Anime Recommendator: Rekomendasi untuk Animasi Berbasis AnimeNewsNetwork

Laporan Tugas Ontologi IF4070

Semester I Tahun 2014 / 2015

Kelompok WbTeladan

oleh :

Iskandar Setiadi / 13511073

Faiz Ilham / 13511080

M Andri Eka Fauzy / 13511088

Sekolah Teknik Elektro dan Informatika (STEI ITB)

Institut Teknologi Bandung

Jl. Ganesha No. 10, Bandung 40132

Tahun 2014

I. Latar Belakang

Seiring perkembangan zaman, teknologi internet memungkinkan persebaran media hiburan dengan cepat, salah satunya adalah animasi Jepang (*anime*). Pada prinsipnya, *anime* memiliki beberapa karakteristik yang cukup dekat dengan *movie*, namun terdapat beberapa aspek tambahan seperti desainer grafis (*graphic designer*), sumber adaptasi (*visual novel*, dll), dan aspek lainnya.

AnimeNewsNetwork (animenewsnetwork.com) adalah website yang didirikan oleh Justin Sevakis pada tahun 1998 dan didesain sebagai katalog untuk *anime*. AnimeNewsNetwork memiliki ± 20.000 entri yang disediakan oleh pengguna dan telah mendapat moderasi dari pengelola *website*. Data yang tersedia pada AnimeNewsNetwork mencakup judul, jenis media (*anime* atau *manga*), genre, staf yang terlibat, forum terkait, dan berita terkait.

Banyaknya jumlah film animasi yang muncul pada setiap quartal dan terbatasnya waktu yang dimiliki untuk menonton menyebabkan penonton untuk memilih jenis animasi yang ingin mereka tonton. Kami memutuskan untuk membuat sebuah sistem rekomendasi animasi untuk menyelesaikan permasalahan mengenai animasi jenis apa yang sebaiknya ditonton.

II. Solusi Permasalahan

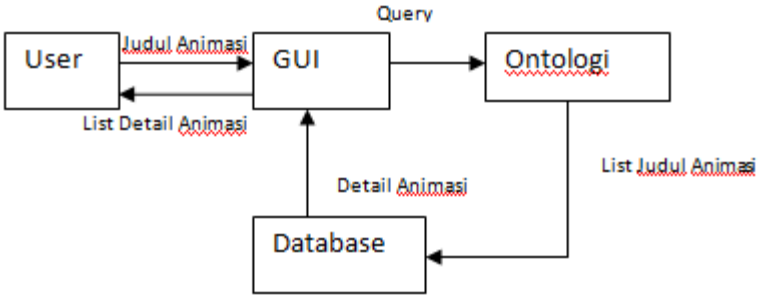
A. System Requirements

Berdasarkan permasalahan diatas dapat disimpulkan bahwa permasalahan yang ada adalah sebagai berikut :

1. Diperlukan sebuah sistem rekomendasi animasi
2. User dapat mengatur jenis kemiripan yang diinginkan

B. Rancangan Sistem

Sistem rekomendasi yang akan dibuat akan berbentuk sebuah halaman web yang menerima masukan dari pengguna berupa judul dari animasi yang disukai. Selanjutnya sistem akan mencari animasi dengan ontologi yang mirip dengan animasi yang disukai kemudian menampilkan secara terurut berdasarkan *score* yang dimiliki dari tertinggi. *User* kemudian dapat menambahkan parameter pencarian seperti tidak mengikutsertakan kategori tertentu untuk inferensi ontologi. Judul dari animasi yang terdapat didalam sistem adalah animasi yang terdapat didalam halaman website AnimeNewsNetwork yang telah disimpan dalam format RDF/XML-ABBREV bersamaan dengan model ontologi yang telah dibuat sebelumnya.

