TUGAS AKHIR – IF184802

EKSTRAKSI ENTITAS GEOSPATIAL PADA ARTIKEL WIKIPEDIA BERTIPE KEJADIAN

RIZVI SOFBRINA
NRP 05111640000037

Dosen Pembimbing 1
Nurul Fajrin Ariyani, S.Kom., M.Sc.

Dosen Pembimbing 2
Abdul Munif, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

TUGAS AKHIR – IF184802

EKSTRAKSI ENTITAS GEOSPATIAL PADA ARTIKEL WIKIPEDIA BERTIPE KEJADIAN

RIZVI SOFBRINA
NRP 05111640000037

Dosen Pembimbing 1
Nurul Fajrin Ariyani, S.Kom., M.Sc.

Dosen Pembimbing 2
Abdul Munif, S.Kom., M.Sc.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

UNDERGRADUATE THESIS – IF184802

EXTRACTION OF GEOSPATIAL ENTITIES IN WIKIPEDIA ARTICLES TYPES OF EVENTS

RIZVI SOFBRINA
NRP 05111640000037

Supervisors 1
Nurul Fajrin Ariyani, S.Kom., M.Sc.

Supervisors 2
Abdul Munif, S.Kom., M.Sc.

Department of Informatics
Faculty of Intelligent Electrical and Informatics Technology
Institut Teknologi Sepuluh Nopember
Surabaya 2020

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

EKSTRAKSI ENTITAS GEOSPATIAL PADA ARTIKEL WIKIPEDIA BERTIPE KEJADIAN

TUGAS AKHIR

Diajukan Guna Memenuhi Salah Satu Syarat Memperoleh
Gelar Sarjana Komputer
pada
Rumpun Mata Kuliah Manajemen Informasi
Program Studi S-1 Departemen Teknik Informatika
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember

Oleh:

RIZVI SOFBRINA
NRP: 05111640000037

Disetujui oleh Dosen Pembimbing Tugas Akhir:

1 Nurul Fajrin Ariyani, S.Kom., M.Sc. .....
NIP. 198607222015042003 (Pembimbing 1)

2 Abdul Munif., S.Kom., M.Sc. .....
NIP. 198608232015041004 (Pembimbing 2)

SURABAYA
JUNI 2020

[Halaman ini sengaja dikosongkan]

EKSTRAKSI ENTITAS GEOSPATIAL PADA ARTIKEL WIKIPEDIA BERTIPE KEJADIAN

Nama Mahasiswa	: Rizvi Sofbrina
NRP	: 05111640000037
Departemen	: Teknik Informatika ITS
Dosen Pembimbing 1	: Nurul Fajrin Ariyani, S.Kom., M.Sc.
Dosen Pembimbing 2	: Abdul Munif, S.Kom., M.Sc.

ABSTRAK

Wikipedia merupakan ensiklopedia yang sangat banyak digunakan. Pada Wikipedia sendiri sudah tersedia jutaan artikel terkait peristiwa atau kejadian. Untuk mempermudah memodelkan informasi menjadi terstruktur dapat memanfaatkan metadata yang telah disediakan oleh DBpedia.org. Namun, hingga saat ini masih terdapat metadata artikel yang belum dapat memberikan informasi yang mudah diperoleh oleh pengguna termasuk informasi spasial yang berada pada artikel bertipe kejadian. Oleh karena itu, maka pada tugas akhir ini akan dibuat suatu aplikasi yang dapat membantu orang dalam mendapatkan informasi spasial pada artikel Wikipedia bertipe kejadian.

Dalam pembuatan aplikasi diperlukan ekstraksi entitas geospasial. Masalah ini dapat diatasi dengan melakukan beberapa tahap proses ekstraksi. Proses ekstraksi melibatkan tiga tahap, yaitu ekstraksi DBpedia, ekstraksi infobox Wikipedia dan ekstraksi raw text deskripsi utama Wikipedia. Hasil ekstraksi disimpan pada triple store Apache Jena Fuseki dalam bentuk pasangan triplets yang merupakan entitas data dengan susunan subject-predicate-object, dengan ontology yang sudah disediakan oleh DBpedia. Pasangan triplets ini selanjutnya dapat digunakan untuk menjawab query tentang spasial suatu artikel Wikipedia.

Hasil dari pengerjaan tugas akhir ini adalah memperluas jangkauan ekstraksi entitas geospasial sehingga entitas geospasial yang terdapat pada plain text dapat dikenali. Berdasarkan uji coba yang dilakukan, hasil query spasial yang dilakukan pada triple store Apache Jena Fuseki aplikasi ini memberikan hasil lebih banyak dibandingkan hasil query pada SPARQL Endpoint DBpedia dikarenakan entitas yang diperoleh tidak bersumber pada DBpedia saja namun juga dari infobox dan raw text deskripsi utama Wikipedia.

Kata Kunci— Apache Jena Fuseki, Dbpedia, Geospasial, Ontologi, SPARQL, Wikipedia.

