LOGIC: Jurnal Penelitian Informatika
Vol. 1, Issue 1, pp. 46–52 (2023)
doi: http://doi.org/10.25124/logic.v1i1.6448

RESEARCH ARTICLE

Pembangunan *Knowledge Graph* dan Sistem Rekomendasi Film dengan GP 2

Rayhan Zanzabila, Gia Septiana Wulandari* and Izzatul Ummah

Fakultas Informatika, Universitas Telkom, Bandung, 40257, Jawa Barat, Indonesia
*Corresponding author: giaseptiana@telkomuniversity.ac.id
Received on 05 August 2023; accepted on 05 September 2023

Abstrak

Film, sebagai salah satu jenis hiburan yang banyak diminati, saat ini sudah berlimpah banyaknya. Setiap orang memiliki pendapat yang berbeda terhadap film-film tersebut. Sistem rekomendasi dapat dimanfaatkan untuk memberikan rekomendasi film yang terpersonalisasi untuk tiap orang, sehingga setiap orang dapat mudah memilih film yang akan mereka tonton. Saat ini sudah banyak sistem rekomendasi film yang digunakan. Metode yang digunakan pun bervariasi, seperti dengan sentiment *analysis, collaborative filtering,* atau dengan pembangunan *knowledge graph*. Namun, sistem rekomendasi film dengan *knowledge graph* yang sudah dilakukan sampai saat ini belum ada yang menggunakan GP 2. GP 2 adalah bahasa pemrograman yang mempunyai tingkat komputabilitas yang tinggi dan dapat divisualisasikan dengan mudah. Pada penelitian ini, sistem rekomendasi dibangun dengan GP 2. Secara umum, sistem yang dibangun memberi rekomendasi berdasarkan dua hal, yaitu kemiripan preferensi antar *user* dan nilai rata-rata suatu film. Dari empat skema rekomendasi yang dibangun, dua di antaranya memberikan akurasi terbaik, yaitu 71%. Namun, salah satunya membutuhkan waktu eksekusi yang jauh lebih singkat dari skema lainnya.

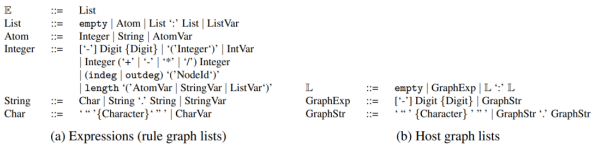
Key words: *Knowledge Graph*, Sistem Rekomendasi, Rekomendasi Film, GP 2.

Pendahuluan

Film merupakan salah satu jenis hiburan yang banyak diminati. Saat ini, terdapat banyak film yang mencakup berbagai jenis *genre*. Pada bulan Mei 2022 saja, terdapat 11 film orisinal yang rilis di *platform streaming* Netflix, dan jumlah ini tidak mencakup film serial dan film non-orisinal Netflix [1]. Setiap orang memiliki pendapat masing-masing mengenai berbagai film tersebut. Dengan banyaknya film yang bisa ditonton, orang-orang dapat merasa kebingungan untuk memilih, apalagi jika mereka tidak mempunyai banyak waktu senggang dan ingin memanfaatkannya dengan sebaik-baiknya. Rekomendasi film dapat dimanfaatkan untuk mengatasi masalah tersebut. Rekomendasi film adalah suatu sistem yang mengusulkan film-film yang mungkin disukai oleh orang-orang. Namun, tiap orang memiliki preferensi yang berbeda. Suatu film bisa menjadi film favorit dari suatu orang, tetapi tidak disukai oleh orang lainnya. Karena itulah rekomendasi sebaiknya dipersonalisasikan untuk tiap orang, sehingga orang-orang mendapatkan rekomendasi yang sesuai dengan preferensi mereka.

Saat ini, sistem rekomendasi merupakan hal yang lazim ditemui, tidak terkecuali pada bidang perfilman. Akan tetapi, sistem rekomendasi film yang biasanya ada saat ini lebih bergantung pada riwayat tontonan pelanggan, bukan berdasarkan penilaian pelanggan terhadap film yang telah mereka tonton. Studi terkait rekomendasi film pun seringkali dilakukan tanpa mempertimbangkan penilaian penonton, seperti yang ada pada studi oleh Pradhan [2] yang memberikan rekomendasi berdasarkan kemiripan antar riwayat tontonan pengguna dan kemiripan film. Terdapat beberapa tipe sistem rekomendasi berdasarkan metode yang digunakan. Salah satunya adalah *knowledge-base recommender system* (KBRS), yang diketahui memiliki efektivitas yang tinggi [3]. Akan tetapi, meskipun hasil yang didapatkan KBRS cukup baik, kelemahan KBRS terletak pada komponen pembelajaran yang kurang dapat memanfaatkan *human-computer interaction* [4]. Kekurangan ini akan semakin terlihat pada sistem rekomendasi film yang memiliki banyak film dan penonton dan akan bertambah terus seiring waktu, sehingga kemudahan pengguna dalam memahami sistem rekomendasi menjadi sangat penting.

Di sisi lain, penilaian orang-orang terhadap film dapat disimpan dalam sebuah *knowledge graph*, sehingga pengelolaannya pun dapat dilakukan dengan lebih mudah karena kita dapat melihat ilustrasi graf dari data yang kita miliki. *Knowledge graph* merupakan sebuah cara untuk merepresentasikan pengetahuan dalam bentuk graf. Terdapat suatu bahasa pemrograman graf, yaitu GP 2, yang dapat digunakan secara gratis, mempunyai tingkat komputabilitas yang tinggi, dan dapat



Gambar 1. *Syntax Abstrak* GP 2 Terkait Ekspresi.

merepresentasikan properti-properti dalam logika orde kedua monadik [5]–[6] sehingga sangat memungkinkan dapat digunakan untuk membangun suatu *knowledge graph*. *Rules* pada program GP 2 juga dapat digambarkan dalam bentuk graf, sehingga pembuat program dapat memanfaatkannya untuk menjelaskan cara kerja suatu program dalam bentuk visual. Oleh karena itu, penelitian ini mengimplementasikan penggunaan GP 2 dalam sistem rekomendasi film.

