BAB I
PENDAHULUAN

1.1 Latar Belakang

Sistem rekomendasi merupakan alat yang krusial dalam aktivitas berinternet, contohnya pada platform-platform *streaming* film seperti Netflix yang mengarahkan pengguna untuk menemukan informasi yang sesuai dengan preferensi mereka. Sistem rekomendasi dapat meningkatkan pengalaman pengguna secara signifikan dengan merekomendasikan film yang relevan dengan tiap pengguna. Sistem rekomendasi yang berkualitas dapat meningkatkan kepuasan pengguna terhadap platform dan meningkatkan loyalitas pengguna pada platform yang mana sangat penting untuk pertumbuhan bisnis dalam lingkup industri hiburan digital. Namun, dalam situasi di mana pengguna baru tidak memiliki rekam jejak interaksi maka masalah *cold-start* dapat terjadi pada sistem, hal ini masih menjadi tantangan utama sehingga penerapan analisis sentimen dalam bidang *Natural Language Processing* (NLP), seperti yang digunakan dalam penelitian ini, menjadi solusi untuk meningkatkan kualitas proses rekomendasi (Panda, et al., 2022; Rodpysh, et al., 2023).

Membeludaknya jumlah konten hiburan linear dengan semakin banyaknya platform-paltform *streaming* film, hal ini menyebabkan kebutuhan akan sistem rekomendasi yang akurat dalam meberikan hasil rekomendasi yang relevan menjadi semakin mendesak. Miliaran pengguna meninggalkan jejak digital dalam jumlah besar sehingga sistem rekomendasi tradisional yang umumnya bergantung pada informasi eksplisit seperti *rating* atau jumlah "*likes*" tidak lagi dapat menyelesaikan masalah *cold-start* yang muncul ketika pengguna baru tidak memiliki riwayat penilaian yang dapat dianalisis oleh sistem. Ketidakmampuan sistem untuk merekomendasikan film yang relevan dapat menyebabkan ketidakpuasan pengguna dan berpotensi hilangnya loyalitas pengguna. Dalam ruang lingkup hiburan digital yang sangat kompetitif, pengembangan sistem yang lebih mumpuni untuk memberikan rekomendasi film yang lebih baik, terutama bagi pengguna baru, menjadi prioritas (Järvelin, et al., 2002; Nakhli, et al., 2019).

Novelty dari penelitian ini terletak pada penggabungan BERT dan CNN sebagai model analisis sentimen dan penggunaan BERT sebagai penyematan kata (*word embedding*) bagi metrik ukur *Cosine Similarity* dalam sistem rekomendasi film. Penerapan BERT telah terbukti berhasil dalam penugasan-penugasan NLP, tetapi potensinya dalam sistem rekomendasi untuk film masih belum banyak dieksplorasi. *Cosine Similarity* adalah metrik ukur yang digunakan untuk menghitung kesamaan antara dua vektor dan apabila vektor-vektor hasil penyematan kata BERT yang kaya akan makna kontekstualnya dihitung menggunakan *Cosine Similarity* dalam konteks rekomendasi film maka menghasilkan pendekatan yang inovatif. Penelitian ini menggunakan metode *scenario-based* yang mampu beroperasi secara efektif ketika

data interaksi pengguna terbatas, memberikan solusi baru untuk masalah *cold-start* pada rekomendasi film.

Penerapan sistem rekomendasi film menggunakan sentimen pengguna ini memungkinkan untuk dilakukan karena majunya tekonlogi *machine learning* masa kini, contohnya seperti model *pre-trained* BERT. Model ini dapat disesuaikan lebih lanjut (*fine-tuned*) untuk penugasan-penugasan spesifik seperti klasifikasi sentimen, contohnya pada klasifikasi sentimen ulasan film. Penerapan ini relevan dalam industri *streaming* di mana ulasan pengguna dan kritik dari kritikus film dapat mempengaruhi pengguna baru dalam menentukan tontonan. Kemampuan untuk menilai sentimen di balik ulasan yang dihasilkan pengguna memberikan platform wawasan yang lebih mendalam tentang preferensi pelanggan, sehingga meningkatkan kualitas rekomendasi film. Penggunaan penyematan kata BERT memperkuat kemampuan sistem untuk memahami makna kontekstual yang lebih dalam dari ulasan, menghasilkan rekomendasi film yang lebih akurat dan relevan.

Penanganan masalah *cold-start* melalui analisis sentimen ulasan film dan penggunakan metrik ukur *Cosine Similarity* memiliki potensi dalam menciptakan sistem rekomendasi yang lebih adaptif. Penelitian ini berfokus untuk meningkatkan akurasi rekomendasi film dalam situasi di mana metode tradisional yang bergantung pada *rating* dan *likes* tidak efektif. Pendekatan yang diusulkan ini mampu mengubah dinamika sistem rekomendasi dalam layanan streaming, membuat platform lebih adaptif dan efisien dalam memberi rekomendasi yang relevan kepada pengguna. Potensi ini menjadikan penelitian ini dibutuhkan penerapannya pada berbagai platform *streaming* modern.

Tujuan penelitian ini adalah untuk mengembangkan sistem rekomendasi film yang mampu mengatasi masalah *cold-start* melalui integrasi analisis sentimen dan *Cosine Similarity*. Sistem ini dirancang untuk meningkatkan akurasi rekomendasi dalam dua skenario berbeda: ketika data interaksi pengguna tidak tersedia, rekomendasi dihasilkan berdasarkan analisis sentimen ulasan film; dan ketika data interaksi tersedia, *Cosine Similarity* diterapkan untuk meningkatkan ketepatan rekomendasi dengan membandingkan riwayat interaksi pengguna dengan informasi film. Pendekatan ini diharapkan dapat memberikan rekomendasi film yang lebih personal dan relevan, sehingga meningkatkan kepuasan pengguna dan loyalitas pada platform.

1.2 Landasan Teori

1.2.1 Sistem Rekomendasi

Dalam penelitian yang dilakukan oleh Liu dan rekan-rekannya (2022), tidak memadainya pemanfaatan umpan balik eksplisit dan implisit dalam sistem rekomendasi karena model yang ada biasanya hanya berfokus pada data implisit yang biner sehingga mengabaikan fitur data lain yang lebih kaya. Peneliti menggunakan metode model-based dan algoritma usulan mereka yaitu *Explicit-Implicit Neural Matrix Factorization Recommendation* (EINMFR) dengan

menggunakan data rating penonton sebagai indikator preferensi dan mendapatkan nilai *Hit Rate* (HR) sebesar ±0.86 untuk menghasilkan top-20 rekomendasi dengan menggunakan dataset MovieLens-1M.

Dalam penelitian yang dilakukan oleh Ahmadian dan rekan-rekannya (2020), sparsitas data merupakan keterbatasan pada sistem rekomendasi yang disebabkan oleh kurangnya data rating atau koneksi sosial (contohnya dalam platform media sosial) sehingga diusunglah metode *Social Recommender based on Reliable Implicit Relationships* (SoRIR) untuk mengatasi masalah ini. SoRIR dibangun dengan mengintegrasikan data eksplisit berupa rating pengguna terhadap item dan koneksi pengguna dengan pengguna lain kemudian memprediksi hubungan implisit berdasarkan indikator tidak langsung dari data eksplisit dengan menggunakan algoritma *Dempster-Shafer Theory*. Model pada penelitian ini mendapatkan nilai MAE sebesar ±0.55, RMSE sebesar ±0.87, *Precision* sebesar ±0.41, *Recall* sebesar ±0.65, *Coverage* (CC) sebesar ±0.51, *Diversity* sebesar ±0.49 dan *Novelty* sebesar ±8.41.

Dalam penelitian yang dilakukan oleh Nan dan rekan-rekannya (2022), informasi yang berlebihan menjadi masalah bagi pengguna internet sehingga dibutuhkannya sistem rekomendasi yang berlandaskan data mining dengan tujuan untu menyediakan rekomendasi yang relevan dan paham konteks (*context-aware*). Sistem rekomendasi yang diusung dinamakan *Collaborative Mining and Filtering Process* (CMFP), sistem ini menggunakan metode *collaborative filtering* dan gabungan algoritma *Support Vector Regression* (SVR) dan *Multilayer Perceptron* (MLP). Model pada penelitian ini mendapatkan nilai akurasi ±9% lebih tinggi dari model dalam penelitian sebelumnya.

1.2.2 Analisis Sentimen

Dalam penelitian yang dilakukan oleh Liu (2022), penonton umumnya mengekspresikan aspek-aspek yang mereka sukai dari sebuah film ke dalam ulasan film dan aspek-aspek ini biasanya merefleksikan potensi preferensi tontonan penonton. *Rating* hanya dapat memberikan indikasi apakah penonoton suka atau tidak suka dengan tontonannya dan tidak memiliki informasi spesifik alasan mengapa penonton bisa suka atau tidak suka dengan tontonannya. Penelitian ini menggunakan metode *collaborative filtering* dan algoritma *Non-negative Matrix Factorization* (NMF) dan mendapatkan hasil nilai MAE sebesar ±0.76 dan *recall* sebesar ±0.04 pada top-5 rekomendasi.