EXTRACTION OF GEOSPATIAL ENTITIES IN WIKIPEDIA ARTICLES TYPES OF EVENTS

Student Name	: Rizvi Sofbrina
NRP	: 05111640000037
Department	: Informatics ITS
Supervisor 1	: Nurul Fajrin Ariyani, S.Kom., M.Sc.
Supervisor 2	: Abdul Munif, S.Kom., M.Sc.

ABSTRACT

Wikipedia is a widely used encyclopedia. On Wikipedia itself, there are articles related to events. To make it easier to model information into a structured form, it can utilize the metadata provided by dbpedia.org. However, until now, there are still metadata of event type articles that have not been able to provide information that is easily obtained by users, including spatial information of the articles. Therefore, this final project will create an application that can help people in getting spatial information on Wikipedia articles of type events.

In making the application required the extraction of geospatial entities. This problem can be overcome by doing several stages of the extraction process. The extraction process has done in three ways: DBpedia extraction, Wikipedia infobox extraction, the raw text extraction of the main description in Wiki's articles. The extraction results are stored in Apache Jena Fuseki triplestore within the form of triplets, which consisted of data entities in a subject-predicate-object arrangement. It used schema or ontology that has been provided by DBpedia. These triplets can then be used to answer queries about spatial Wikipedia articles.

The purpose of this work is to expand the coverage of geospatial entities extraction in raw text. The testing process is done by comparing the results obtained from this work with SPARQL Endpoint Dbpedia in the same keywords of query. Based

on testing, the results of spatial queries carried out in the Apache Jena Fuseki of this application gives more results than the query result on SPARQL Endpoint DBpedia because the entity was obtained not only from DBpedia but also from infobox and raw text extraction of the main description in articles of Wiki.

Keywords— Apache Jena Fuseki, DBpedia, Geospatial, Ontology, SPARQL, Wikipedia.

KATA PENGANTAR

Puji syukur saya ucapkan kepada Allah Yang Maha Kuasa. Karena atas karunia serta rahmat-Nya saya dapat menyelesaikan tugas akhir yang berjudul:

EKSTRAKSI ENTITAS GEOSPATIAL PADA ARTIKEL WIKIPEDIA BERTIPE KEJADIAN

Selesainya tugas akhir ini tidak lepas dari bantuan dan dukungan beberapa pihak. Sehingga pada kesempatan ini penulis mengucapkan syukur dan terima kasih kepada:

1. Allah SWT Yang Maha Kuasa, karena limpahan rahmat dan karunia-Nya penulis dapat menjalankan perkuliahan di Departemen Informatika Institut Teknologi Sepuluh Nopember dan dapat menyelesaikan tugas akhir guna memenuhi syarat kelulusan sebagai Sarjana.
2. Keluarga tercinta, Bapak dan Ibu yang telah memberikan dukungan moral dan material serta doa yang tak terhingga untuk penulis.
3. Ibu Nurul Fajrin Ariyani, S.Kom., M.Sc. selaku dosen pembimbing I yang telah memberi dukungan, semangat, dan motivasi selama masa perkuliahan penulis, serta mengorbankan waktu dan tenaga untuk membimbing penulis dalam menyelesaikan tugas akhir ini.
4. Bapak Abdul Munif, S.Kom., M.Sc. selaku pembimbing II yang telah memberi masukan dan saran serta mengorbankan waktu dan tenaga untuk membimbing penulis dalam menyelesaikan tugas akhir ini.
5. Segenap dosen dan karyawan Departemen Informatika ITS yang telah memberikan ilmu dan pengalaman kepada penulis selama menjalani masa kuliah di Informatika ITS.
6. Meutia, April, Fina, Yasinta, Rifka, Farida dan Sari yang berjuang bersama dan menemani penulis dari awal perkuliahan hingga sekarang.

7. Teman-teman seperjuangan tugas akhir; yang selalu membantu menemukan solusi selama pengerjaan Tugas Akhir.
8. Liah, Syifa, dan Ulfa selaku teman seperantauan yang selalu memberikan penulis semangat dan menghilangkan rasa lelah selama mengerjakan TA.
9. Khairunnisa' Rahma Mardiyani sebagai teman seperjuangan tugas akhir yang selalu memberi semangat dan bantuan dalam pengerjaan tugas akhir ini.
10. Teman-teman Informatika ITS angkatan 2016 yang tidak bisa disebutkan namanya satu persatu.
11. Serta semua pihak yang yang telah turut membantu penulis dalam menyelesaikan tugas akhir ini.

Penulis memohon maaf jika terdapat kesalahan maupun kekurangan pada tugas akhir ini. Oleh karena itu, dengan segala kerendahan hati penulis mengharapkan kritik dan saran yang membangun dapat disampaikan sebagai bahan perbaikan untuk kedepannya. Semoga laporan tugas akhir ini dapat berguna bagi pembaca.