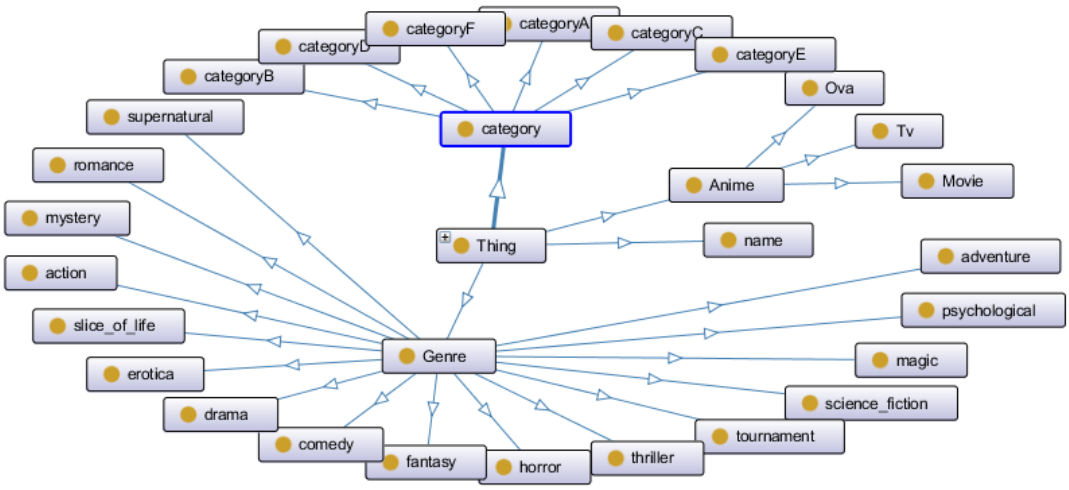


Gambar 1. Arsitektur Sistem *Anime Recommendator*

III. Implementasi

A. Ontologi yang Dibangun

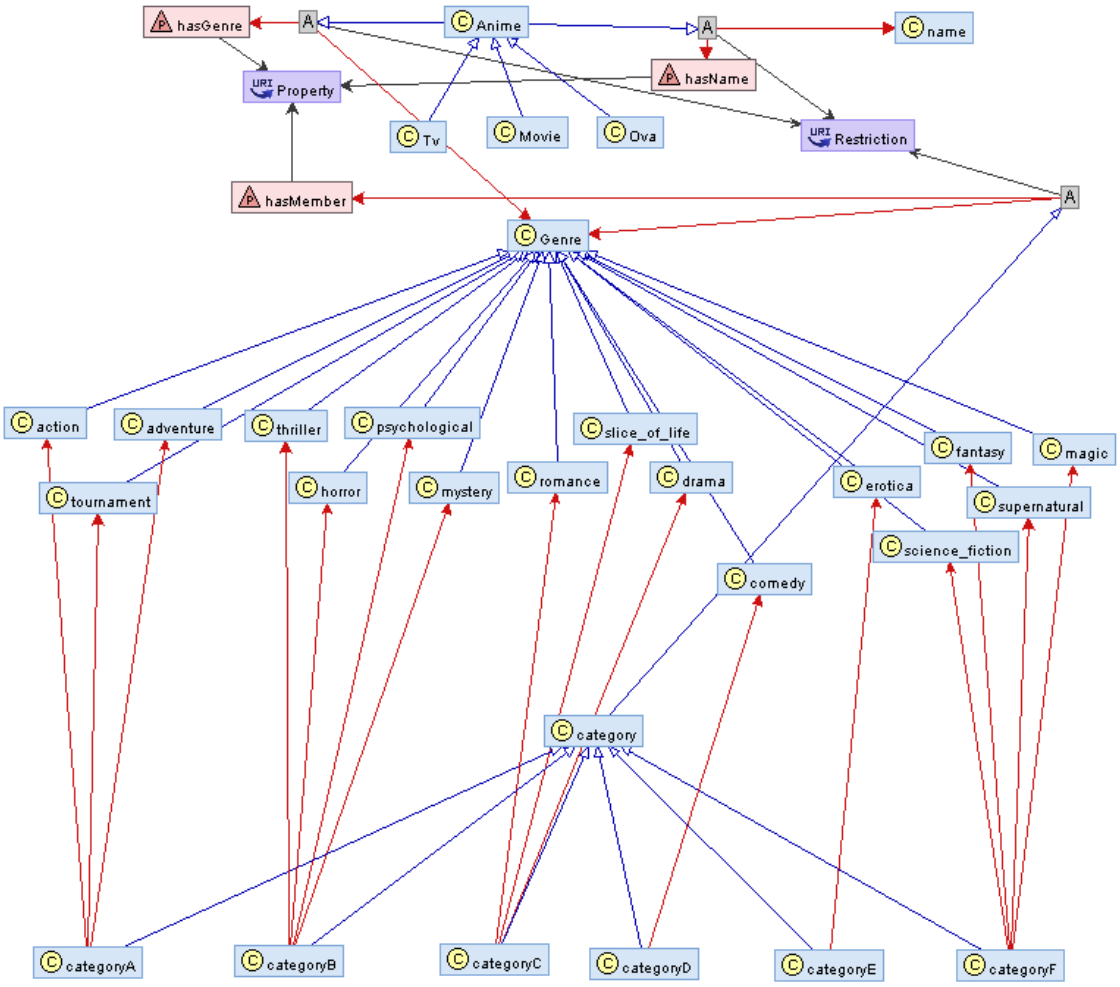
Ontologi yang akan digunakan berisikan dengan 4 hal yaitu kelas Anime, kelas Category kelas Genre dan kelas Name. Kelas Anime berisikan entitas individu dari *anime*. Berdasarkan data yang didapatkan dari AnimeNewsNetwork kelas Anime dapat dibagi menjadi 3 kelas berbeda berdasarkan jenis dari *anime*nya. Kelas Genre berisikan semua genre yang mungkin terdapat didalam sebuah *anime*. Berdasrkan data yang dimiliki kelas Genre dapat diturunkan menjadi 16 kelas berbeda. Kelas Category merupakan kelas yang menyimpan keterkaitan antara suatu genre dengan genre lainnya. Berdasarkan hasil pengamatan suatu animasi memiliki lebih dari satu genre dan terdapat beberapa genre yang muncul secara bersamaan sehingga dari hasil pengamatan tersebut dibuat beberapa kategori kemunculan genre. Kelas Name adalah nama dari entitas animasi.