Dalam penelitian yang dilakukan oleh Latha dan Rao (2024), metode sistem rekomendasi konvensional seperti *collaborative filtering* memiliki masalah sparsitas data, sedangkan *content-based filtering* membutuhkan sumber daya komputasi yang mahal sedangkan metode *hybrid* memiliki karakteristik yang kompleks sehingga diusunglah model gabungan *Convolutional Neural Network* (CNN) dan *Singular Value Decomposition* (SVD) yang dinamakan *Modified Convolutional Neural Network* (MCNN). Peneliti mengklaim berhasil mendapatkan nilai akurasi sebesar ±94%.

Dalam penelitian yang dilakukan oleh Amangeldi dan rekan-rekannya (2024), media sosial telah menjadi sumber daya sentimen publik. Sentimen publik dari sosial media bisa digunakan untuk menyorot masalah global berupa perubahan iklim, kualitas air, emisi, plastik dan lain-lain. Model *Valance Aware Dictionary and Sentiment Reasoner* (VADER) dan *Pointwise Mutual Information* (PMI) digunakan untuk mendeteksi emosi yang terkandung dalam kalimat komentar pengguna sosial media diberbagai platform dan didapatkan hasil nilai akurasi masing-masing sebesar 0.64% dan 0.65%.

Dalam penelitian yang dilakukan oleh Rohalia (2023), opini masyarakat dapat digunakan sebagai umpan balik bagi Dinas Daerah Kabupaten/Kota untuk mengevaluasi kinerja pengimplementasian kebijakan pemerintah kota serta mengidentifikasi kekuatan atau kelemahan pemerintah kota. Peneliti melakukan analisis sentimen berupa keluhan dan tanggapan masyarakat terhadap 16 dinas terkait, sesuai dengan fungsi dan tugasnya. Dengan menggunakan kombinasi algoritma CNN dan *Random Forest*, peneliti mengklaim berhasil mencapai akurasi model sebesar 87%. Penelitian oleh Rohalia dijadikan sebagai pembanding dalam penelitian ini, di mana model CNN + *Random Forest* tersebut dilatih ulang menggunakan dataset dan jumlah klasifikasi yang sesuai dengan penelitian ini.

1.2.3 Metrik Ukur Kesamaan

Dalam penelitian yang dilakukan oleh Garigliotti dan rekan-rekannya (2023), tantangan dari sistem rekomendasi berbasis kueri adalah kurangnya data khusus yang dibutuhkan untuk pelatihan model sehingga metode *content-based filtering* digunakan untuk mengatasi masalah ini. Kalkulasi kesamaan semantik antara kueri dengan atribut data dilakukan dengan menggunakan *cosine similarity* dan algoritma perangkingan yang digunakan adalah BM25 (BM singkatan dari *Best Matching*) sehingga didapatkan nilai *Normalized Discountend Cumulative Gain* (NDCG) sebesar ±0.36 untuk judul dan ±0.29 untuk deskripsi.

Dalam penelitian yang dilakukan oleh Chia dan Najafabadi (2022), masalah *cold-start* kerap kali muncul pada sistem rekomendasi pada saat adanya inisialisasi area baru dikarenakan sistem belum memiliki data historis pengguna sehingga sistem tidak mampu menghasilkan rekomendasi. Oleh karena itu untuk mengatasi masalah tersebut penelitian ini menggunakan metode *content-based filtering* dengan metrik ukur *cosine similarity*, *Pearson correlation coefficient* dan *Euclidean distance* dan didapatkan hasil nilai *average of similarity* dari *cosine similarity* sebesar 81%, *Euclidean distance* sebesar 62% dan *Pearson correlation* sebesar 80% pada top-10 rekomendasi.

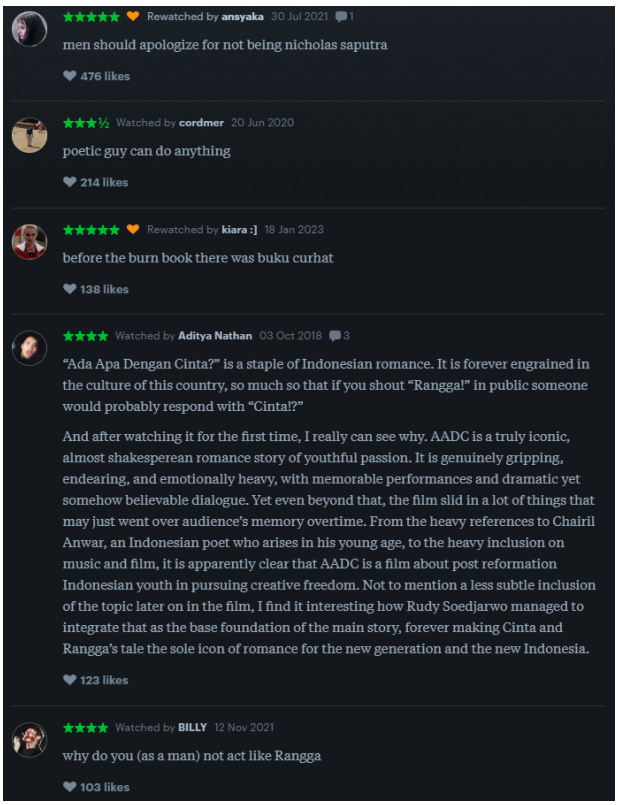
Dalam penelitian yang dilakukan oleh Artajaya dan rekan-rekannya (2024), tujuan utamanya adalah mengembangkan sistem rekomendasi magang yang presisi dengan mengekstraksi preferensi dari resume mahasiswa. Dataset yang telah di-*preprocessing* kemudian diverktorisasi menggunakan Word2Vec, di mana vektor-vektor dari data teks resume mahasiswa ini nantinya akan diukur jaraknya dengan data pekerjaan menggunakan *Euclidean distance*. Peneliti membandingkan metrik

Euclidean distance dengan beberapa metrik lain, termasuk *Cosine Similarity*, dan mendapati bahwa *Euclidean distance* memiliki keunggulan dengan nilai error rata-rata terendah, yaitu 0.051, sedangkan *Cosine Similarity* memiliki error rata-rata sebesar 0.26. Penelitian oleh H. Artajaya dan rekan-rekannya dijadikan sebagai pembanding dalam penelitian ini, di mana metrik *Euclidean distance* diimplementasikan ulang menggunakan dataset yang berbeda dan BERT sebagai metode vektorisasi kata, sesuai dengan penelitian ini.

1.2.4 Dataset

Dalam ilmu komputer, dataset adalah koleksi data yang diformat dan disusun secara terorganisir untuk mendukung analisis, pemrosesan, atau pembelajaran mesin. Dataset dapat terdiri dari berbagai jenis data, termasuk angka, teks, gambar, atau sinyal, yang semuanya dikumpulkan untuk tujuan penelitian, pengembangan algoritma, atau pemecahan masalah tertentu dalam domain komputasi.

Gambar-gambar di bawah ini merupakan cuplikan kecil dari dataset yang digunakan dalam penelitian ini, termasuk ulasan film dari platform database film LetterBoxd, dataset kritik film untuk pelatihan dan validasi model, dataset untuk pengujian, hingga dataset yang berisi informasi tentang film-film Indonesia.



Gambar 1. Ulasan Film AADC dari LetterBoxd

Dataset yang digunakan untuk pelatihan dan validasi model berasal dari kritik film di Rotten Tomatoes, yang dapat diunduh melalui Kaggle. Sebelum digunakan dalam model, kritik-kritik film tersebut perlu dianotasi terlebih dahulu. Proses anotasi dilakukan menggunakan RoBERTa, dan hasilnya disimpan dalam dataset pada kolom "*sentiment*".

	Id	rotten_tomatoes_link	critic_name	review_score	review_date	review_content	cleaned_review	sentiment
0	0	m/0814255	Andrew L. Urban	NaN	2010-02-06	A fantasy adventure that fuses Greek mythology...	a fantasy adventure that fuses greek mythology...	0
1	2	m/0814255	NaN	NaN	2010-02-09	With a top-notch cast and dazzling special eff...	with a topnotch cast and dazzling special effe...	0
2	3	m/0814255	Ben McEachen	3.5/5	2010-02-09	Whether audiences will get behind The Lightnin...	whether audiences will get behind the lightnin...	0
3	5	m/0814255	David Germain	NaN	2010-02-10	It's more a list of ingredients than a movie-m...	its more a list of ingredients than a moviemag...	0
4	7	m/0814255	Bill Goodykoontz	3.5/5	2010-02-10	Percy Jackson isn't a great movie, but it's a ...	percy jackson isnt a great movie but its a goo...	0

Gambar 2. Dataset Kritik Film dari *Rotten Tomatoes*

	Movie Title	Username	Full Review
0	What's Up with Cinta?	yves	men should apologize for not being nicholas sa...
1	What's Up with Cinta?	cordmer	poetic guy can do anything
2	What's Up with Cinta?	Aditya Nathan	"What's Up with Cinta?" is a staple of Indones...
3	What's Up with Cinta?	kiara :]	before the burn book there was diary book
4	What's Up with Cinta?	BILLY	why do you (as a man) not act like Rangga