Surabaya, Juni 2020

Rizvi Sofbrina

DAFTAR ISI

LEMBAR PENGESAHAN..... Error! Bookmark not defined.
ABSTRAK ix
ABSTRACT xi
KATA PENGANTAR..... xiii
DAFTAR ISI xv
DAFTAR GAMBAR..... xix
DAFTAR TABEL xxiii
DAFTAR KODE SUMBER.....xxv
1. BAB I PENDAHULUAN..... 1
1.1 Latar Belakang 1
1.2 Rumusan Masalah 2
1.3 Batasan Masalah..... 3
1.4 Tujuan..... 3
1.5 Manfaat..... 3
1.6 Metodologi..... 3
1.7 Sistematika Penulisan 6
2. BAB II TINJAUAN PUSTAKA 9
2.1 Geospasial..... 9
2.2 Metadata..... 9
2.3 DBpedia 10
2.4 *Ontology* 11
2.5 GeoNames..... 11
2.6 *Linked Open Data* 12

2.7 *Natural Language Processing* 12
2.8 SPARQL 12
2.9 Apache Jena Fuseki 13
2.10 Flask 14
2.11 Penelitian Terdahulu 15
2.12 *Confusion Matrix* 16
3. BAB III ANALISIS DAN PERANCANGAN 19
3.1 Analisis Permasalahan 19
3.2 Deskripsi Umum Sistem 19
3.3 Kasus Penggunaan Sistem 22
3.3.1 Memasukkan Data Artikel DBpedia 23
3.3.2 Mencari Data Artikel DBpedia 25
3.4 Analisis dan Perancangan Data 26
3.5 Perancangan Proses 27
3.6 Perancangan Antarmuka Sistem 28
3.7 Evaluasi dan Uji Coba 31
4. BAB IV IMPLEMENTASI 33
4.1 Lingkungan Implementasi 33
4.2 Persiapan dan Pengambilan Data 34
4.3 Implementasi Proses 35
4.3.1 Implementasi Ekstraksi DBpedia 35
4.3.2 Implementasi Ekstraksi *Infobox* 37
4.3.3 Implementasi Ekstraksi Deskripsi Artikel 41

4.3.4 Implementasi Pencarian Nama Tempat pada Geonames dan DBpedia 42
4.3.5 Implementasi Pembuatan *Dataset* pada Apache Jena Fuseki 44
4.3.5.1 Pembuatan *Dataset* Baru pada Apache Jena Fuseki 44
4.3.5.2 Penyimpanan Data pada Apache Jena Fuseki. 45
4.3.6 Implementasi *Insert* Data ke Apache Jena Fuseki 53
4.3.7 Implementasi *Query* dari Apache Jena Fuseki ... 59
4.4 Implementasi Antarmuka Sistem 62
5. BAB V UJI COBA DAN EVALUASI 67
5.1. Lingkungan Uji Coba 67
5.2. Data Uji Coba..... 67
5.3. Skenario Uji Coba 67
5.3.1 Skenario Pengujian 1 68
5.3.2 Skenario Pengujian 2 74
5.3.3 Skenario Pengujian 3 80
5.3.4 Skenario Pengujian 4 83
5.4. Evaluasi..... 85
6. BAB VI KESIMPULAN DAN SARAN 87
6.1 Kesimpulan 87
6.2 Saran 88
DAFTAR PUSTAKA..... 89
LAMPIRAN 91

BIODATA PENULIS..... 95

DAFTAR GAMBAR

Gambar 1.1 Artikel Kejadian Tragedi Trisakti pada Wikipedia... 1
Gambar 1.2 Metadata Artikel Kejadian Tragedi Trisakti pada DBpedia 2
Gambar 2.1 Contoh *infobox* Wikipedia 9
Gambar 2.2 Contoh Event DBpedia 11
Gambar 2.3 Contoh *query editor* 13
Gambar 2.4 Hasil dari *query* 14
Gambar 2.5 Daftar API Apache Jena Fuseki..... 14
Gambar 2.6 *Confusion Matrix*..... 17
Gambar 3.1 Arsitektur Sistem 20
Gambar 3.2 Arsitektur sistem perangkat lunak 22
Gambar 3.3 Diagram kasus penggunaan..... 22
Gambar 3.4 Contoh data *triplets*..... 27
Gambar 3.5 Diagram alir seluruh tahapan..... 27
Gambar 3.6 Rancangan antarmuka halaman beranda aplikasi ... 29
Gambar 3.7 Rancangan antarmuka halaman *input* data..... 29
Gambar 3.8 Rancangan antarmuka halaman *query* data 30
Gambar 3.9 Rancangan antarmuka halaman hasil ekstraksi 30
Gambar 4.1 Hasil Ekstraksi DBpedia pada Artikel *Iberian War* 37
Gambar 4.2 Hasil Ekstraksi Deskripsi Artikel *2000 Turkmenistan earthquake*..... 42
Gambar 4.3 Hasil Ekstraksi *2000 Turkmenistan earthquake* dengan ID Geonames 44
Gambar 4.4 Hasil Ekstraksi *2000 Turkmenistan earthquake* dengan DBpedia 44
Gambar 4.5 Pembuatan *dataset* baru pada Apache Jena Fuseki. 45
Gambar 4.6 Contoh *triplets* pada Apache Jena Fuseki 53
Gambar 4.7 Contoh data *input*..... 59
Gambar 4.8 Contoh hasil *insert* data ke Apache Jena Fuseki..... 59
Gambar 4.9 Contoh kata kunci pencarian 61