Gambar 2. Model Ontologi yang Digunakan

Relasi antara kategori dengan genre dapat dilihat pada gambar berikut ini. Kelas Anime memiliki relasi *hasGenre* dengan kelas genre dan memiliki relasi *hasName* dengan kelas

name.Kelas Anime sendiri memiliki beberapa anak kelas. Kelas Category memiliki restriksi *hasMember* dengan kelas Genre dan memiliki beberapa anak kelas. Kelas genre sendiri memiliki beberapa anak kelas.



Gambar 3. Konstrain antar Kelas dalam Model Ontologi

Setiap *anime* akan disimpan dalam bentuk entitas *Individual* seperti contoh berikut ini:

```
<j.0:Tv
rdf:about="http://www.semanticweb.org/lenovo/ontologies/2014/12/Guyver
_(TV)">
    <j.0:hasGenre
rdf:resource="http://www.semanticweb.org/lenovo/ontologies/2014/12/adv
enture"/>
    <j.0:hasName>Guyver_(TV)</j.0:hasName>
  </j.0:Tv>
```

B. Query yang Digunakan

```
// Mapping a title to its genre -> get all category -> retrieve all genres in
all categories
        queryString = queryString
                + "SELECT DISTINCT ?anime_result_name WHERE {"
                + "?anime_entity
<http://www.semanticweb.org/lenovo/ontologies/2014/12/hasName>  \""
                + anime_title
                + "\" . "
                + "?anime_entity
<http://www.semanticweb.org/lenovo/ontologies/2014/12/hasGenre> ?genre . "
                + "?category
<http://www.semanticweb.org/lenovo/ontologies/2014/12/hasMember> ?genre . "
                + "?category
<http://www.semanticweb.org/lenovo/ontologies/2014/12/hasMember>
?related_genre . "
                + "?anime_result_entity
<http://www.semanticweb.org/lenovo/ontologies/2014/12/hasGenre>
?related_genre . ";

        // Find whether TV / OVA / Movie in Title
        queryString = queryString + checkSubClass(anime_title);
        // Retrieve Anime name
        queryString = queryString
                + "?anime_result_entity
<http://www.semanticweb.org/lenovo/ontologies/2014/12/hasName>
?anime_result_name ";

        // Subtract original title from result set
        queryString = queryString
                + "MINUS { ?anime_result_entity
<http://www.semanticweb.org/lenovo/ontologies/2014/12/hasName>  \""
                + anime_title + "\" }}";
```