Gambar 3. Dataset Ulasan Film Indonesia dari LetterBoxd

	Title	Synopsis	Cast	Director	Producers	Writers	Modified Genres
0	Whats Up with Cinta?	A popular high school girl strains her relatio...	Dian Sastrowardoyo, Nicholas Saputra, Ladya Ch...	Rudi Soedjarwo	Riri Riza, Mira Lesmana	Riri Riza, Mira Lesmana, Rako Prijanto, Jujur ...	romance, comedy, teens, drama
1	Whats Up with Cinta 2	14 years after their budding romance in high s...	Dian Sastrowardoyo, Nicholas Saputra, Adinia W...	Riri Riza	Mira Lesmana, Mandy Marahimin	Mira Lesmana, Prima Rusdi	romance, comedy, family, drama
2	Marlina the Murderer in Four Acts	After encountering a group of bandits with pla...	Marsha Timothy, Egy Fedly, Tumpal Tampubolon, ...	Mouly Surya	Fauzan Zidni, Rama Adi	Mouly Surya, Rama Adi	action, crime, gory, thriller, western, drama,...
3	Fiction.	When a sheltered young woman becomes enamored ...	Ladya Cheryl, Donny Alamsyah, Kinaryosih, Inon...	Mouly Surya	Rama Adi, Tia Hasibuan, Sapto Soetardjo	Joko Anwar, Mouly Surya	thriller, drama
4	Tjoet Nja Dhien	Set in 1896, "Tjoet Nja Dhien" celebrates one ...	Christine Hakim, Slamet Rahardjo, Piet Burnama...	Eros Djarot	Handi Muljono, Eros Djarot, Alwin Abdullah, Al...	None	history, action, drama

Gambar 4. Dataset Informasi Film Indonesia dari LetterBoxd

1.2.5 Fitur Penilaian

Sistem rekomendasi dalam penelitian ini menerapkan metode *scenario-based*. Metode *scenario-based* pada penelitian ini adalah sistem memberikan hasil rekomendasi berdasarkan dari situasi pengguna, skenario pertama menggunakan teknik klasifikasi sentimen, sementara skenario kedua mengandalkan teknik pengukuran kesamaan. Kedua teknik tersebut memiliki fitur penilaian yang berbeda.

Fitur penilaian untuk skenario pertama adalah polaritas ulasan, yaitu apakah ulasan tersebut berkonotasi positif, negatif, atau netral. Penelitian ini menggunakan RoBERTa untuk mengklasifikasikan polaritas dari kritik-kritik film sebagai *ground truth* dalam pelatihan dan evaluasi model. Agar memudahkan pemahaman, berikut adalah contoh dari 3 sentimen yang dinilai positif, negatif dan netral:

a. Positif

- *A fantasy adventure that fuses Greek mythology to contemporary American places and values. Anyone around 15 (give or take a couple of years) will thrill to the visual spectacle*. Kalimat ini memuji story-building dari film.
- *With a top-notch cast and dazzling special effects, this will tide the teens over until the next Harry Potter instalment*. Kalimat ini memuji penggunaan CGI pada film.
- *Whether audiences will get behind The Lightning Thief is hard to predict. Overall, it's an entertaining introduction to a promising new world -- but will the consuming shadow of Potter be too big to break free of?* Kalimat ini memuji alur cerita film yang cukup menghibur.
- *Percy Jackson isn't a great movie, but it's a good one, trotting out kernels of Greek mythology like so many Disney Channel references*. Kalimat ini memberikan pujian bahwa film tersebut bagus meski tidak sempurna.
- *Fun, brisk and imaginative*. Kalimat ini memuji atmosfer dari film.

b. Negatif

- *What's really lacking in The Lightning Thief is a genuine sense of wonder, the same thing that brings viewers back to Hogwarts over and over again*. Kalimat ini mengkritik bahwa film tersebut kurang imajinatif.
- *Harry Potter knockoffs don't come more transparent and slapdash than this wannabe-franchise jumpstarter directed by Chris Columbus.* Kalimat ini berkonotasi hinaan terhadap film.
- *This cast is simply too generic. None of the young thespians stick out*. Kalimat ini mengkritik pemain dari film tidak ada yang menarik.
- *A little bit worse than lifeless; it's clueless, like a Medusa running around with her head cut off*. Kalimat ini mengkritik bahwa film terasa tidak hidup dan tidak jelas arahnya.
- *As a whole the picture doesn't come off, partly because Quinn completely overshadows Alan Bates. Partly, too, because so much is unexplained*. Kalimat ini mengkritik bahwa storyline film memiliki banyak *plot-hole*.

c. Netral

- *Uma Thurman as Medusa, the gorgon with a coiffure of writhing snakes and stone-inducing hypnotic gaze is one of the highlights of this bewitching fantasy*. Kalimat ini mendeskripsikan karakter antagonis pada film.
- *Crammed with dragons, set-destroying fights and things exploding, [Columbus] squeezes in a few well-meaning pause breaks about friendship and absent fathers before swiftly moving on to the next pyrotechnics display.* Kalimat ini merangkum singkat isi dari film.
- *Admirably, the movie isn't bogged down in the exposition common to first entries, but at the end, it's a franchise searching for its voice*. Kalimat ini mengomentari soal franchise dari film.
- *When the movie slows down to catch its breath, there's very little of the heart and soul -- and wry wit -- that make the Riordan books so beloved*. Kalimat ini mengomentari novel yang menjadi inspirasi dari film tersebut.
- *Columbus aims at nothing more than providing a rambunctious romp. He nimbly avoids the fate that has snared other film adaptations, like Eragon and Inkheart.* Kalimat ini mengomentari sutradara yang karyanya berhasil terhindar dari kegagalan seperti film-film adaptasi novel yang lain

1.2.6 *Preprocessing* NLP

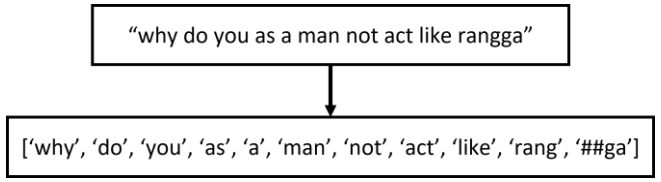
Data yang diambil dari website sering kali berbentuk kalimat yang tidak terstruktur, seperti yang mengandung karakter alfanumerik, URL, dan tata bahasa yang tidak konsisten. Ketidakteraturan dan *noise* dalam data tersebut dapat mempengaruhi proses pelatihan model. Oleh karena itu, preprocessing NLP dilakukan sebagai langkah untuk menghilangkan *noise* yang tidak diinginkan dari dataset, sehingga menghasilkan data yang lebih terstruktur. Berikut adalah tahapan-tahapan *preprocessing* tersebut:

a. *Character Removal*

Sebelum dataset diberi label dengan RoBERTa, data dibersihkan terlebih dahulu dari karakter yang tidak relevan, seperti URL, *hashtag*, *username*, tanda baca, dan karakter non-alfanumerik, seperti @, #, $, %, dan sebagainya. Karakter-karakter ini dapat mempengaruhi proses pelatihan model karena tidak menambah nilai pemahaman dan dapat menimbulkan *noise*.

b. *Tokenization*

Tokenisasi adalah proses mengubah rangkaian teks menjadi unit-unit yang lebih kecil dan mudah dikelola, yang disebut token. Token ini dapat berupa kata-kata, subkata, atau bahkan karakter individual, tergantung pada strategi tokenisasi yang digunakan. Tokenisasi bertujuan untuk mempersiapkan teks untuk pemrosesan lebih lanjut, seperti input ke dalam model pembelajaran mesin.



Gambar 5. Ilustrasi Proses *Tokenization*

Gambar 5 menunjukkan adanya simbol “##” pada token terakhir, yang mengindikasikan bahwa token tersebut merupakan subkata yang dipisahkan dari kata yang lebih besar selama proses tokenisasi. Dalam konteks ini, kata “rangga” dipisahkan menjadi “rang” dan “##ga” dalam tokenisasi BERT. Prefiks “##” menunjukkan bahwa token tersebut bukanlah kata utuh, melainkan kelanjutan dari token sebelumnya.

Teknik ini disebut WordPiece, yang berarti BERT tidak memerlukan proses stemming atau lemmatization. WordPiece membagi kata menjadi token subkata, sebuah pendekatan yang lebih fleksibel dibandingkan dengan stemming. Hal ini memungkinkan BERT untuk menangani kata-kata baru yang tidak terdapat dalam kosakatanya dan tetap dapat menangkap maknanya, seperti pada kata “rangga” yang ditunjukkan pada Gambar 5.

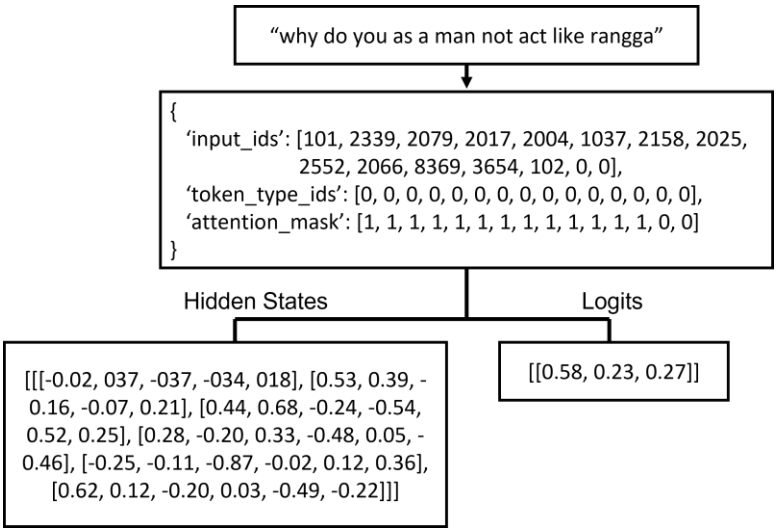
1.2.7 BERT *Vectors*

Vektorisasi teks secara umum merupakan proses mengubah teks menjadi representasi numerik, atau “vektor”, yang dapat dipahami oleh model pembelajaran mesin. Proses ini mengonversi teks tak terstruktur menjadi data numerik terstruktur dengan tujuan merepresentasikan makna semantik teks dalam format matematika.