Gambar 4.10 Contoh *output* hasil *query* dari Apache Jena Fuseki 62
Gambar 4.11 Halaman beranda aplikasi 62
Gambar 4.12 Halaman *input* data 63
Gambar 4.13 Pesan insert data pada Apache Jena Fuseki berhasil 64
Gambar 4.14 Halaman *query* data 64
Gambar 4.15 Halaman hasil *query* data 64
Gambar 4.16 Halaman Geonames Italy 65
Gambar 4.17 Halaman DBpedia Italy 65
Gambar 5.1 Query SPARQL Data Uji 1 69
Gambar 5.2 Hasil Query Apache Jena Fuseki Data Uji 1 69
Gambar 5.3 Hasil Query SPARQL Endpoint DBpedia Data Uji 1 70
Gambar 5.4 Query SPARQL Data Uji 2 70
Gambar 5.5 Hasil Query Apache Jena Fuseki Data Uji 2 71
Gambar 5.6 Hasil Query SPARQL Endpoint DBpedia Data Uji 2 71
Gambar 5.7 Query SPARQL Data Uji 3 72
Gambar 5.8 Hasil Query Apache Jena Fuseki Data Uji 3 72
Gambar 5.9 Hasil Query SPARQL Endpoint DBpedia Data Uji 3 73
Gambar 5.10 Contoh data artikel 74
Gambar 5.11 Hasil ekstraksi Data Uji 4 76
Gambar 5.12 Hasil ekstraksi Data Uji 5 77
Gambar 5.13 Hasil ekstraksi Data Uji 6 78
Gambar 5.14 Grafik evaluasi hasil ekstraksi menggunakan semua data uji 80
Gambar 5.15 Hasil *input* Data Uji 9 yang tersimpan pada Apache Jena Fuseki 81
Gambar 5.16 Hasil *input* Data Uji 10 yang tersimpan pada Apache Jena Fuseki 82

Gambar 5.17 Hasil *input* Data Uji 11 yang tersimpan pada Apache Jena Fuseki 82

[Halaman ini sengaja dikosongkan]

DAFTAR TABEL

Tabel 2.1 Penelitian terdahulu 15
Tabel 3.1 Daftar kasus penggunaan 23
Tabel 3.2 Spesifikasi kasus penggunaan UC01 23
Tabel 3.3 Spesifikasi kasus penggunaan UC02 25
Tabel 3.4 Model data *triplets*..... 26
Tabel 3.5 Contoh format data *triplets* yang disimpan dalam *triple store* 27
Tabel 4.1 *Tools* yang digunakan pada tugas akhir 33
Tabel 4.2 Contoh data hasil query SPARQL Dbpedia..... 36
Tabel 5.1 Data uji evaluasi *query* spasial..... 68
Tabel 5.2 Data uji evaluasi hasil ekstraksi 74
Tabel 5.3 Hasil evaluasi ekstraksi menggunakan Data Uji 4 76
Tabel 5.4 Hasil evaluasi ekstraksi menggunakan Data Uji 5 78
Tabel 5.5 Hasil evaluasi ekstraksi menggunakan Data Uji 6 79
Tabel 5.6 Hasil evaluasi ekstraksi menggunakan semua Data Uji 79
Tabel 5.7 Data uji coba untuk *input* data 81
Tabel 5.8 Data uji coba untuk *query* label..... 83
Tabel 5.9 Hasil uji coba *query*..... 84

[Halaman ini sengaja dikosongkan]

DAFTAR KODE SUMBER

Kode Sumber 4.1 Query pada DBpedia 34
Kode Sumber 4.2 Implementasi Ekstraksi DBpedia..... 37
Kode Sumber 4.3 Query judul artikel 38
Kode Sumber 4.4 Contoh ekstraksi *infobox* artikel *Iberian War* 40
Kode Sumber 4.5 Keseluruhan Implementasi Ekstraksi *Infobox* 40
Kode Sumber 4.6 Implementasi Ektraksi Deskripsi Artikel 42
Kode Sumber 4.7 Mencari geonames dan DBpedia 43
Kode Sumber 4.8 *Query* SPARQL untuk menampilkan semua pasangan *triplets* 44
Kode Sumber 4.9 Memasukkan data nama artikel ke Apache Jena Fuseki 47
Kode Sumber 4.10 Memasukkan data hasil ekstraksi ke Apache Jena Fuseki 52
Kode Sumber 4.11 Implementasi *insert* data ke Apache Jena Fuseki 58
Kode Sumber 4.12 Implementasi *query* dari Apache Jena Fuseki 61