Tinjauan Pustaka

Sistem Rekomendasi

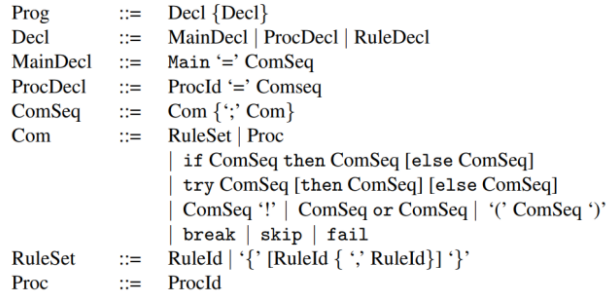
Sistem rekomendasi adalah alat penyaring informasi yang digunakan untuk memberikan rekomendasi terpersonalisasi. Sistem rekomendasi membantu penggunanya untuk mendapatkan produk/jasa/informasi yang sesuai dengan preferensi dan kebutuhan di antara banyaknya informasi yang ada saat ini [3]. Berdasarkan teknik yang digunakna, sistem rekomendasi dapat diklasifikasikan menjadi lima, yaitu *collaborative filtering, content-based filtering, knowledge-based filtering, ontology-based filtering,* dan *hybrid* [2], [3].

Knowladge Graph

Knowledge graph adalah graf multi-relasional yang menggambarkan entitas-entitas sebagai node dan relasi sebagai edge [7]. *Knowledge graph* memuat data yang saling berinterrelasi dalam representasi pengetahuan eksplisit berupa graf [8], [9]. Dalam satu dekade terakhir, banyak *knowledge graph* berskala besar yang dibangun. Pada dasarnya, *knowledge graph* adalah penggabungan *knowledge base* dengan sistem penalaran. Pembuatan *knowledge graph* bergantung pada ontologi yang digunakan. Ontologi sendiri merupakan model dari dunia (kemungkinan hanya berupa subhimpunan) yang mencakup tipe entitas, relasi penghubung, dan batasan antara kombinasi entitas dan relasi tersebut. Dengan kata lain, ontologi menggambarkan aturan-aturan cara tiap entitas tehubung dengan dengan entitas lainnya di dunia ini [10].

GP 2

Program GP 2 melakukan transformasi *graf input* menjadi *graf output*. Graf tersebut dapat berupa graf berarah dan juga mungkin terdapat sisi paralel dan loop. Secara formal, suatu graf G dapat didefinisikan sebagai sistem ¡VG, EG, sG, tG, lG, mG, pG¿ yang mengandung dua himpunan terbatas berisi simpul dan sisi, fungsi sumber dan target, fungsi pelabelan node parsial, fungsi pelabelan sisi, dan fungsi akar parsial [11]. *Syntax* bahasa pemrograman GP 2 didefinisikan pada Gambar 1 [12]. Kelas E pada Gambar 1 merupakan tipe data pada bahasa pemrograman GP 2. *Char* termasuk karakter yang direpresentasikan sebagai 32, 33, dan 35-126 dalam ASCII. Tipe String dapat berupa semua yang termasuk ke dalam Char dan gabungan dari Char. Tipe Integer mencakup bilangan bulat. Tipe Atom dapat berupa Integer atau String. List dapat berupa Atom atau kosong. Gambar 2 [12] menunjukkan *syntax* GP 2 terkait konstruksi kontrol program Main. *RuleId* adalah identifier dari suatu rule. Contoh *RuleId* dapat dilihat pada Gambar 3 [11], yaitu link. Konstruksi pemrograman utama pada



Gambar 2. *Syntax Abstrak Program* GP 2.

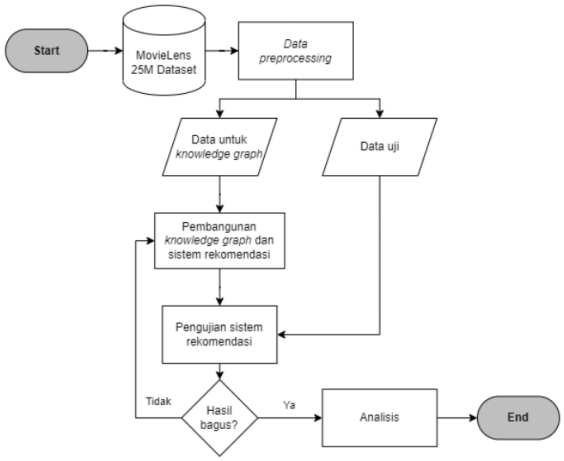
Gambar 3. Program *Transitive Closure*.

GP 2 adalah aturan transformasi graf yang diberi label dengan ekspresi. Contohnya dapat dilihat juga pada Gambar 3. [12]. [12]. Pada Gambar 3, aturan link mempunyai lima parameter formal bertipe list, left-hand graph dan *right-hand graph* yang dispesifikasi secara visual, dan kondisi yang diawali dengan kata kunci *where*. *Identifier* dari *node* tertulis di bawah tiap *node* sedangkan teks lainnya merupakan label dari *node* atau *edge* terkait. *Node* juga dapat ditandai dengan warna merah, hijau, biru, atau abu-abu. *Edge* dapat diberi warna atau berupa garis putus-putus.

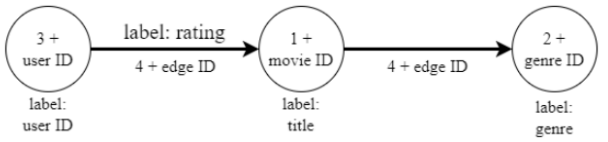
Sebuah program terdiri atas deklarasi aturan kondisional dan prosedur (*nonrekursif*), termasuk prosedur khusus bernama Main. *Rule-set call* (r1, . . . , rn) menjalankan salah satu aturan yang dapat diterapkan pada *host graph* (graf input) secara nondeterministik. *Rule-set call* ini akan gagal jika tidak ada aturan yang dapat diterapkan ke *host graph*. Komposisi sekuensial program P dan Q dituliskan P; Q. Misalkan perintah "if C *then* P *else* Q" dijalankan pada *host graph* G, maka pertama C akan dijalankan pada graf G. Jika eksekusi C ini menghasilkan graf, maka P akan dijalankan pada graf awal G. Namun, jika C tidak menghasilkan graf, Q akan dijalankan pada graf awal G. Perintah "*try C then* P *else* Q" mempunyai efek serupa, tetapi pada perintah *try*, P akan dijalankan pada graf hasil eksekusi C. *Loop* P! mengeksekusi P berulang kali sampai terjadi kegagalan. Setelahnya, P! akan terminasi dengan menghasilkan graf terakhir dari perulangan. Break dalam *loop* akan terminasi dari *loop* tersebut dan melanjutkan ke perintah setelahnya.