Penelitian ini menggunakan dua varian dari penyematan kata BERT dan setiap metode menerapkan varian BERT yang berbeda. Metode pertama menggunakan TFBertForSequenceClassification, yang dirancang khusus untuk tugas klasifikasi, seperti analisis sentimen, dan cocok dengan model yang digunakan dalam metode ini. Sementara itu, metode kedua menggunakan varian BERT biasa.

TFBertForSequenceClassification adalah varian BERT yang dirancang secara khusus untuk memproses data sekuensial seperti pada penugasan analisis sentimen, penentuan jumlah kelas pada parameter model secara manual serta penggunaan dataset sentimen yang telah dianotasi untuk pelatihan dan evaluasi model disebut sebagai “*fine-tuning*”. Dalam model *Fine-Tuned* BERT + CNN ini, TFBertForSequenceClassification berperan dalam memberikan penyematan (*embedding*) kontekstual pada setiap token di seluruh urutan. Penyematan ini menangkap makna tiap kata dalam konteks keseluruhan kalimat dalam dokumen. Dalam BERT, penyematan kontekstual kata ini disebut hidden states. TFBertForSequenceClassification juga menghasilkan *logits* dari token [CLS], yaitu token khusus dalam BERT *tokenizer* yang ditambahkan di awal setiap urutan. *Logits* merupakan nilai prediksi mentah yang belum dinormalisasi untuk setiap kelas. Istilah “*logits*” berasal dari rumus fungsi *logit* dalam *logistic regression* yang dikembangkan

oleh David Cox pada (1958). *Multinomial Logistic Regression* (*softmax regression*) digunakan untuk tugas klasifikasi multi-kelas, di mana model menghasilkan nilai *logit* untuk tiap kelas dan nilai-nilai ini kemudian ditransformasikan menjadi probabilitas (Bishop, 2006). *Logits* inilah yang digunakan untuk klasifikasi. Gambar 6 menunjukkan ilustrasi hasil peyematan kata dari TFBertForSequenceClassification atau *fine-tuned* BERT dan juga nilai *logits* dari kalimat tersebut.

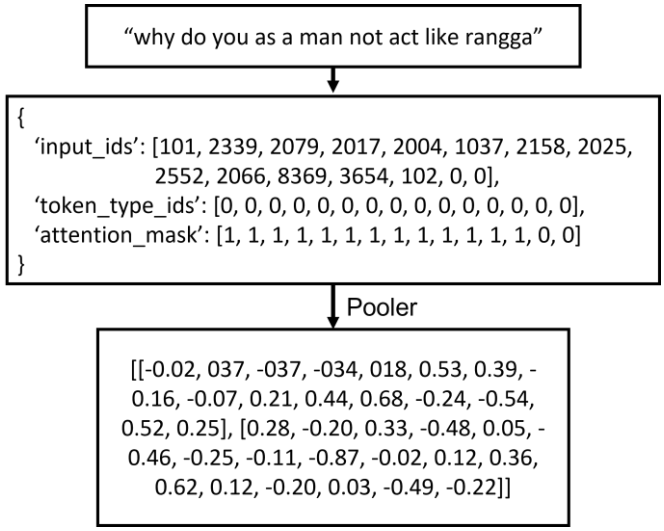


Gambar 6. Ilustrasi *Hidden States Fine-Tuned* BERT dan *Logits*

Ukuran penyematan kata pada BERT adalah 768, yang ditetapkan oleh para pengembangnya. Hal ini tercatat dalam penelitian oleh Devlin dan rekan-rekannya (2018), di mana BERT dibandingkan dengan OpenAI GPT, yang juga memiliki ukuran penyematan kata sebesar 768. Ukuran ini dipilih untuk memastikan perbandingan yang adil antara kedua model. Model GPT dari OpenAI pada waktu itu merupakan versi pertama yang mengikuti arsitektur Transformer dengan ukuran 512, namun pada pengembangan model GPT-1 ini, ukuran penyematannya ditingkatkan sebesar 50% sehingga menjadi 768 (Vaswani, et al., 2017; Radford, et al., 2024).

Skenario kedua menggunakan varian BERT biasa. Berbeda dengan model pada metode pertama yang dirancang khusus untuk klasifikasi, model ini lebih umum penggunaannya. Metode kedua adalah pencarian semantik, di mana makna antara kueri dan dokumen dihitung menggunakan *Cosine Similarity*. Vektorisasi diperlukan terlebih dahulu sebelum proses ini dapat dilakukan. Metode ini menggunakan keluaran *pooler* dari BERT, berbeda dengan metode pertama yang mengambil keluaran *hidden states*. Meskipun keduanya adalah vektor dari BERT, terdapat perbedaan antara *hidden states* dan *pooler*: hidden states merupakan *word-level embedding*, sedangkan *pooler* merupakan *sentence-level embedding*. Artinya, *hidden states* menangkap makna tiap kata dalam kalimat sedangkan *pooler* menangkap makna dari keseluruhan kalimat. *Hidden states* cocok digunakan untuk tugas klasifikasi seperti pada metode pertama. Sebaliknya, pada metode kedua,

pooler sudah sangat memadai karena lebih efisien dalam penggunaan memori dan tidak digunakan untuk klasifikasi.

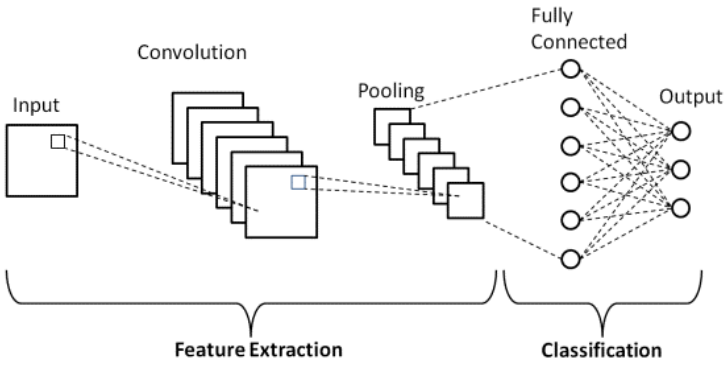


Gambar 7. Ilustrasi *Pooler* BERT

Hasil penyematan kata pada Gambar 6 dan Gambar 7 menunjukkan perbedaan antara penyematan *hidden states* dan penyematan *pooler*. Gambar 6 memperlihatkan banyaknya *array* dalam satu urutan, di mana setiap *array* mewakili penyematan tiap kata. Sebaliknya, Gambar 7 memperlihatkan bahwa penyematan diwakili oleh satu *array* dalam satu urutan.

1.2.8 CNN

CNN adalah salah satu jenis algoritma *deep learning* yang umumnya digunakan untuk tugas-tugas terkait pengenalan dan pemrosesan gambar. CNN terdiri dari berbagai lapisan, termasuk lapisan konvolusi, lapisan *pooling*, dan lapisan *fully-connected*. Arsitektur CNN terinspirasi oleh cara otak manusia mengolah gambar, sehingga sangat cocok untuk menangkap pola hierarkis dan ketergantungan spasial dalam gambar (LeCun, et al., 1998).



Gambar 8. Arsitektur Dasar CNN (Rohalia, 2023)

Penelitian ini menggunakan CNN untuk memproses teks atau *natural language processing*. Lapisan-lapisan CNN yang digunakan dalam penelitian ini adalh sebagai berikut:

a. Lapisan Konvolusi

Lapisan konvolusi dalam penelitian ini mengoperasikan urutan teks yang direpresentasikan oleh penyematan kata. *Filter* atau *kernel* pada lapisan ini bekerja pada jendela tertentu dari urutan *input*. Sebagai contoh, filter dengan nilai 3 akan memproses jendela yang berisi 3 kata pada satu waktu.

Filter pada lapisan konvolusi adalah matriks bobot kecil yang meluncur di atas data input untuk melakukan operasi konvolusi. Bobot ini dipelajari selama pelatihan melalui *backpropagation*. Ketika model menjadi terlalu rumit, ada risiko model akan "menghafal" data latih daripada menggeneralisasi dengan baik ke data yang tidak terlihat, yang menyebabkan *overfitting*. Teknik regularisasi diterapkan untuk mengatasi masalah ini.

Lapisan ini juga memiliki *stride* yang menentukan berapa banyak posisi gerakan filter di sepanjang urutan. Sebagai contoh, *stride* dengan nilai 1 berarti *filter* bergerak satu kata pada satu waktu, menghasilkan peta fitur yang padat. *Stride* dengan nilai besar akan mengakibatkan lebih sedikit posisi yang dievaluasi.