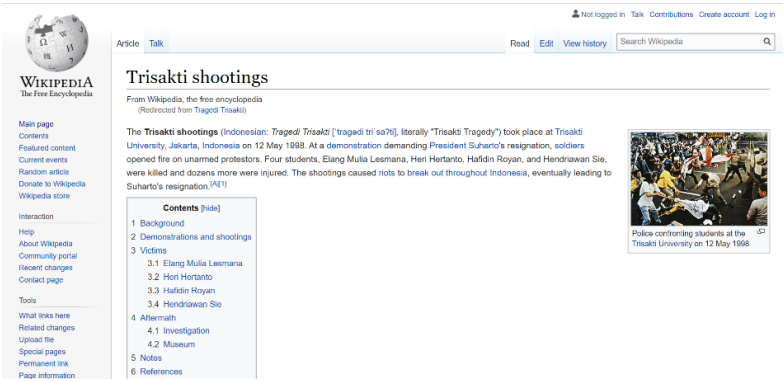
[Halaman ini sengaja dikosongkan]

BAB I
PENDAHULUAN

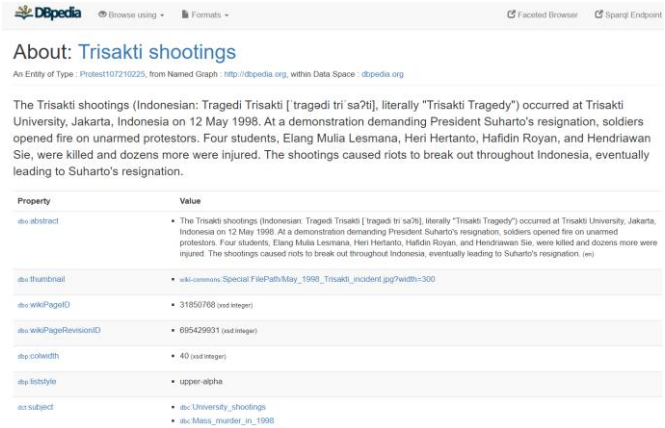
1.1 Latar Belakang

Wikipedia merupakan ensiklopedia yang sangat banyak digunakan. Pada Wikipedia sendiri sudah tersedia jutaan artikel terkait peristiwa atau kejadian. Untuk mempermudah memodelkan informasi menjadi terstruktur dapat memanfaatkan metadata yang telah disediakan oleh DBpedia.org. DBpedia dapat mengekstraksi informasi terstruktur dari Wikipedia terutama dengan menggunakan informasi terstruktur yang ada, seperti *infobox* Wikipedia dan dengan membuat informasi tersebut tersedia di Semantic Web. DBpedia saat ini mengekstraksi data geospasial terbatas dari Wikipedia, hal ini dibatasi pada konten yang disediakan dalam *infobox*.

Kendala saat ini adalah untuk beberapa artikel Wikipedia bertipe *Event*, nama tempat masih belum terdeteksi dan tidak distrukturkan di DBpedia maupun di *infobox*, sehingga entitas geospasial hanya diartikan sebagai teks yang *meaningless*. Penjelasan mengenai hal ini dapat dilihat pada Gambar 1.1 dan Gambar 1.2.



Gambar 1.1 Artikel Kejadian Tragedi Trisakti pada Wikipedia



Gambar 1.2 Metadata Artikel Kejadian Tragedi Trisakti pada DBpedia

Pada artikel *Trisakti Shootings*, tidak terdapat informasi nama tempat pada *infobox* artikel Wikipedia dan juga tidak tersedia *property* dbo:place pada metadata artikel DBpedia. Padahal seharusnya jika dilihat pada teks paragraf pertama sudah disebutkan lokasi kejadian terjadi di Jakarta.

Berangkat dari masalah diatas, maka tugas akhir ini akan dibuat suatu sistem yang dapat memperluas jangkauan ekstraksi entitas geospasial dari artikel Wikipedia sehingga entitas geospasial yang terdapat pada *plain text* dapat dikenali. Hasil yang diharapkan dari tugas akhir ini adalah memaknai atau memberikan konteks pada entitas geospasial yang berhasil disaring oleh sistem untuk mengetahui entitas geospasial yang terdapat pada suatu artikel.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat pada tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara mengekstraksi entitas geospasial pada artikel Wikipedia dengan memanfaatkan struktur metadata?

2. Bagaimana cara mengekstraksi entitas geospasial pada artikel Wikipedia dengan menerapkan NLP (*Natural Language Processing*)?

1.3 Batasan Masalah

Batasan masalah pada tugas akhir ini antara lain:

1. Dataset bersumber dari 5000 artikel bertipe Event yang dimuat Wikipedia.
2. Lokasi bertipe *landmark* tidak termasuk dalam tugas akhir ini.
3. Corpus yang digunakan adalah pustaka Geotext yang mengekstraksi nama negara dan kota.
4. Tidak membuat aplikasi yang dapat melakukan *query* spasial tetapi hanya digunakan untuk pengujian menggunakan SPARQL.
5. *Raw text* yang diproses pada artikel Wikipedia diperoleh dari *raw text* deskripsi utama.
6. Inputan pada sistem berupa *link* artikel DBpedia.
7. Proses ekstraksi dibangun berdasarkan *raw text* dan *semistructure* seperti DBpedia dan *Infobox*. Sehingga penelitian ini tidak sampai ke ranah *semantic*.