Gambar 4. Query SPARQL untuk Inferensi dalam Ontologi

Query digunakan untuk mendapatkan Genre dari suatu judul *anime*, kemudian dicari kategori dari genre tersebut. Dari kategori-kategori yang didapatkan, sistem akan melakukan inferensi untuk mendapatkan semua *anime* yang masuk kedalam kategori-kategori tersebut. Kemudian query ini akan mencari judul *anime* dengan jenis yang sama dengan *anime* yang yang disukai dengan melakukan inferensi terhadap kategori yang serupa / mirip.

C. Komunikasi antara Query (Java) dengan GUI (Python)

Setelah *queries* SPARQL tersebut diimplementasikan dalam Apache Jena (Java), tahapan selanjutnya dari implementasi ini adalah mendesain komunikasi antara hasil inferensi dengan tampilan pengguna (Python).

Pada bagian III.A dan III.B, kita telah mengimplementasikan sistem inferensi untuk *query* ontologi dalam Apache Jena. Hasil implementasi ini disimpan dalam sebuah *file executable* .jar yang dapat dijalankan dengan menerima beberapa parameter masukan, yaitu judul *anime* dan filter yang digunakan. *Executable* ini akan mengembalikan judul-

judul *anime* hasil inferensi dalam format JSON, yang akan digunakan oleh implementasi Python-*side* untuk menampilkan rekomendasi animasi ke pengguna.

Untuk lebih jelasnya, perhatikan kode dibawah ini:

Java-*side*

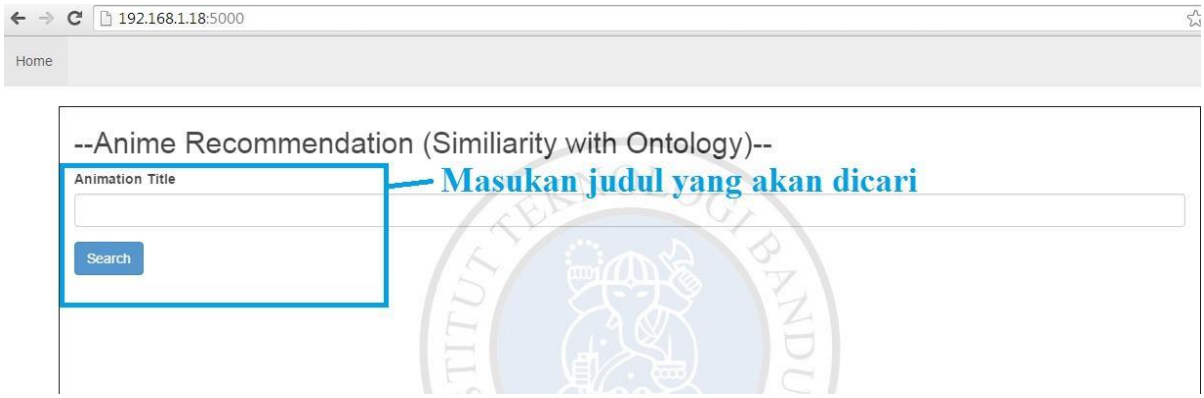
```
String queryString = QueryHelper.composeQuery(anime_title, args);

Query query = QueryFactory.create(queryString);

// Execute the query and obtain results
QueryExecution qe = QueryExecutionFactory.create(query, ontModel);
ResultSet results = qe.execSelect();

// Output query results
ResultSetFormatter.outputAsJSON(results);

// Free up resources used running the query
qe.close();
```