Sistem Rekomendasi Film

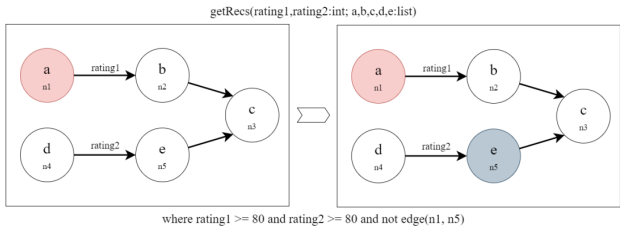
Studi terkait rekomendasi film telah dilakukan sebelumnya dengan berbagai metode. Sebagai contoh, Pavitha [13] melakukan studi terkait rekomendasi film dengan menggunakan sentiment *analysis*. Pada studi tersebut, sistem rekomendasi didasari oleh komentar penonton, sehingga komentar yang bersifat sarkastik mungkin saja diartikan berbeda oleh sistem. Selain itu, pembuatan sistem rekomendasi film menggunakan *collaborative filtering* dengan pendekatan SVD++ pun telah dilakukan oleh Anwar [11]. Pada studi tersebut, dilakukan perhitungan kemiripan antara film dan penonton sebagai dasar dari rekomendasi yang akan diberikan. Pembuatan *knowledge graph* untuk rekomendasi film pun juga telah dilakukan oleh Han [14]. Akan tetapi, beberapa relasi yang mungkin ada pada film masih belum cukup dieksplorasi, sehingga penambahan relasi masih harus dilakukan.



Gambar 4. Diagram Alur Pembangunan Sistem.



Gambar 5. Struktur *Knowledge Graph*.



Gambar 6. *Rule Getrecs* Pada Skema 1.

Metodologi Penelitian

Sistem Yang Dibangun

Alur Pembangunan Sistem

Kegiatan diawali dengan mengambil dataset yang akan dibangun menjadi *knowledge graph* dan *preprocessing* pada dataset tersebut. Hasil preprocessing dibagi menjadi dua, yaitu untuk *knowledge graph* dan data uji. Data untuk *knowledge graph* dimasukkan sebagai *input* sistem rekomendasi. Sistem rekomendasi mengeluarkan *output* graf yang kemudian digabung dengan data uji dalam bentuk graf juga. Graf data uji kemudian dijadikan *input* untuk program testing, yang mengeluarkan *output graf* berisi informasi hasil pengujian. Jika hasil pengujian buruk, tahap pembangunan *knowledge* graph dan sistem rekomendasi diulang kembali. Namun, jika hasil sudah cukup baik, maka kegiatan dilanjutkan ke tahap analisis.

Data *Preprocessing*

Dari kedua dataset yang digunakan, dilakukan beberapa tahap *preprocessing* dengan menggunakan *Python*. Judul film yang mengandung huruf non-ASCII ditransliterasi ke ASCII, karena string yang diterima program GP 2 adalah yang mengandung karakter ASCII [15]. Karakter yang tidak kompatibel dengan GP 2 atau GP 2 *Online Graph Visualiser* juga diganti atau dihapus. Judul film yang melebihi 63 karakter akan dipotong karena label graf pada GP 2 memiliki batas 63 karakter. Rating film yang sebelumnya berskala 0–5 dengan increment 0,5 diganti menjadi skala 0–100 dengan *increment* 1. Penggantian skala penilaian ini dilakukan karena sistem rekomendasi yang dibangun menggunakan hitungan rata-rata, sedangkan GP 2 sendiri hanya menerima bilangan bulat. Skala 0–100 meningkatkan presisi perhitungan. Berikutnya dilakukan pengambilan sampel dan *data splitting*. Diambil lima set sampel dari dataset yang ada. Set film yang diambil berjumlah 25, 50, 75, 100, dan 125. Berdasarkan sampel film, diambil *rating* dari 'ratings.csv' yang menilai film terkait. *User* yang penilaiannya diambil hanyalah *user* yang setidaknya sudah memberi penilaian terhadap 10 film sampel. Sampel penilaian tersebut kemudian dibagi menjadi dua dataset, yaitu dataset yang akan dibuat ke dalam *knowledge graph* untuk menghasilkan rekomendasi dan dataset yang digunakan dalam tahap *testing*, dengan rasio 4:1. Tahap ini diakhiri dengan tahap meng-export data yang diperlukan untuk tahap selanjutnya dalam format csv.

Pembangunan *Knowledge Graph*

Pembangunan *knowledge graph* dilakukan dengan bantuan *Python*. Hasil tahap *preprocessing* kemudian diubah ke dalam *knowledge graph* dengan struktur yang diilustrasikan pada Gambar 5. Node kiri pada Gambar 5 adalah *node user*. *Node user* memiliki *identifier* dengan prefiks 3 yang diikuti *user ID* sesuai dataset dan berlabel *user ID*. Node tengah menggambarkan film dengan identifier berprefiks 1 yang diikuti *movie ID* dan berlabel judul film. Node kanan melambangkan *genre* dengan identifier berprefiks 2 yang diikuti dengan *genre* ID. Node ini berlabel string yang merupakan nama *genre*. Seluruh *edge* pada graf memiliki identifier yang berprefiks 4 dan diikuti dengan *edge ID*. *Edge* yang menghubungkan *user* ke film menggambarkan penilaian *user* terhadap film tersebut. *Edge* ini memiliki label rating yang merupakan penilaian itu sendiri. Sedangkan *edge* yang menghubungkan film ke *genre* menunjukkan bahwa film tersebut memiliki *genre* terkait.