1.4 Tujuan

Tujuan dari pengerjaan tugas akhir ini adalah mengekstraksi entitas geospasial dari artikel Wikipedia, sehingga dapat digunakan untuk membangun perangkat lunak yang dapat melakukan *query* informasi spasial dari Wikipedia.

1.5 Manfaat

Manfaat dari hasil pembuatan tugas akhir ini adalah untuk menghasilkan suatu sistem yang dapat mengenali entitas geospasial yang tersedia sehingga mesin lebih mudah mengenali entitas yang disebutkan dan dapat meningkatkan pencarian informasi dan penalaran mesin.

1.6 Metodologi

Langkah-langkah yang harus ditempuh dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Penyusunan Proposal Tugas Akhir

Proposal tugas akhir berisi tentang penjelasan mengenai pendahuluan dari tugas akhir yang dibuat. Pendahuluan ini terdiri atas deskripsi latar belakang, rumusan masalah, batasan masalah, tujuan, dan manfaat dari tugas akhir. Selain itu, dijelaskan pula tinjauan pustaka yang digunakan sebagai referensi dalam pengerjaan tugas akhir. Pada subbab metodologi dijelaskan mengenai urutan pengerjaan tugas akhir dimulai dari penyusunan proposal sampai penyusunan buku tugas akhir.

2. Studi Literatur

Pada studi literatur ini akan dipelajari sejumlah referensi yang dapat mendukung proses pengerjaan aplikasi terutama mengenai *SPARQL, NLP, GeoNames* dan *Apache Jena Fuseki*.

3. Analisis dan desain sistem

Analisis yang dilakukan pada tugas akhir ini adalah dengan memahami dataset yang telah diambil. Kemudian dilakukan analisa pada ekstraksi geografis menggunakan struktur metadata dan menggunakan NLP.

4. Implementasi

Sistem akan dibangung menggunakan bahasa pemrograman Python dengan tools Pycharm dengan berbagai macam library pendukungnya yang terdapat pada Python.

Sedangkan untuk perangkat keras dan sistem operasi yang digunakan selama tugas akhir adalah:

- Processor : Intel® Core™ i5-8250U CPU @ 1.60GHz
- Installed RAM : 8,00 GB (7,88 GB usable)
- System Type : 64-bit operating system, x64-based processor
- Windows Edition : Windows 10 Home

5. Uji Coba dan Evaluasi

Pengujian pada tugas akhir ini dilakukan untuk memeriksa entitas geospasial. Setelah semua kandidat entitas geospasial telah diekstraksi, maka dilakukan disambiguasi dengan referensi geospasial yang lain. Jika referensi entitas memetakan ke satu titik

geospasial, entitas kandidat diterima. Namun, jika pencarian tidak memetakan ke satu titik geospasial, maka tidak dianggap sebagai entitas kandidat.

Kemudian juga dilakukan pengujian untuk mengetahui apakah penggunaan metode ini sudah tepat demi tercapainya tujuan dari tugas akhir ini. Uji coba terlebih dahulu dibuat skenarionya agar pengujiannya tepat sasaran dan sesuai dengan tujuan yang diinginkan.

Evaluasi dalam tugas akhir ini bermanfaat untuk mendapatkan nilai kuantitatif dari scenario uji coba yang dilakukan. Pada tugas akhir ini, perhitungan yang digunakan adalah *precision* dan *recall*. Hasil kedua perhitungan ini diintrepretasikan untuk menjawab apakah tujuan dari tugas akhir ini tercapai.

6. Penyusunan Buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan pada tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
    a. Latar Belakang
    b. Rumusan Masalah
    c. Batasan Tugas Akhir
    d. Tujuan
    e. Manfaat
    f. Metodologi
    g. Sistematika Penulisan
2. Tinjauan Pustaka
3. Analisis dan Perancangan
4. Implementasi
5. Uji Coba dan Evaluasi
6. Kesimpulan dan Saran
7. Daftar Pustaka

1.7 Sistematika Penulisan

Sistematika Penulisan Buku tugas akhir ini bertujuan untuk mendapatkan gambaran dari pengerjaan tugas akhir. Selain itu, diharapkan dapat berguna untuk pembaca yang tertarik untuk melakukan pengembangan lebih lanjut. Secara garis besar, buku tugas akhir terdiri atas beberapa bagian seperti berikut ini:

BAB I. **Pendahuluan**
Bab ini berisi latar belakang masalah, tujuan dan manfaat pembuatan tugas akhir, permasalahan, batasan masalah, metodologi yang digunakan, dan sistematika penyusunan tugas akhir.

BAB II. **Tinjauan Pustaka**
Bab ini menjelaskan beberapa teori yang dijadikan penunjang dan berhubungan dengan pokok pembahasan yang mendasari pembuatan tugas akhir.

