6775. https://ejournal.upi.edu/index.php/JATIKOM

Nuha, A. S. (2020). Implementation of the Cosine Similarity Method Using Thesaurus in the Search Application of English Al-Quran Translation. In *Skripsi*.

Pratondo, D. A. (2023). *Pengembangan Sistem Rekomendasi Berbasis Content-based Filtering Pada Data Dinamis* [Universitas Islam Negeri Syarif

Hidayatullah Jakarta]. https://repository.uinjkt.ac.id/dspace/handle/123456789/66813

Ricci, F., Rokach, L., Shapira, B., Kantor, P. B., & Ricci, F. (2011). Recommender Systems Handbook. In *Recommender Systems Handbook*. https://doi.org/10.1007/978-0-387-85820-3

Shah, L., Gaudani, H., & Badani, P. (2016). Survey on Recommendation Systems. *International Journal of Computer Applications*, *137*(7), 1–7. https://doi.org/10.1145/3447568.3448518

Shihab, M. Q. (2005). *Tafsir Al-Msibah* (IV). Lentera Hati.

Singla, R., Gupta, S., Gupta, A., & Vishwakarma, D. K. (2020). FLEX: A content based movie recommender. *2020 International Conference for Emerging Technology, INCET 2020*, 8–11. https://doi.org/10.1109/INCET49848.2020.9154163

Siringoringo, R., Jamaluddin, J., & Lumbantoruan, G. (2021). Sistem Perekomendasi Dengan Singular Value Decomposition Dan Teknik Similaritas Pearson Correlation. *METHODIKA: Jurnal Teknik Informatika Dan Sistem Informasi*, *7*(1), 19–24. https://doi.org/10.46880/mtk.v7i1.257

Sorde, R. K., & Deshmukh, S. N. (2015). Comparative Study on Approaches of Recommendation Systems. *International Journal of Computer Applications*, *118*(2), 10–14. https://doi.org/10.1007/978-981-15-0947-6_72

Syamil, A. (2023). *Makna Hadits Sampaikan Dariku Walau Satu Ayat, Ballighu 'Anni walau Ayah*. 9 May 2023. https://www.risalahislam.com/2023/05/makna-hadits-sampaikan-dariku-walau.html

Unnikrishnan, C. S. (2021). *How sklearn's Tfidfvectorizer Calculates tf-idf Values*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/11/how-sklearns-tfidfvectorizer-calculates-tf-idf-values/

Wahyuni, R. T., Prastiyanto, D., & Supraptono, E. (2017). Penerapan Algoritma Cosine Similarity dan Pembobotan TF-IDF pada Sistem Klasifikasi Dokumen Skripsi. *Jurnal Teknik Elektro Universitas Negeri Semarang*, *9*(1), 18–23. https://journal.unnes.ac.id/nju/index.php/jte/article/download/10955/6659