Pembangunan Sistem Rekomendasi

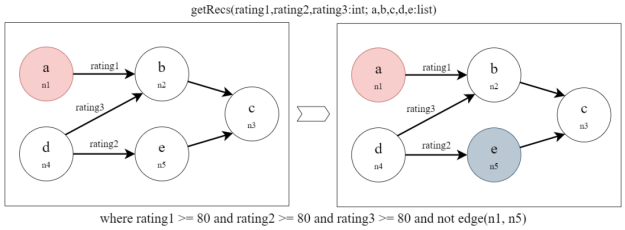
Sistem Rekomendasi Skema 1

Skema 1 memberikan rekomendasi berdasarkan kemiripan preferensi *user input* dengan *user* lainnya. Sistem mengambil *genre-genre* yang disukai *user input* dari film yang dinilai bagus olehnya, kemudian seluruh film yang memiliki setidaknya *genre* yang sama dan dinilai bagus oleh setidaknya satu *user* lain akan direkomendasikan.

Pada Gambar 6, setiap subgraf pada *graf input* akan ditransformasi dari graf kiri menjadi graf kanan. *User* yang dijadikan *input* ditandai dengan node berwarna merah. Sistem akan menandai suatu film sebagai rekomendasi (node abu-abu) jika film tersebut belum dinilai oleh *user input* dan terdapat *user lain* yang menilai film tersebut dengan batas nilai bagus ($\geq$ 80), serta film tersebut memiliki setidaknya satu *genre* yang sama dengan film lain yang dinilai baik oleh *user input*. Skema ini memiliki variasi lain yang menggunakan batas nilai $\geq$ 60. Pada skema ini, pertama dicari film (n2) yang disukai oleh *user input* (n1). Kemudian, akan dicari film lain (n5) yang memiliki *genre* (n3) yang sama dengan film yang disukai *user input* tersebut dan juga memiliki setidaknya satu *user* lain (n4) yang memberi penilaian baik. Film lain tersebut juga tidak boleh mempunyai penilaian dari *user input*. Jika semua kondisi ini terpenuhi, maka film lain tersebut ditandai abu-abu yang berarti film tersebut masuk ke dalam rekomendasi.

Sistem Rekomendasi Skema 2

Pada dasarnya, skema kedua yang diilustrasikan pada Gambar 7 mirip dengan skema pertama. Namun, skema 2 lebih memberikan fokus ke



Gambar 7. *Rule Getrecs* Pada Skema 2.

kesamaan preferensi kedua *user*. Perbedaannya terletak pada satu *edge* tambahan dari *user* selain *user input* ke arah film yang pernah dinilai baik oleh *user input*. *Edge* ini juga harus memiliki nilai yang baik. Seperti pada skema 1, batas nilai ini juga memiliki dua variasi, yaitu $\geq$ 80 dan $\geq$ 60. Pada skema ini, pertama dicari film (n2) yang disukai oleh *user input* (n1). Kemudian sistem juga mencari *user* lain (n4) yang juga menyukai film tersebut. Jika kedua kondisi ini terpenuhi, maka sistem mengganggap kedua *user* memiliki selera yang mirip untuk film ber*genre* sama dengan film yang mereka sukai tersebut. Maka, sistem akan mencari film lain (n5) yang mempunyai *genre* sama dengan film pertama dan juga disukai oleh *user* kedua. Film ini juga tidak boleh memiliki penilaian dari *user input*. Jika semua kondisi terpenuhi, maka film kedua ini akan direkomendasikan.

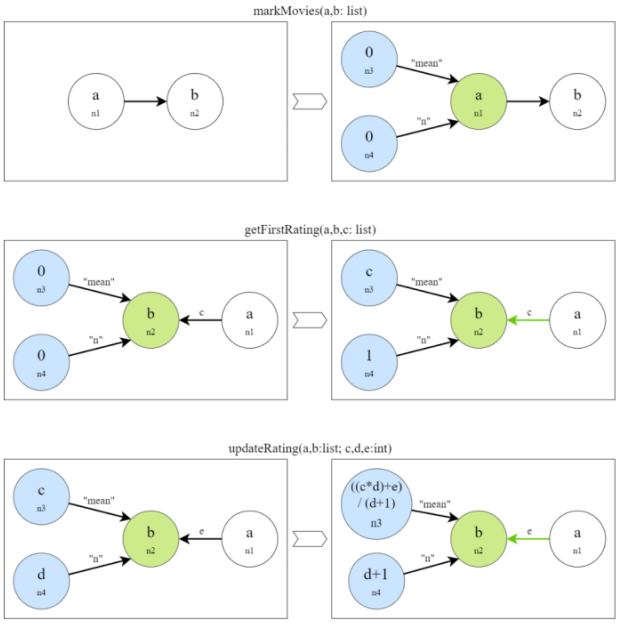
Sistem Rekomendasi Skema 3

Berbeda dengan dua skema sebelumnya yang memberi rekomendasi berdasarkan kemiripan preferensi *user*, skema 3 memberikan rekomendasi berdasarkan nilai rata-rata film dari seluruh *user*. Karena itu, program menghitung rata-rata tiap film sebelum memberikan rekomendasi. Penghitungan rata-rata diilustrasikan pada Gambar 8. *Rule* markMovies berfungsi untuk memuat sepasang node untuk tiap film (n1 pada markMovies) yang digunakan untuk menyimpan nilai rata-rata dan jumlah penilaian yang sudah dihitung ke dalam rata-rata tersebut. Dua node (node berwarna biru) ini pada awalnya diisi nilai 0.