BAB III. **Analisis dan Perancangan**
Bab ini membahas mengenai perancangan sistem yang akan dibangun. Perancangan sistem meliputi perancangan data dan alur proses dari sistem itu sendiri.

BAB IV. **Implementasi**
Bab ini berisi implementasi dari perancangan sistem yang telah ditentukan sebelumnya.

BAB V. **Uji Coba dan Evaluasi**
Bab ini membahas pengujian dari metode yang ditawarkan pada tugas akhir untuk mengetahui kesesuaian metode dengan data yang ada.

BAB VI. **Kesimpulan dan Saran**
Kesimpulan bab ini berisi kesimpulan dari hasil pengujian yang telah dilakukan. Bab ini juga

membahas saran-saran untuk pengembangan sistem lebih lanjut.

Daftar Pustaka

Merupakan daftar referensi yang digunakan untuk mengembangkan tugas akhir.

Lampiran

Merupakan bab tambahan yang berisi data atau daftar istilah yang penting pada tugas akhir ini.

[Halaman ini sengaja dikosongkan]

BAB II
TINJAUAN PUSTAKA

Pada bab ini akan dibahas mengenai teori-teori yang menjadi dasar tugas akhir ini.

2.1 Geospasial

Spasial adalah aspek keruangan yang mencakup suatu lokasi, letak dan posisinya. Seiring adanya sistem referensi koordinat peta secara umum, berkembang menjadi geospasial. Geospasial merupakan aspek keruangan yang menjelaskan letak, lokasi, maupun posisi suatu obyek di bumi. Geospasial saat ini telah sering dipakai dalam mengembangkan sistem khususnya sistem informasi geografis.

2.2 Metadata

Metadata adalah data yang menjelaskan tentang suatu data. Dalam pembuatan metadata harus secara terstruktur. Metadata memiliki kode yang digunakan untuk mendeskripsikan suatu data. Metadata bermanfaat untuk memudahkan dalam bertukar data maupun informasi antar sistem yang berbeda. Dalam metadata berisi tanda kode(tag) yang digunakan dalam penyimpanan suatu data [1]. Gambar 2.1 merupakan contoh dari metadata pada *infobox* Wikipedia.

Gambar 2.1 Contoh *infobox* Wikipedia

2.3 DBpedia

DBpedia mengekstrak pengetahuan multibahasa terstruktur dari Wikipedia dan membuatnya tersedia secara bebas di Web menggunakan teknologi *Semantic Web* dan Linked Data. DBpedia mengekstrak pengetahuan dari 111 edisi bahasa Wikipedia yang berbeda. DBpedia memetakan *infobox* Wikipedia dari 27 edisi bahasa yang berbeda menjadi satu ontologi bersama yang terdiri dari 320 kelas dan 1.650 properti. Pemetaan dibuat melalui upaya pencarian kerumunan di seluruh dunia dan memungkinkan pengetahuan dari berbagai edisi Wikipedia untuk digabungkan. DBpedia memiliki basis pengetahuan langsung yang diperbarui setiap kali halaman di Wikipedia berubah. DBpedia menetapkan 27 juta tautan RDF yang menunjuk ke lebih dari 30 sumber data eksternal dan dengan demikian memungkinkan data dari sumber-sumber ini digunakan bersama dengan data DBpedia. Beberapa ratus set data di web mempublikasikan tautan RDF yang menunjuk ke DBpedia sendiri dan dengan demikian menjadikan DBpedia salah satu hub interlinking pusat di cloud Linked Open Data (LOD) [2].

Setiap entitas di DBpedia memiliki satu atau lebih tipe. Tipe yang didefinisikan oleh predikat http://www.w3.org/1999/02/22-rdf-syntax-ns ##type dan biasanya menunjuk ke objek yang mendefinisikan konsep abstrak, seperti benda, orang, atau tempat. Subjek dikatakan kejadian jika dan hanya jika dijelaskan dalam triple yang tipenya adalah "event", diidentifikasi oleh URI http://dbpedia.org/ontology/Event. Penjelasan mengenai hal ini dapat dilihat pada Gambar 2.2.

Gambar 2.2 Contoh Event DBpedia

Pada metadata artikel *Battle of Surabaya* yang bertipe *event*, *dbr:Surabaya* dikenali sebagai isian pada *property dbo:place*. *Dbo:place* ini bertujuan untuk mendefinisikan lokasi kejadian pada artikel bertipe event tersebut.

2.4 *Ontology*

Ontology adalah teori tentang makna suatu objek dengan objek lainnya yang dapat menghasilkan pengetahuan. *Ontology* juga menjelaskan tentang sebuah konsep dalam sebuah domain tertentu (*Classes*), *object properties*, dan *data properties*. Secara umum, *ontology* digunakan pada Artificial Intelligence (AI) dan rekayasa pengetahuan. Metode ontologi dapat diterapkan dalam berbagai bidang agar dapat terhubung dan saling bertukar informasi [3].