Nilai *filter* dan *stride* ditentukan terlebih dahulu, kemudian fungsi aktivasi diterapkan pada lapisan ini (umumnya ReLU) untuk memperkenalkan non-linearitas. Langkah ini membantu model mempelajari pola yang lebih kompleks dan representasi dalam teks.

b. Lapisan *Pooling*

Lapisan *pooling* adalah operasi *down-sampling* yang berfungsi untuk mengurangi dimensi spasial atau temporal dari peta fitur input, biasanya dengan memilih fitur yang paling menonjol di suatu wilayah. Dua jenis operasi *pooling* yang paling umum adalah *max pooling* dan *average pooling*.

Average pooling menghitung nilai rata-rata elemen dalam setiap wilayah, yang memungkinkan diperolehnya representasi fitur-fitur yang lebih halus. Namun, *max pooling* lebih umum digunakan karena mampu menangkap fitur paling menonjol. *Max*

pooling menyoroti fitur-fitur ini dengan memilih elemen-elemen yang paling signifikan dari setiap wilayah teks.

c. Lapisan *Dropout*

Lapisan *dropout* adalah teknik regularisasi yang digunakan untuk menghindari *overfitting*. Teknik ini bekerja dengan "membuang" (menjadikan nol) sebagian neuron atau unit secara acak selama pelatihan model, yang memaksa jaringan mempelajari fitur-fitur yang lebih tangguh dan tidak bergantung pada neuron tertentu. *Dropout* membantu model melakukan generalisasi lebih baik terhadap data yang tidak terlihat, sehingga mengurangi kemungkinan *overfitting* pada data latih.

d. Lapisan *Dense* (*Fully-Connected*)

Lapisan *dense*, atau yang biasa disebut lapisan *fully-connected*, memiliki peran krusial dalam menggabungkan fitur-fitur yang diekstraksi oleh lapisan sebelumnya, seperti lapisan konvolusi dan lapisan *pooling*, untuk menghasilkan keluaran akhir dari model.

Lapisan *dense* biasanya menerapkan fungsi aktivasi non-linear seperti ReLU, *sigmoid*, atau *softmax* setelah penjumlahan input tertimbang. Non-linearitas ini memungkinkan jaringan memodelkan hubungan yang kompleks dalam data, yang penting untuk tugas-tugas seperti klasifikasi teks di mana batas-batas linear sederhana mungkin tidak memadai.

1.2.9 *Cosine Similarity*

Cosine similarity adalah metrik yang digunakan dalam pembelajaran mesin, khususnya dalam bidang *natural language processing* dan pengambilan informasi, untuk mengukur kesamaan antara dua vektor bukan nol. Teknik ini digunakan secara luas karena efektivitasnya dalam membandingkan orientasi vektor daripada besarnya, sehingga membuatnya ideal untuk tugas-tugas yang mengutamakan distribusi relatif komponen dalam vektor. Diberikan 2 vektor v_1 dan v_2, kesamaan kosinus didefinisikan sebagai:

$$cosine\_similarity\ (v_1, v_2) = \frac{v_1 \cdot v_2}{\|v_1\| \cdot \|v_2\|} \qquad (1)$$

Di mana, $v_1 \cdot v_2$ merupakan produk titik dari v_1 dan v_2. $\|v_1\|$ dan $\|v_2\|$ merupakan megnitudo (atau panjang) dari vektor v_1 dan v_2. Dalam konteks ini, vektor v_1 dan v_2 adalah vektor yang dihasilkan oleh penyematan kata BERT, yang masing-masing mewakili kueri dan dokumen. Produk titik $v_1 \cdot v_2$ menghitung kesamaan arah antara kedua vektor, sementara magnitudo $\|v_1\|$ dan $\|v_2\|$ menormalkan nilainya.

1.2.10 Metrik Evaluasi

Dalam penelitian ini, RMSE (*Root Mean Square Error*), MAE (*Mean Absolute Error*), dan NDCG (*Normalized Discounted Cumulative Gain*) digunakan untuk

mengevaluasi kinerja model pencarian semantik yang mengimplementasikan *Cosine Similarity* dan BERT.

Root Mean Square Error (RMSE) mengukur akar kuadrat dari rata-rata kuadrat perbedaan antara peringkat yang diprediksi dan peringkat aktual (Makridakis, et al., 1997). RMSE lebih sensitif terhadap kesalahan yang lebih besar, sehingga memberikan bobot lebih pada *outliers* atau deviasi yang signifikan.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}|(y_i - \hat{y}_i)^2|} \quad (2)$$

Di mana, y_i adalah nilai asli. $\hat{y}_i$ adalah nilai prediksi. n adalah jumlah asli observasi.

Mean Absolute Error (MAE) menghitung rata-rata perbedaan absolut antara peringkat yang diprediksi dan peringkat aktual (Makridakis, et al., 1997), memberikan ukuran kesalahan yang sederhana dan mudah dipahami:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \quad (3)$$

MAE kurang sensitif terhadap kesalahan besar dibandingkan dengan RMSE, memberikan ukuran langsung dari besarnya kesalahan. NDCG mengevaluasi kualitas peringkat dengan mempertimbangkan posisi item yang relevan dalam daftar peringkat (Järvelin dan Kekäläinen, 2002). Item relevan yang berada di posisi lebih tinggi memberikan kontribusi lebih besar terhadap skor:

$$DCG = \sum_{i=1}^{n}\frac{2^{rel_i} - 1}{\log_2(i+1)} \quad (4)$$

$$NDCG = \frac{DCG}{IDCG} \quad (5)$$

Di mana, rel_i merupakan nilai relevan dari item pada posisi i. DCG merupakan *Discounted Cumulative Gain*. $IDCG$ merupakan DCG ideal, yang mana merupakan nilai maksimum DCG.

Model klasifikasi sentimen *Fine-Tuned* BERT + CNN dievaluasi menggunakan beberapa metrik yang digunakan untuk mengukur efektivitasnya, antara lain:

Akurasi mengukur tingkat keseluruhan kebenaran prediksi model, yaitu rasio antara data yang diprediksi dengan benar (baik positif maupun negatif) terhadap total jumlah prediksi (Bishop, 2006; Duda, et al., 2000):

$$Accuracy = \frac{TP_{total}}{Total\ Data} \quad (6)$$

Di mana, TP_{total} adalah total jumlah True Positive dari tiap kelas. Total Data merupakan jumlah kesuluran data termasuk prediksi yang benar dan prediksi yang tidak benar.

Confusion Matrix, *Precision*, *Recall* dan F1-*Score*: Selain akurasi dan *loss*, kinerja model dijelaskan lebih lanjut menggunakan *confusion matrix*, *precision*, *recall*,

dan F1-*score*, yang memberikan gambaran komprehensif tentang performa klasifikasi untuk setiap kelas sentimen (positif, netral dan negatif).

Precision mengukur akurasi prediksi positif, yaitu rasio antara *True Positive* (TP) dengan total prediksi positif, yang mencakup *True Positive* dan *False Positive* (FP) (Bishop, 2006; Duda, et al., 2000).

$$Precision = \frac{TP}{TP + FP} \quad (7)$$

Recall (juga disebut *Sensitivity* atau *True Positive Rate*): *Recall* mengukur seberapa baik model mengidentifikasi semua data yang relevan. *Recall* merupakan rasio antara *True Positive* (TP) dengan total data positif aktual, yang mencakup *True Positive* dan *False Negative* (FN) (Bishop, 2006; Duda, et al., 2000).

$$Recall = \frac{TP}{TP + FN} \quad (8)$$

F1-*score* adalah rata-rata harmonis dari *precision* dan *recall*. Metrik ini menyeimbangkan antara *precision* dan *recall*, menjadikannya metrik yang baik ketika distribusi kelas tidak seimbang (Bishop, 2006; Duda, et al., 2000).

$$F1 - Score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (9)$$

F1-*score* menyediakan satu metrik yang menggabungkan keunggulan *precision* dan *recall*. Metrik ini sangat berguna dalam situasi di mana keseimbangan antara kedua metrik tersebut diperlukan, karena bergantung hanya pada salah satunya mungkin tidak memberikan penilaian yang menyeluruh terhadap kinerja model.

1.3 Perumusan Masalah

Berdasarkan latar belakang masalah di atas, maka rumusan masalah pada penelitian ini adalah sebagai berikut:

1. Bagaimana analisis sentimen dapat diintegrasikan secara efektif ke dalam sistem rekomendasi untuk meningkatkan relevansi rekomendasi dalam skenario *cold-start*?
2. Bagaimana kombinasi *Fine-Tuned* BERT dan CNN mampu mengungguli model klasifikasi sentimen sebelumnya?
3. Bagaimana pemberian bobot pada ulasan berdasarkan polaritas sentimen dapat meningkatkan peringkat item rekomendasi dengan mengutamakan item yang memiliki ulasan lebih positif?
4. Bagaimana penggunaan *Cosine Similarity* dengan vektor yang diekstrak oleh BERT meningkatkan keselarasan semantik antara data interaksi pengguna dan item yang direkomendasikan, menghasilkan rekomendasi yang lebih personal dan relevan?

1.4. Tujuan dan Manfaat

1.4.1 Tujuan

Adapun tujuan dari penelitian ini adalah sebagai berikut:

1. Mengimplementasikan analisis sentimen untuk meningkatkan relevansi rekomendasi dalam skenario *cold-start.*
2. Menganalisis efektivitas kombinasi *Fine-Tuned* BERT dan CNN dalam mengungguli model klasifikasi sentimen pada penelitian sebelumnya.
3. Menentukan bobot ulasan berdasarkan polaritas sentimen untuk memperbaiki peringkat item rekomendasi dan mempromosikan item yang memiliki ulasan lebih positif.
4. Menganalisis kinerja metrik ukur *Cosine Similarity* dalam mengukur persamaan antar item dan membandingkannya dengan metrik ukur lain yang digunakan pada penelitian sebelumnya.