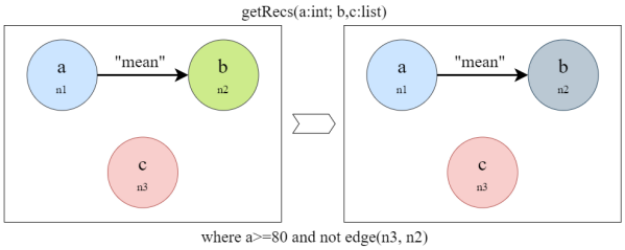
Kedua nilai ini dikalkulasi dengan dua *rule* berikutnya. *Rule getFirstRating* mengambil nilai pertama dan menyimpannya ke dalam kedua node biru, kemudian *updateRating* mengkalkulasi penilaian lainnya yang belum ditandai dengan *edge* hijau. Setelah itu, film yang memiliki nilai rata-rata $\geq$80 akan ditandai abu-abu yang berarti film tersebut masuk ke dalam daftar rekomendasi. Penandaan ini diilustrasikan pada Gambar 9. Skema 3 ini juga memiliki variasi lain yang dijalankan, yaitu kondisi a$\geq$80 yang diganti menjadi a$\geq$60, sesuai dengan variasi yang ada pada skema lainnya.

Sistem Rekomendasi Skema 4

Seperti skema 3, skema 4 juga menggunakan rata-rata. Namun, skema 4 menggunakan **rooted node** untuk meningkatkan efisiensi program. Penghitungan rata-rata pada skema 4 dapat dilihat pada Gambar 10. *Rule* init akan menandai satu node film (n2) sebagai *rooted node*. Kemudian akan dibuat dua *node* biru, n4 dan n5. *Node* n4 menyimpan salah satu rating dari film tersebut, sedangkan n4 menyimpan bilangan 1. *Node* n4 ini digunakan untuk menyimpan banyak rating yang sudah ditambahkan ke dalam n3, yang saat ini baru menyimpan satu rating. *Rule getSum* akan menambahkan rating yang belum ditambahkan ke *node* n3 dan menandai *edge*-nya dengan warna hijau, yang berarti sudah dihitung. *Rule* ini juga menambah satu label *node* n4 yang menyimpan banyak rating yang sudah dimasukkan ke *node* n3. *Rule getMean* akan menghitung nilai rata-rata dari seluruh penilaian yang diberikan oleh *user* kepada suatu film. *Rule* ini akan dijalankan setelah jumlah nilai suatu film selesai dihitung dalam *node* dengan *edge* berlabel "*mean*". Setelah itu, node film yang sebelumnya *rooted* ini diubah



Gambar 8. *Rule Markmovies, Getfirstrating,* Dan *Updaterating*.



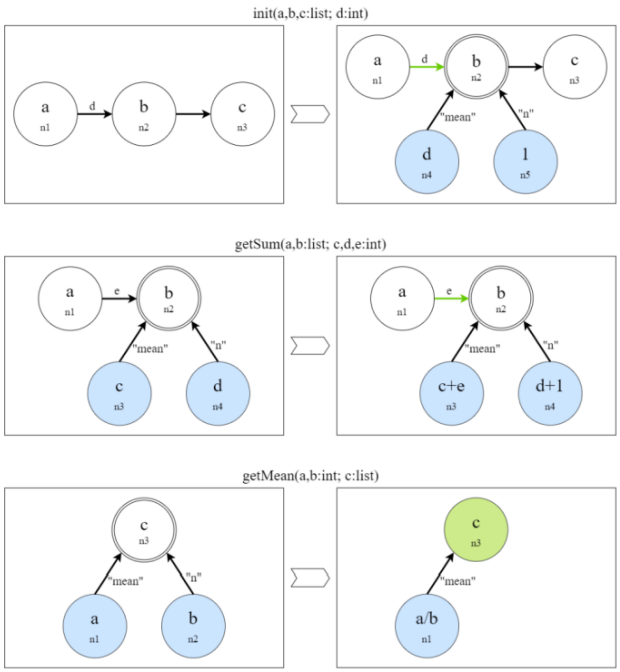
Gambar 9. *Rule Getrecs* Pada Skema 3.

menjadi *node* biasa dan ditandai warna hijau yang menandakan bahwa film tersebut sudah dihitung rata-ratanya. Ketiga *rule* ini akan dijalankan sampai seluruh *node* film sudah ditandai warna hijau, yang berarti seluruh film sudah mempunyai nilai rata-rata masing-masing. Kemudian, akan dijalankan *rule getRecs* yang diilustrasikan pada Gambar 11. *Rule* ini akan menandai semua node film yang memiliki nilai rata-rata di atas batas tertentu ($\geq$80 atau $\geq$60) dan belum pernah dinilai oleh *user input*.

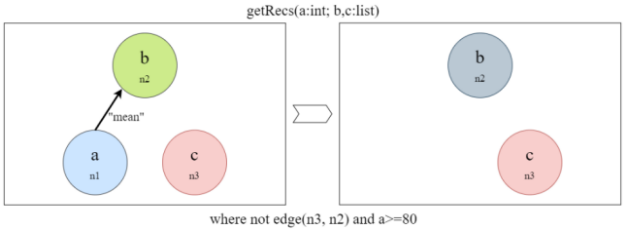
Testing

Metrik evaluasi yang digunakan pada tahap *testing* adalah akurasi, sehingga sebelumnya dilakukan penghitungan banyak *true positive*, *false positive, false negative,* dan *true negative*. Tahap *testing* menggunakan *Python* dan GP 2. *Python* digunakan untuk mengubah graf hasil rekomendasi menjadi graf untuk *input testing* dengan menggabungkannya dengan dataset yang sebelumnya dipisahkan untuk testing pada tahap *preprocessing*. Graf yang sudah digabungkan dengan data untuk *testing* ini kemudian akan dijadikan *input* ke dalam program *testing* GP 2. Program *testing* ini memiliki beberapa *rule* utama. Hal yang pertama dilakukan oleh program ini adalah menghapus film yang tidak berkaitan dengan *user input*.