Python-*side*

```
def inference(Title, param=None):
    parameters = ['java', '-jar', './lib/animeRecommendation.jar',
Title]
    if param != None:
        parameters = parameters + param
    p = Popen(parameters, stdout=PIPE, stderr=STDOUT)
    output = ''
    for line in p.stdout:
        output = output + line
    j = json.loads(output)
    return j
```

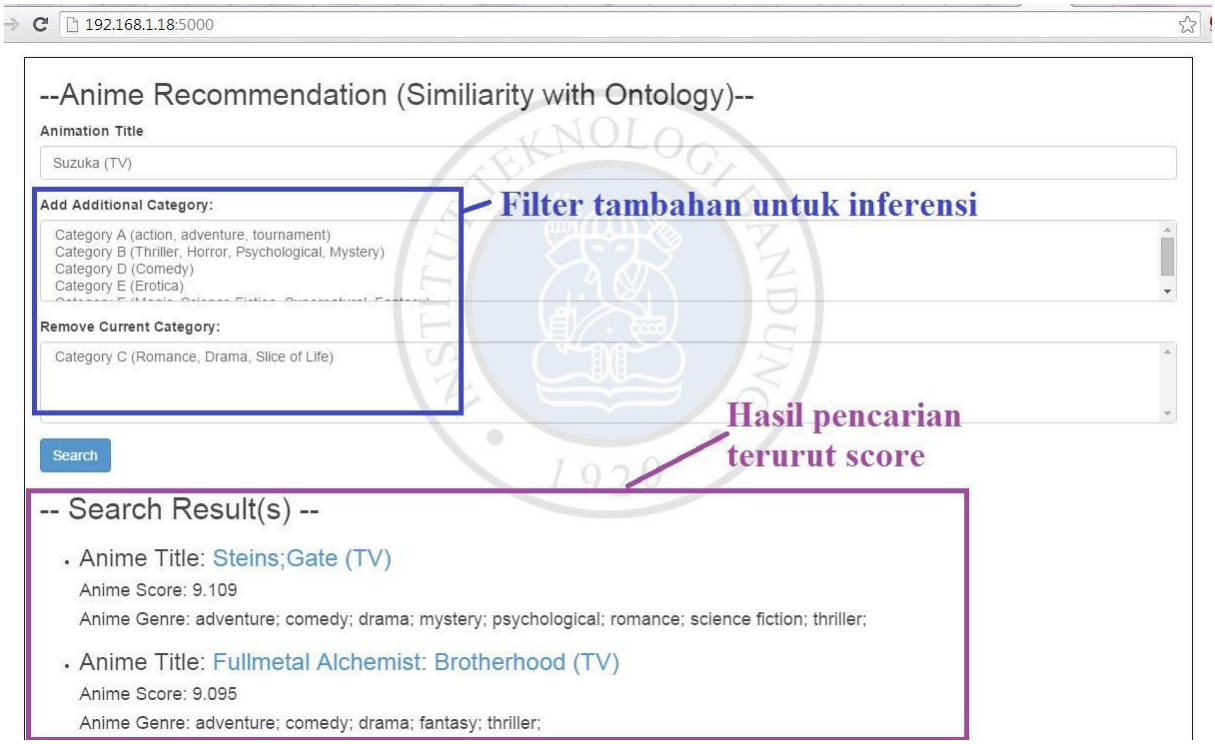
D. Hasil Aplikasi

Hasil implementasi dari aplikasi ini adalah sebuah halaman web seperti pada gambar dibawah. Terdapat kolom *Animation Title* yang dapat diisikan dengan judul animasi yang disukai. Kolom ini memiliki fitur *auto complete* sehingga *user* dapat menghindari kesalahan penulisan judul. Kemudian tombol *search* digunakan untuk mendapatkan rekomendasi animasi. Rekomendasi didapatkan dari query yang telah dijelaskan sebelumnya.