1.4.2 Manfaat

Manfaat dari penelitian ini adalah sebagai berikut:

1. Mengembangkan solusi inovatif untuk mengatasi masalah dalam sistem rekomendasi dengan mengintegrasikan sentimen ulasan sebagai fitur tambahan.
2. Merancang pendekatan baru dengan memasukkan sentimen ulasan sebagai fitur tambahan yang melengkapi *rating* dan *likes* dalam sistem rekomendasi dengan mengadopsi teknik pemberian bobot pada sentimen positif, netral dan negatif untuk meningkatkan keakuratan peringkat item, memungkinkan sistem memberikan rekomendasi yang lebih mendalam dan relevan bagi pengguna.
3. Mengevaluasi penggunaan *Cosine Similarity* pada vektor yang dihasilkan dari model BERT untuk meningkatkan relevansi rekomendasi dalam sistem rekomendasi berbasis konten sehingga meningkatkan pemahaman kesamaan semantik antara data interaksi pengguna dan item yang direkomendasikan, menghasilkan rekomendasi yang lebih personal dan relevan bagi setiap pengguna.

BAB II
METODE PENELITIAN

2.1 Metode Penyelesaian Masalah

2.1.1 Metode Rekomendasi

Strategi sistem rekomendasi dalam penelitian ini diilustrasikan pada Gambar 12. Dalam skenario ketika sistem tidak memiliki rekam jejak interaksi pengguna, sistem akan mengeksekusi metode pertama, yaitu rekomendasi berdasarkan sentimen pengguna lain. Sebaliknya, jika sistem memiliki rekam jejak interaksi pengguna, sistem akan mengeksekusi metode kedua, yaitu rekomendasi berdasarkan kalkulasi kesamaan kueri pengguna.

Metode pertama menggunakan sistem peringkat di mana peringkat ditentukan dengan melakukan pembobotan pada skor sentimen yang. *Ground truth* pada metode ini menggunakan rating IMDB skala 1 sampai 5. Sedangkan metode kedua menggunakan *Cosine Similarity* untuk mengkalkulasi kesamaan semantik antar kueri dan dokumen. Kueri merujuk pada rekam jejak interaksi pengguna, sedangkan dokumen mengacu pada dataset informasi film-film. Skala *ground truth* pada metode ini adalah [-1, 1] di mana -1 artinya tidak relevan dan 1 artinya relevan.

2.1.2 Metode Klasifikasi

Klasifikasi sentimen yang menggabungkan *Fine-Tuned* BERT dan CNN termasuk dalam bidang *Natural Language Processing* (NLP). CNN telah lama menjadi metode yang umum untuk klasifikasi, sementara BERT juga bukan model baru dalam tugas-tugas klasifikasi di *machine learning*.

TFBertForSequenceClassification adalah model BERT yang dirancang khusus untuk tugas klasifikasi. BERT secara umum memiliki kemampuan menghasilkan penyematan kontekstual pada setiap token dalam urutan, yang memungkinkan BERT menangkap makna tiap kata dalam setiap kalimat. TFBertForSequenceClassification sangat baik dalam menangkap makna kontekstual dan dilatih khusus (*fine-tuned*) untuk klasifikasi senitmen menunjukkan bahwa model ini berpotensi menghasilkan prediksi sentimen yang berkualitas.

Model ini dimodifikasi melalui penambahan lapisan CNN. CNN mampu menangkap pola lokal dan fitur hierarkis dalam data. Kemampuan ini memungkinkan CNN mengidentifikasi fitur lokal penting, seperti *n-grams* dari *hidden states* yang dihasilkan oleh TFBertForSequenceClassification, yang berkontribusi pada tugas klasifikasi. Penggabungan kedua model klasifikasi ini berpotensi menghasilkan prediksi sentimen yang lebih akurat.

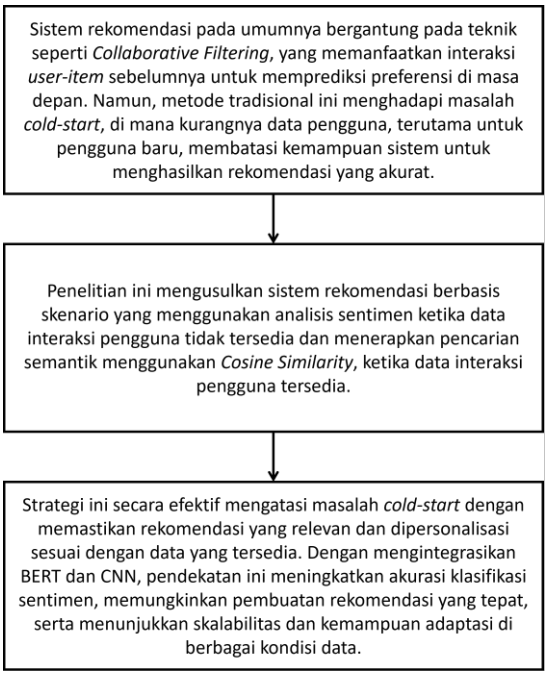
2.1.3 Hipotesis Penelitian

Penelitian ini dilakukan dengan mengangkat asumsi atau hipotesis bahwa:

1. Menggabungkan CNN dan BERT untuk klasifikasi sentimen memanfaatkan efisiensi CNN dalam mengekstraksi fitur lokal dan pemahaman kontekstual BERT yang mendalam, menghasilkan model dengan akurasi yang unggul dan generalisasi yang lebih baik. CNN unggul dalam mengidentifikasi pola lokal dengan cepat, sementara BERT menangkap konteks semantik kata-kata, penggabungan model ini sangat efektif untuk analisis sentimen.
2. Pemberian bobot pada ulasan berdasarkan polaritas (misalnya, ulasan positif diberi bobot lebih tinggi) dapat meningkatkan kualitas urutan rekomendasi dengan mendorong item yang memiliki ulasan positif lebih banyak ke posisi teratas.
3. Penggunaan *Cosine Similarity* dengan vektor yang diekstrak menggunakan BERT juga dapat meningkatkan kesamaan semantik antara data interaksi pengguna dan item yang direkomendasikan, menghasilkan rekomendasi yang lebih personal dan relevan.

2.1.4 Kerangka Pikir Penelitian

Kerangka pikir pada penelitian ini dapat dilihat pada Gambar 9 yang menggambarkan kontribusi pada penelitian ini.



Gambar 9. Kerangka Pikir

2.2 Tempat dan Waktu

Penelitian ini dimulai pada bulan Desember 2023 dan berlangsung hingga Oktober 2024. Objek penelitian berupa ulasan-ulasan film diambil dengan teknik *web scraping* dari platform database film LetterBoxd dan kritik film dari Kaggle. Penelitian dilakukan di Laboratoriun *Computer Based System* (CBS), Departemen Teknik Informatika, Fakultas Teknik, Universitas Hasanuddin.

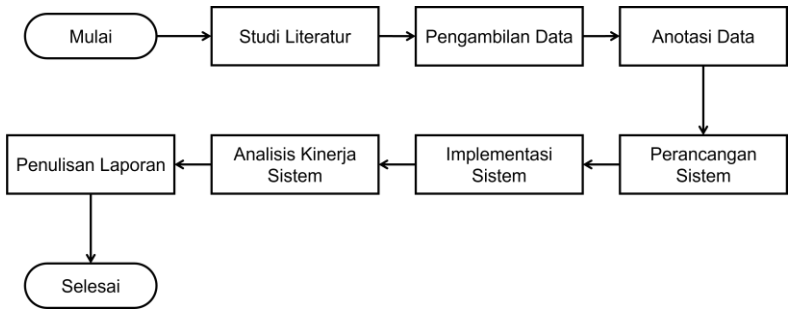
2.3 Benda Uji dan Alat

Benda dan alat yang digunakan dalam penelitian ini meliputi perangkat lunak dan perangkat keras berikut:

1. Perangkat lunak
   a. Windows 11 Pro, 64 bit
   b. *Jupyter Notebook*
   c. *Conda Environment*
2. Perangkat keras
   a. Sistem: Asus
   b. Prosesor: 12th Gen Intel® Core™ i5-12400F (12 CPUs), ~2.5GHz
   c. GPU: NVIDIA GeForce RTX 3060, 12 GB
   d. RAM: 16 GB

2.4 Tahapan penelitian

Penelitian dilakukan dengan beberapa tahapan seperti yang ditunjukkan pada Gambar 10 berikut:



Gambar 10. Tahapan Penelitian

2.5 Teknik Pengambilan Data

Penelitian ini menggunakan 3 dataset. Dataset pertama digunakan untuk *training* dan *validation*, dataset ini berupa kritik-kritik film dari *Rotten Tomatoes*. Dataset kedua digunakan untuk *testing*, dataset ini berupa ulasan-ulasan film dari IMDb. Dataset ketiga digunakan untuk visualisasi sistem, dataset ini berupa ulasan-ulasan film dari LetterBoxd.