Setelah itu, dijalankan empat *rule* pada Gambar 12. Keempat *rule* ini digunakan untuk membuat *node* yang akan digunakan sebagai penyimpanan jumlah *true positive, false positive, false negative*, dan *true*



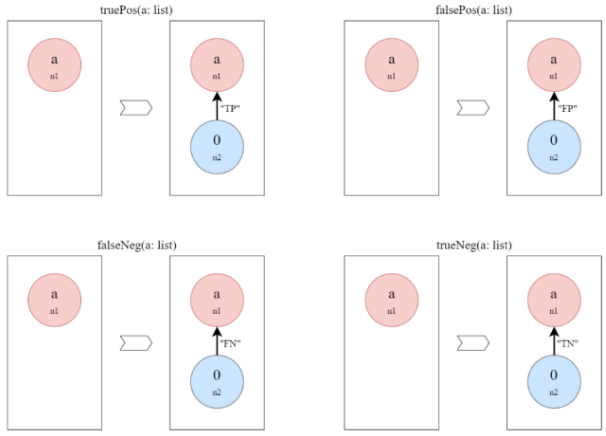
Gambar 10. *Rule Init, Getsum*, dan *Getmean*.



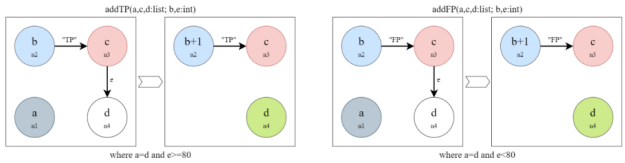
Gambar 11. *Rule Getrecs* Pada Skema 4.

negative dari hasil rekomendasi. *Node* yang dibuat ini akan ditandai dengan warna biru. Berikutnya, terdapat dua *rule* yang dijalankan selama masih memungkinkan yang diilustrasikan pada Gambar 13. *Rule* addTP menghitung semua film hasil rekomendasi yang dinilai bagus oleh *user input* dan menyimpannya pada *node* biru yang memiliki *edge* mengarah ke *user input* berlabel "TP". *Rule* addFP memiliki fungsi yang serupa, tetapi digunakan untuk menghitung nilai *false positive*, yaitu banyaknya hasil rekomendasi yang ternyata dinilai buruk (>80) oleh u*ser input*.

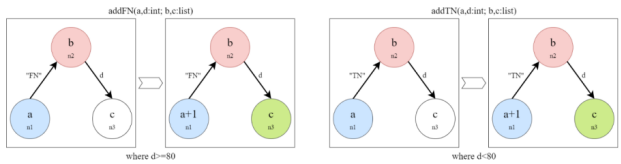
Dilakukan juga dua *rule* yang serupa dengan addTP dan addFP, yaitu addFN dan addTN yang diilustrasikan pada Gambar 14. *Rule* addFN menghitung banyak *film* yang dinilai bagus oleh *user input* tetapi tidak termasuk ke dalam film-film hasil rekomendasi. *Rule* addTN menghitung banyak film yang dinilai buruk oleh *user input* dan tidak termasuk ke dalam rekomendasi. *Rule* addTP, addFP, addFN, dan addTN juga memiliki variasi lain. Di variasi lain ini, batas film yang dinilai bagus oleh sistem adalah 60, bukan 80 seperti pada Gambar 13 dan Gambar 14. Kedua variasi pada empat *rule* ini menyesuaikan dengan variasi yang ada pada sistem rekomendasi pada subbab 3.4.



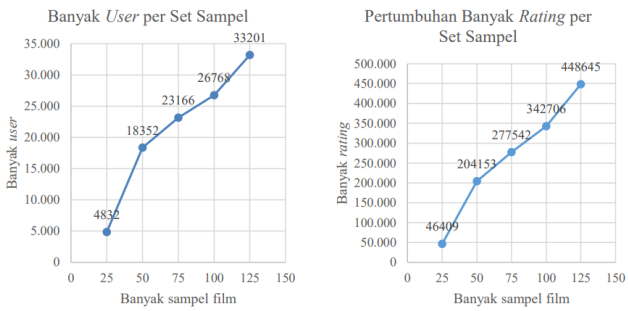
Gambar 12. *Rule* Addtp Dan Addfp.



Gambar 13. *Rule* Addtp Dan Addfp.



Gambar 14. *Rule* Addfn Dan Addtn.



Gambar 15. Banyak *User* Dan *Rating* Untuk Tiap Set Sampel.

Hasil dan Pembahasan

Hasil Pengujian

Hasil pengujian secara umum dapat dibagi menjadi dua, yaitu hasil skenario yang menyatakan nilai yang baik adalah ≥ 60 dan ≥ 80 dari skala rating. Nilai-nilai akurasi dari hasil pengujian untuk batas nilai ≥ 60 diuraikan pada Tabel 1 .

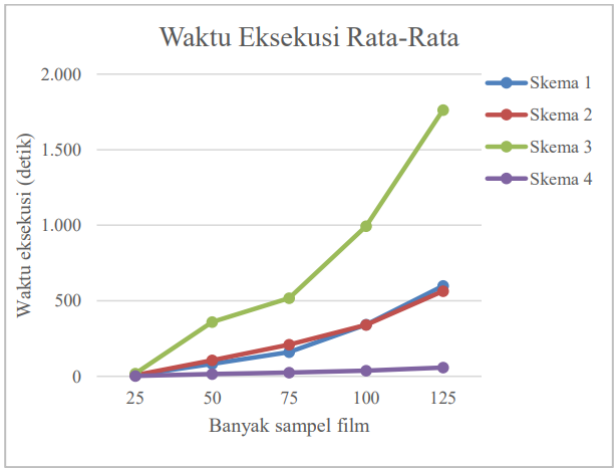
Nilai akurasi dari hasil pengujian untuk batas nilai ≥ 80 diuraikan pada Tabel 2. *User* dan rating yang masuk ke *knowledge graph input* juga memiliki jumlah yang bervariasi untuk tiap set sampel. Pertumbuhan banyak *user* dan rating ini dapat dilihat pada Gambar 15. Sedangkan waktu eksekusi rata-rata sistem rekomendasi untuk tiap

Table 1. Nilai Akurasi Untuk Skenario Batas Nilai ≥ 60

		Sistem Rekomendasi				Rata-Rata
		Skema 1	**Skema 2**	**Skema 3**	**Skema 4**	
Jumlah Sampel Film	25	33,3%	33,3%	66,7%	33,3%	41,7%
	50	60%	60%	40%	60%	55%
	75	40%	60%	40%	80%	55%
	100	50%	50%	50%	66,7%	54,2%
	125	66,7%	66,7%	33,3%	83,3%	62,5%
Rata-Rata		50%	54%	46%	64,7%	