Gambar 5. Tampilan Awal dari *UI* Pencarian

Terdapat parameter tambahan yaitu untuk menambahkan parameter query dengan mengatur kategori genre apa yang ingin direkomendasikan atau yang tidak ingin direkomendasikan.



Gambar 6. Hasil Inferensi Ontologi dengan Opsi untuk Filter Tambahan

IV. Analisis

Dengan menggunakan ontologi, proses pencarian rekomendasi menggunakan query SPARQL lebih mudah untuk dilakukan dibandingkan query menggunakan SQL pada basis data relasional. Hal ini disebabkan apabila menggunakan SQL diperlukan operasi melakukan *join table* pada tabel Genre, tabel Category, dan tabel Anime untuk mendapatkan hasil yang sama dengan menggunakan ontologi.

Sebagai contoh, animasi A memiliki 3 genre, yaitu Action, Drama, dan Comedy. Ketiga genre ini akan diinferensikan kedalam 3 kategori berbeda, yaitu Category A, Category C, dan Category D. Masing-masing Category ini memiliki satu atau lebih genre sejenis. Dalam kasus ini, terdapat 7 buah genre yang akan menjadi hasil inferensi. Pada proses selanjutnya, 7 buah genre ini akan digunakan untuk mencari judul animasi terkait yang akan direkomendasikan ke pengguna. Jika kita menggunakan basis data relasional, operasi ini membutuhkan *query* yang menggabungkan 3 buah tabel berbeda. Apabila terdapat 20.000 buah animasi, 20 kelas Genre, dan 10 kelas Category, maka operasi *join table* akan menghasilkan jutaan *record* berbeda. Hal ini tentunya tidak efisien dari segi memori maupun waktu. Selain itu, basis data relasional tidak *scalable* untuk kasus ini, dimana jumlah animasi > 1.000.000 akan membuat proses pencarian berlangsung dalam waktu yang sangat lama. Dengan implementasi menggunakan skema ontologi, proses rekomendasi animasi menggunakan query SPARQL tidak memerlukan operasi *join table* sehingga menghemat memori dalam proses pencariannya.

Keuntungan lain dari penggunaan ontologi adalah basis data relasional tidak dapat melakukan inferensi untuk mengetahui bahwa TV, movie, OVA dll terhubung sebagai anak dari kelas (*subclass*) Anime. Seluruh kelas tersebut akan dianggap tabel yang berbeda oleh basis data relasional. Dengan ontologi, kita dapat mendefinisikan dan mengelompokkan suatu Anime kedalam *subclass* berbeda dengan mudah (struktur RDF/OWL yang mendukung).

V. Kesimpulan

Aplikasi yang dibuat telah memenuhi *user requirements* yang dimiliki, tetapi untuk meningkatkan hasil rekomendasi yang diberikan diperlukan modifikasi lanjutan. Modifikasi dapat berupa penambahan data yang dimiliki dan menambahkan kelas turunan pada kelas Genre dan kelas Category pada ontologi yang dimiliki. Hal ini disebabkan karena terdapat beberapa kelas yang masih memiliki properti kurang spesifik. Selain itu, penggunaan ontologi lebih baik dibandingkan penggunaan SQL untuk kasus semacam ini (*anime recommendator*).

Keterangan tambahan: hasil implementasi dapat diakses disini.

VI. Referensi

[1] Apache Jena. Diakses 1 Desember 2014 pukul 18.30.
<https://jena.apache.org/>

[2] Flask (A Python Framework). Diakses 1 Desember 2014 pukul 19.00.
<http://flask.pocoo.org/>

[3] Protege. Diakses 2 Desember 2014 pukul 20.00.
<http://protege.stanford.edu/>

[4] RDF Gravity (RDF Graph Visualization Tool). Diakses 10 Desember 2014 pukul 20.00.
<http://semweb.salzburgresearch.at/apps/rdf-gravity/>

[5] SPARQL 1.1 Query Language. Diakses 2 Desember 2014 pukul 20.30.
<http://www.w3.org/TR/sparql11-query/>