(a)

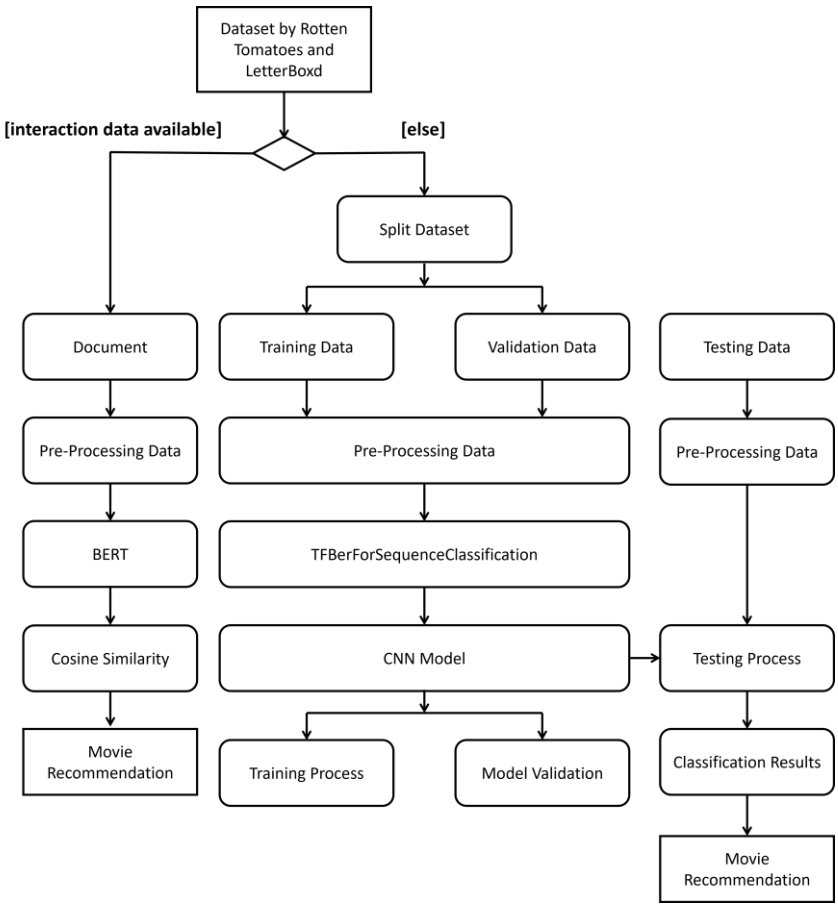
(b)

(c)

Gambar 11. Sumber Data dari (a) *Rotten Tommatoes;* (b) IMDb; dan (c) LetterBoxd

2.6 Perancangan dan Implementasi Sistem

Perancangan sistem bertujuan untuk memberikan gambaran detail mengenai sistem yang akan dikembangkan, serta memahami alur proses dalam sistem tersebut. Adapun rancangan sistem rekomendasi berbasis skenario ditunjukkan pada Gambar 12.



Gambar 12. Rancangan Sistem

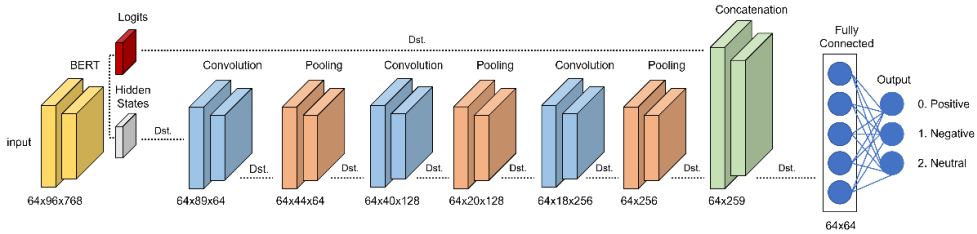
Dataset diambil dari Kaggle yang meliputi ulasan film dan informasi umum tentang film. Penganotasian ulasan sebagai dataset latih dan validasi dilakukan secara otomatis menggunakan RoBERTa, karena penganotasian manual membutuhkan banyak sumber daya manusia dan waktu, yang tidak tersedia bagi peneliti. Ulasan tersebut dianotasikan menjadi tiga kategori, yaitu positif, netral dan negatif. Kategori-kategori ini disebut sebagai kelas pada dataset. Model kemudian dilatih menggunakan dataset yang telah diberi anotasi dan model klasifikasi ini digunakan untuk memprediksi sentimen pada data uji. Hasil prediksi dari model kemudian digunakan sebagai fitur untuk rekomendasi film. Proses ini dijalankan dalam skenario di mana sistem tidak memiliki data rekam jejak interaksi pengguna.

Dalam skenario di mana sistem memiliki data rekam jejak interaksi pengguna, yang disebut sebagai kueri dalam penelitian ini, metrik kesamaan *Cosine Similarity* digunakan untuk menghitung kesamaan semantik antara kueri dan dokumen. Dokumen dalam penelitian ini adalah dataset yang berisi informasi umum tentang film. Diagram alur sistem rekomendasi pada penelitian ini mengikuti standar *Unified*

Modeling Language (UML) pada kasus penggunaan *activity diagrams* (Fowler, 2004).

2.6.1 Arsitektur Model

Penelitian ini menggunakan dua metode, yang masing-masing memiliki cara pemecahan masalah tersendiri. Sistem rekomendasi ini menggunakan *Fine-Tuned* BERT + CNN sebagai model prediksi sentimen dan *Cosine Similarity* untuk pencarian semantik



Gambar 13. Arsitektur Model *Fine-Tuned* BERT + CNN

Model *Fine-Tuned* BERT + CNN pada penelitian ini memiliki 10 lapisan dan untuk membangun model ini dibutuhkan *python* libraries yaitu:

1. *Pickle* untuk menyimpan objek *python* ke dalam bentuk *file* dengan cara serialisasi dan deserialiasi, ini berguna untuk menyimpan hasil penyematan dari BERT agar bisa dipakai kembali.
2. *Pandas* untuk membaca dataset yang dipakai pada penelitian ini.
3. *Tensorflow* untuk memfasilitasi pembuatan dan pelatihan model *deep learning*.
4. *Numpy* untuk membuat *array* dari dataset.
5. *BertTokenizer* untuk tokenisasi dataset penelitian.
6. *TFBertForSequenceClassification* merupakan model BERT untuk mengolah data sekuensial.
7. *cosine_similarity* untuk mengkomputasi persamaan kosinus antara 2 vektor.

Model *Fine-Tuned* BERT + CNN menggunakan dua model BERT; yang pertama berfungsi sebagai tokenisasi, sedangkan yang kedua sebagai model *deep learning*. BERT menghasilkan tiga keluaran, yaitu urutan teks yang ditokenisasi, yang mencakup "*input_ids*", "*token_type_ids*" dan "*attention_mask*".

```
input_ids = Input(shape=(shape,), dtype=tf.int32, name='input_ids')
token_type_ids = Input(shape=(shape,), dtype=tf.int32, name='token_type_ids')
attention_mask = Input(shape=(shape,), dtype=tf.int32, name='attention_mask')
```

Ketiga keluaran ini menjadi masukan untuk model *Fine-Tuned* BERT. Model *Fine-Tuned* BERT kemudian menghasilkan keluaran berupa *hidden states* dan *logits.*

```
bert_outputs = bert_model([input_ids, token_type_ids, attention_mask], output_hidden_states=True)
hidden_states = bert_outputs.hidden_states[-1]
```

```
logits = bert_outputs.logits
```

Setelah itu keluaran dari model BERT ini akan memasuki beberapa lapisan, yaitu:

a. Lapisan *Convolutional Neural Network*

Terdapat 3 lapisan Conv1D yang masing-masing diberi regularization untuk mengurangi potensi *overfitting.*

```
conv1d_layer = Conv1D(64, 8, activation='relu', kernel_regularizer=regularizers.l2(0.025))(hidden_states)
conv1d_layer_2 = Conv1D(128, 5, activation='relu', kernel_regularizer=regularizers.l2(0.025))(dropout_layer_1)
conv1d_layer_3 = Conv1D(256, 3, activation='relu', kernel_regularizer=regularizers.l2(0.025))(dropout_layer_2)
```

b. Lapisan *MaxPooling1D*

Terdapat 2 lapisan *MaxPooling1D* yang fungsinya untuk melakukan *downsampling* secara progresif.

```
maxpooling1d_layer = MaxPooling1D(2)(conv1d_layer)
maxpooling1d_layer_2 = MaxPooling1D(2)(conv1d_layer_2)
```

c. Lapisan *GlobalMaxPooling1D*

Terdapat 1 lapisan *GlobalMaxPooling1D* yang fungsinya untuk meringkas seluruh masukan menjadi representasi yang berukuran tetap.

```
globalmaxpooling1d_layer = GlobalMaxPooling1D()(conv1d_layer_3)
```

d. Lapisan *Dropout*

Terdapat 2 lapisan Dropout yang fungsinya untuk mengurangi potensi *overfitting*.

```
dropout_layer_1 = Dropout(0.45)(maxpooling1d_layer)
dropout_layer_2 = Dropout(0.45)(maxpooling1d_layer_2)
```

e. Lapisan *Concatenation*

Terdapat 1 lapisan penggabungan yang fungsinya adalah untuk menggabungkan keluaran klasifikasi oleh BERT (*logits*) dan fitur-fitur dari CNN.

```
combined_features = tf.concat([logits, globalmaxpooling1d_layer], axis=-1)
```