Table 2. Nilai Akurasi Untuk Skenario Batas Nilai ≥ 80

		Sistem Rekomendasi				Rata-Rata
		Skema 1	**Skema 2**	**Skema 3**	**Skema 4**	
Jumlah Sampel Film	25	0%	0%	66,7%	66,7%	33%
	50	20%	20%	80%	80%	50%
	75	40%	40%	60%	60%	50%
	100	16,7%	16,7%	83,3%	83,3%	50%
	125	33,3%	33,3%	66,7%	66,7%	50%
Rata-Rata		22%	22%	**71%**	**71%**	



Gambar 16. Rata-Rata Waktu Eksekusi Sistem Rekomendasi.

skema dapat dilihat pada Gambar 16. Kemudian, karena skema 4 memberikan nilai akurasi yang paling baik, dilakukan eksperimen lebih lanjut pada skema ini dengan menggunakan set sampel film 125. Eksperimen sebelumnya menggunakan *user input* yang sama, yaitu *user* 12. Eksperimen lanjutan ini akan menguji skema 4 dengan lima *user* lain. Hasil eksperimen lanjutan ini diuraikan pada Tabel 3.

Analisis Hasil Pengujian

Skema 1 dan 2 memiliki hasil yang nyaris sama. Hal ini dapat dimengerti, mengingat *rule* yang dipakai pada skema 2 hanya memiliki satu edge tambahan. Kedua skema ini lebih cocok dengan batas nilai ≥ 60. Jika dibandingkan dengan skema lainnya, skema 1 dan 2 ini memiliki akurasi yang buruk. Hal ini kemungkinan terjadi karena kedua skema ini menandai film sebagai rekomendasi jika terdapat satu *user* yang menyukai film tersebut. Jadi jika terdapat film yang tidak disukai semua

Table 3. Nilai Akurasi Skema 4 Pada Eksperimen Lanjutan

		Batas Nilai	
		60	80
User Input	986	36,4%	**72,7%**
	1748	33,3%	**93,3%**
	2082	**90%**	40%
	2177	**100%**	50%
	2982	60%	60%
Rata-Rata		63,9%	63,2%

orang tetapi film tersebut dinilai baik oleh satu *user*, maka film tersebut dapat direkomendasikan dan kemungkinan besar *user input* juga tidak akan menyukai film tersebut.

Skema 3 memberikan akurasi yang cukup buruk dengan batas nilai ≥ 60, tetapi mempunyai salah satu akurasi terbaik dengan batas nilai ≥ 80, yaitu 71%. Namun, skema 3 ini membutuhkan waktu eksekusi yang berkembang dengan cepat seiring jumlah sampel bertambah. Hal ini dapat dilihat pada Gambar 15 yang menunjukkan bahwa eksekusi terhadap 125 sampel skema 3 membutuhkan 29 menit 23 detik (1763 detik) yang naik pesat dari eksekusinya terhadap 100 sampel yang membutuhkan 16 menit 34 detik (994 detik). Hal ini kemungkinan terjadi karena skema 3 memiliki *rule* yang lebih banyak dan lebih kompleks daripada skema 1 dan 2. Karena GP 2 akan mencari subgraf yang cocok untuk dijadikan *left-hand graph* secara satu per satu, maka proses ini akan membutuhkan waktu yang lama. Hal ini membuat skema 3 tidak cocok diimplementasikan untuk data yang besar.

Di sisi lain, skema 4 mempunyai akurasi yang sama dengan skema 3 dengan batas nilai ≥ 80, tetapi dengan waktu yang jauh lebih singkat. Skema 4 ini bahkan memiliki waktu eksekusi paling rendah dibandingkan dengan skema-skema lainnya. Perkembangan waktu eksekusinya pada Gambar 15 pun sangat lambat dibandingkan skema lainnya. Untuk 125 sampel, skema 4 hanya membutuhkan waktu 58 detik, jauh

dari skema 3 yang membutuhkan 29 menit 23 detik. Lambatnya perkembangan waktu eksekusi ini disebabkan oleh penggunaan *rooted node* pada skema 4. Saat *compiler* GP 2 mencari subgraf yang cocok untuk dijadikan *left-hand graph* pada suatu *rule*, *compiler* akan mengutamakan *rooted node* terlebih dahulu. Karena itulah penggunaan *rooted node* pada *rule* di skema 4 ini membuat waktu eksekusi jauh lebih singkat dari skema lainnya walaupun skema 4 ini juga mempunyai *rule* yang lebih banyak dan kompleks.

Node genre juga sebenarnya tidak digunakan pada dalam pemberian rekomendasi pada skema 3 dan 4. Jika *graf input* disederhanakan dengan penghapusan *node genre*, kemungkinan kedua skema ini akan membutuhkan waktu eksekusi yang lebih singkat. Pada Gambar 15, terlihat bahwa banyak *user* dan rating mengalami pertumbuhan yang serupa antara satu sama lain. Dapat dilihat juga bahwa jumlah *user* dan rating mengalami kenaikan paling drastis dari set sampel 25 ke 50. Pada Tabel 1 dan Tabel 2 pun dapat dilihat bahwa selain skema 3 dengan batas nilai ≥ 60, akurasi semua skema mengalami kenaikan dari set sampel 25 ke 50. Selain skema 3 dengan batas nilai ≥ 60 serta skema 3 dan 4 dengan batas nilai ≥ 80 pada 75 sampel, skema lainnya juga hampir tidak pernah menyentuh akurasi di bawah akurasi pada set sampel 25. Nilai akurasi skema 3 dan 4 pada 75 sampel ini pun tidak jauh di bawah akurasi pada sampel 25. Hasil akurasi untuk jumlah sampel 25 juga sangat buruk jika dibandingkan dengan set sampel lainnya. Hal ini menunjukkan bahwa sistem rekomendasi membutuhkan sejumlah data minimum untuk dapat memberikan rekomendasi yang cukup baik.

Eksperimen lanjutan yang dilakukan pada skema 4 juga memberikan hasil yang fluktuatif. Pada Tabel 3, dapat dilihat bahwa selain user 2982 yang mendapat akurasi sama untuk kedua batas nilai, *user* lainnya mempunyai selisih yang cukup jauh antara hasil rekomendasi dengan batas nilai ≥ 60 dan batas nilai ≥ 80. User 2082 dan 2177 mendapat rekomendasi yang bagus dengan batas nilai ≥ 60, sedangkan user 986 dan 1748 mendapat rekomendasi yang bagus dengan batas nilai ≥ 80. Hal ini menunjukkan bahwa sistem rekomendasi yang cocok dengan satu orang belum tentu cocok dengan orang lainnya. Ini juga dapat menjadi alasan nilai akurasi yang bersifat fluktuatif pada Tabel 1 dan Tabel 2.

Kesimpulan

Berdasarkan analisis yang pada Bab 4, sistem rekomendasi yang memberi rekomendasi berdasarkan rating rata-rata film (skema 3 dan 4) lebih baik daripada yang memberi rekomendasi berdasarkan kemiripan preferensi pengguna (skema 1 dan 2). Penggunaan *rooted node* pada program GP 2 juga membuat waktu eksekusi yang jauh lebih singkat. Melihat nilai akurasi yang fluktuatif, pembuatan sistem rekomendasi dengan pengimplementasian GP 2 ini masih perlu diteliti lebih lanjut. Pengimplementasian GP 2 dalam sistem rekomendasi film juga sepertinya masih belum cukup efektif, mengingat pengolahan 125 sampel film saja membutuhkan waktu 1 menit, terlebih lagi jika dibandingkan dengan sistem rekomendasi film lainnya.

Untuk penelitian di topik ini ke depannya, disarankan untuk membuat sistem rekomendasi yang lebih efisien sehingga dapat menangani data yang lebih besar dalam waktu yang lebih singkat. Pembangunan sistem rekomendasi yang lebih efisien ini dapat dilakukan dengan membuat *knowledge graph input* yang lebih sederhana atau dengan penggunaan *rooted node*. Penelitian berikutnya dapat menggunakan *rule* yang berfokus pada hal lain, seperti mengambil *genre* dari film yang dinilai baik oleh *user input*, kemudian memberi rekomendasi film dengan nilai rata-rata terbaik yang juga memiliki *genre* tersebut. Penelitian berikutnya juga dapat memanfaatkan *timestamp* pada dataset untuk memberikan rekomendasi berdasarkan penilaian *user input* yang paling baru, karena mungkin saja terdapat perubahan pada kesukaan *user* seiring berjalannya waktu.

Daftar Pustaka

1. Netflix - Watch TV Shows Online, Watch Movies Online — netflix.com;. [Accessed 04-08-2023]. https://www.netflix.com/.
2. Pradhan R, Swami AC, Saxena A, Rajpoot V. A Study on Movie Recommendations using Collaborative Filtering. IOP Conference Series: Materials Science and Engineering. 2021 mar;1119(1):012018. Available from: https://doi.org/10.1088%2F1757-899x%2F1119%2F1%2F012018.
3. Anwar T, Uma V. Comparative study of recommender system approaches and movie recommendation using collaborative filtering. International Journal of System Assurance Engineering and Management. 2021 apr;12(3):426-36. Available from: https://doi.org/10.1007%2Fs13198-021-01087-x.
4. Ricci F, Rokach L, Shapira B. Introduction to Recommender Systems Handbook. In: Recommender Systems Handbook. Springer US; 2010. p. 1-35. Available from: https://doi.org/10.1007%2F978-0-387-85820-3_1.
5. Plump D. The Design of GP 2. Electronic Proceedings in Theoretical Computer Science. 2012 apr;82:1-16. Available from: https://doi.org/10.4204%2Feptcs.82.1.
6. Verification of graph programs with monadic second-order logic - White Rose eTheses Online — https;. [Accessed 04-08-2023]. https://etheses.whiterose.ac.uk/29370/.
7. Wang Z, Zhang J, Feng J, Chen Z. Knowledge Graph Embedding by Translating on Hyperplanes. Proceedings of the AAAI Conference on Artificial Intelligence. 2014 jun;28(1). Available from: https://doi.org/10.1609%2Faaai.v28i1.8870.
8. Pujara J, Miao H, Getoor L, Cohen W. Knowledge Graph Identification. In: Advanced Information Systems Engineering. Springer Berlin Heidelberg; 2013. p. 542-57. Available from: https://doi.org/10.1007%2F978-3-642-41335-3_34.
9. Martin S, Szekely B, Allemang D. The Rise of the Knowledge Graph. O'Reilly Media, Incorporated; 2021.
10. Tamašauskaitė G, Groth P. Defining a Knowledge Graph Development Process Through a Systematic Review. ACM Transactions on Software Engineering and Methodology. 2023 jan;32(1):1-40. Available from: https://doi.org/10.1145%2F3522586.
11. Wulandari GS, Plump D. Verifying Graph Programs with Monadic Second-Order Logic. In: Graph Transformation. Springer International Publishing; 2021. p. 240-61. Available from: https://doi.org/10.1007%2F978-3-030-78946-6_13.
12. Wulandari GS, Plump D. Verifying Graph Programs with First-Order Logic. Electronic Proceedings in Theoretical Computer Science. 2020 dec;330:181-200. Available from: https://doi.org/10.4204%2Feptcs.330.11.
13. Pavitha N, Pungliya V, Raut A, Bhonsle R, Purohit A, Patel A, et al. Movie recommendation and sentiment analysis using machine learning. Global Transitions Proceedings. 2022 jun;3(1):279-84. Available from: https://doi.org/10.1016%2Fj.gltp.2022.03.012.
14. Han W, Wang Q. Movie recommendation algorithm based on knowledge graph. In: 2019 2nd International Conference on Safety Produce Informatization (IICSPI). IEEE; 2019. Available from: https://doi.org/10.1109%2Fiicspi48186.2019.9095901.
15. Bak C. GP 2: efficient implementation of a graph programming language. University of York; 2015.