f. Model *Compilation* dan *Training*

Model dikompilasi menggunakan *Adam optimizer* dan *categorical cross-entropy* untuk *loss*, yang sesuai untuk klasifikasi multi-kelas. Proses pelatihan model menerapkan teknik *early stopping*, yang secara otomatis menghentikan pelatihan ketika nilai *val_loss* tidak lagi meningkat, sehingga membantu menghindari *overfitting* dan meningkatkan kemampuan generalisasi model. Penentuan nilai *learning rate* dilakukan secara otomatis menggunakan teknik *learning rate scheduler*, yang mengurangi nilai *learning rate* jika *val_loss* tidak menunjukkan peningkatan pada setiap iterasi yang ditentukan.

```
combined_model.compile(optimizer=Adam(learning_rate=3e-6), loss='categorical_crossentropy', metrics=['accuracy'])
early_stopping = EarlyStopping(monitor='val_loss', patience=3, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=2, min_lr=1e-6)
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_ids (InputLayer)         [(None, 96)]         0           []

 token_type_ids (InputLayer)    [(None, 96)]         0           []

 attention_mask (InputLayer)    [(None, 96)]         0           []

 tf_bert_for_sequence_classific  TFSequenceClassifie  109484547  ['input_ids[0][0]',
 ation (TFBertForSequenceClassi  rOutput(loss=None,               'token_type_ids[0][0]',
 fication)                      logits=(None, 3),                 'attention_mask[0][0]']
                                 hidden_states=((No
                                ne, 96, 768),
                                 (None, 96, 768),
                                 (None, 96, 768),
                                 (None, 96, 768),
                                 (None, 96, 768),
                                 (None, 96, 768),
                                 (None, 96, 768),
                                 (None, 96, 768),
                                 (None, 96, 768),
                                 (None, 96, 768),
                                 (None, 96, 768),
                                 (None, 96, 768),
                                 (None, 96, 768)),
                                 attentions=None)

 conv1d (Conv1D)                (None, 89, 64)       393280      ['tf_bert_for_sequence_classifica
                                                                 tion[0][12]']

 max_pooling1d (MaxPooling1D)   (None, 44, 64)       0           ['conv1d[0][0]']

 dropout_38 (Dropout)           (None, 44, 64)       0           ['max_pooling1d[0][0]']

 conv1d_1 (Conv1D)              (None, 40, 128)      41088       ['dropout_38[0][0]']

 max_pooling1d_1 (MaxPooling1D)  (None, 20, 128)     0           ['conv1d_1[0][0]']

 dropout_39 (Dropout)           (None, 20, 128)      0           ['max_pooling1d_1[0][0]']

 conv1d_2 (Conv1D)              (None, 18, 256)      98560       ['dropout_39[0][0]']

 global_max_pooling1d (GlobalMa  (None, 256)         0           ['conv1d_2[0][0]']
 xPooling1D)

 tf.concat (TFOpLambda)         (None, 259)          0           ['tf_bert_for_sequence_classifica
                                                                 tion[0][13]',
                                                                  'global_max_pooling1d[0][0]']

 dense (Dense)                  (None, 64)           16640       ['tf.concat[0][0]']

 dense_1 (Dense)                (None, 3)            195         ['dense[0][0]']

==================================================================================================
Total params: 110,034,310
Trainable params: 110,034,310
Non-trainable params: 0
__________________________________________________________________________________________________
```

Gambar 14. Ringkasan Model *Fine-Tuned* BERT + CNN

Model prediksi sentimen ini kemudian diimplementasikan pada sistem rekomendasi dalam skenario di mana sistem tidak memiliki data berupa jejak interaksi pengguna dengan logika pemberian bobot pada tiap kelas, berikut tahapannya:

a. Prediksi Sentimen

Prediksi polaritas ulasan dilakukan dengan menggunakan model *Fine-Tuned* BERT + CNN, kemudian hasil prediksi disimpan pada dataset yang kolomnya diberi nama "*Predicted Sentiment*".

```
predictions = predict_sentiments(reviews_list)
new_data['Predicted Sentiment'] = predictions
```

b. Hitung Jumlah Sentimen

Setelah menyimpan hasil prediksi sentimen, langkah berikutnya adalah menghitung jumlah setiap kelas (positif, negatif, netral) pada setiap film.

```
sentiment_counts = new_data['Predicted Sentiment'].value_counts()

sentiment_counts = new_data.groupby(
        ['Movie Title', 'Predicted  Sentiment']
).size().unstack(fill_value=0)
```

c. Pemberian Bobot

Setiap kelas dari ulasn film diberi bobot. Penelitian ini telah melakukan pengujian pada 3 metode pembobotan dan didapatkan *Bayesian Weighting* mengungguli 2 metode pembobotan lainnya. Metode pembobotan ini memastikan film dengan lebih banyak ulasan positif dan netral, serta lebih sedikit ulasan negatif, diposisikan lebih tinggi.

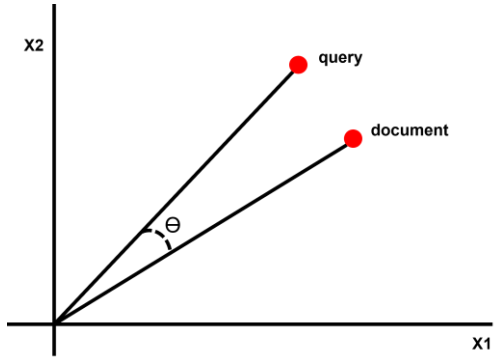
```
def bayesian_score(row, global_positive_mean, global_neutral_mean, global_negative_mean, weights, weight=weight):
  return (
    ((row['Positive'] + global_positive_mean * weights['positive']) /
     (row['Total'] + weight)) +
    ((row['Neutral'] + global_neutral_mean * weights['neutral']) /
     (row['Total'] + weight)) +
    ((row['Negative'] + global_negative_mean * weights['negative']) /
     (row['Total'] + weight))
  )
```

d. Rekomendasi Film

Film kemudian dapat direkomendasikan berdasarkan *score*-nya.

```
sorted_movies = sentiment_counts.sort_values(by='Score', ascending=False)
```

Pencarian semantik pada sistem rekomendasi ini menggunakan metrik ukur *Cosine Similarity*, metrik ukur ini superior dibanding metrik ukur persamaan lainnya dalam konteks karakter data berupa penyematan teks yang dihasilkan oleh BERT.



Gambar 15. Persamaan Kosinus

Gambar 15 yang mewakili *Cosine Similarity* menunjukkan bahwa skor kesamaan antara dua vektor diukur dalam rentang dari 0 hingga 1, di mana 0 menunjukkan ketidaksamaan total (tidak relevan) dan 1 menunjukkan kesamaan sempurna (relevan). Nilai *Cosine Similarity* yang mendekati 1 menunjukkan kesamaan yang kuat, yang berarti kueri dan dokumen memiliki keselarasan kontekstual. Gambar 15 secara visual menunjukkan bagaimana sudut kosinus *θ* antara vektor-vektor tersebut mewakili tingkat kesamaan.

Berbeda dengan model *Fine-Tuned* BERT + CNN, model pencarian semantik ini tidak memerlukan proses *training* dan *validation*, berikut adalah tahapan dari algoritma pencarian semantik menggunakan *Cosine Similarity:*

a. Konversi Teks Menjadi Vektor

Cosine Similarity adalah metrik untuk mengukur kesamaan dua vektor dengan menghitung sudut kosinus di antara keduanya. Oleh karena itu, dokumen perlu diubah menjadi vektor menggunakan BERT terlebih dahulu, kemudian keluaran hidden states dari vektor tersebut diekstrak.

```
def encode_text(text):
inputs = tokenizer(text, return_tensors='tf', truncation=True, padding=True)
outputs = model(**inputs)
return outputs.pooler_output.numpy()
```

b. Mengatur Susunan Vektor

Hasil penyematan teks kemudian disimpan dalam format .pkl agar dapat digunakan kembali. Untuk menggunakannya, perlu dilakukan penumpukan secara vertikal dengan tujuan mengubah vektor menjadi *array* 2D (matriks) agar memungkinkan perkalian vektor-matriks.

```
synopsis_vectors = np.vstack(document['synopsis_vectorized'].values)
cast_vectors = np.vstack(document['cast_vectorized'].values)
director_vectors = np.vstack(document['director_vectorized'].values)
producer_vectors = np.vstack(document['producer_vectorized'].values)
writer_vectors = np.vstack(document['writer_vectorized'].values)
title_vectors = np.vstack(document['title_vectorized'].values)
genres_vectors = np.vstack(document['genres_vectorized'].values)
```

c. Mengkomputasi Kesamaan Vektor

Logika dari pencarian semantik ini adalah membandingkan kueri dengan dokumen. *Cosine Similarity* digunakan untuk menghitung kesamaan vektor dari keduanya. Karena dokumen sudah berbentuk *array* 2D, kueri juga harus berbentuk serupa. Hasil *similarity score* nantinya akan berupa *array* 1D.

```
vectors = valid_query_types[query_type]
if isinstance(query, list):
    query_vectors = [encode_text(keyword).reshape(1, -1)
                     for keyword in query]
    query_vector = np.mean(query_vectors, axis=1)
else:
    query_vector = encode_text(query).reshape(1, -1)
similarities = cosine_similarity(query_vector, vectors).flatten()
```