2.5 GeoNames

GeoNames adalah salah satu sumber data yang tersedia dalam mesin yang dapat dibaca format atau layanan web. Basis data geografis GeoNames mengandung lebih dari 10 juta nama geografis dan terdiri lebih dari 9 juta fitur unik dari 2,8 juta tempat

berpenduduk dan 5,5 juta nama alternatif. Basis data diperbarui secara berkala dan informasinya bersumber dari puluhan sumber [4].

2.6 *Linked Open Data*

Linked Open Data adalah perpaduan dari *Linked Data* dan *Open Data*, keduanya terhubung dan menggunakan sumber terbuka. Salah satu contoh dari serangkaian LOD adalah DBpedia – upaya crowd-sourced untuk mengekstrak informasi terstruktur dari Wikipedia dan membuatnya tersedia di Web [5].

2.7 *Natural Language Processing*

Natural Language Processing (NLP) adalah bagian dari Kecerdasan Buatan yang memfokuskan pada kemampuan komputer dalam memahami dan memproses bahasa manusia, agar komputer lebih dekat dalam pemahaman bahasa setara *level* manusia. Karena komputer tidak memiliki kemampuan intuisi dalam memahami bahasa natural yang manusia lakukan, komputer tidak dapat memahami apa yang sebenarnya sebuah teks maksud. Dengan demikian, kemajuan dalam *Machine Learning* telah memungkinkan komputer dalam melakukan banyak hal yang bermanfaat dalam *natural language*. Pada tugas akhir ini, GeoText digunakan untuk melakukan pemrosesan text yang dilakukan pada deskripsi utama artikel Wikipedia. GeoText ini berfungsi untuk mengekstraksi nama negara dan nama kota yang terdapat pada teks deskripsi utama artikel Wikipedia. Corpus yang dimiliki oleh GeoText ini adalah Geonames.

2.8 SPARQL

SPARQL merupakan bahasa *query* RDF yang berfungsi untuk mengambil dan memanipulasi data dari sebuah *triple store*. Data RDF dapat dianggap sebagai tabel dengan tiga kolom - kolom subjek, kolom predikat, dan kolom objek, yang dikenal dengan sebutan *triplets*. SPARQL memungkinkan untuk dijalankan pada

semua format RDF, seperti RDF/XMLTurtle, N-triples, dan lain-lain. Pada tugas akhir ini, SPARQL *query* digunakan untuk mendapatkan data awal, mencari data artikel yang tersimpan dalam *triple store* Apache Jena Fuseki dan untuk meng-*input* data artikel pada *triple store* Apache Jena Fuseki.

2.9 Apache Jena Fuseki

Apache Jena Fuseki adalah *graph store server* yang menyediakan protokol SPARQL untuk melakukan *query* dan *update* menggunakan protokol SPARQL Graph Store HTTP. Apache Jena Fuseki dapat dijalankan sebagai *service* pada sistem operasi, Java web application (WAR), atau sebagai *standalone server*. Apache Jena Fuseki dapat digunakan untuk menyediakan mesin protokol untuk *query* sistem dan penyimpanan RDF lainnya [6].

Pada tugas akhir ini Apache Jena Fuseki bertindak sebagai basis data *triple store* yang bisa diakses melalui *request* HTTP. Dataset yang telah diunggah dapat dikueri menggunakan SPARQL. Apache Jena Fuseki memberikan fasilitas *examples queries* untuk memudahkan pengguna dengan memberikan contoh dari kueri. Contoh ditunjukkan pada Gambar 2.3 Kueri dapat dijalankan dengan menekan tombol play yang terdapat pada kotak *query editor*.

```
1  SELECT ?subject ?predicate ?object
2  WHERE {
3    ?subject ?predicate ?object
4  }
5  LIMIT 5
```

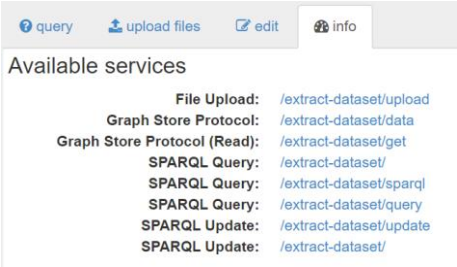
Gambar 2.3 Contoh *query editor*

Hasil kueri dapat ditampilkan dalam bentuk tabel atau raw response seperti ditunjukkan pada Gambar 2.4. Selain itu, hasil kueri dapat ditampilan dalam tipe JSON, XML, CSV, maupun TSV, yang dapat dipilih melalui bagian content type (select).

	subject	predicate	object
1	<http://dbpedia.org/resource/2009_Honduras_earthquake>	<http://www.w3.org/2000/01/rdf-schema#label>	"2009 Honduras earthquake"
2	<http://dbpedia.org/resource/1905_Calabria_earthquake>	<http://www.w3.org/2000/01/rdf-schema#label>	"1905 Calabria earthquake"
3	<http://dbpedia.org/resource/1984_Morgan_Hill_earthquake>	<http://www.w3.org/2000/01/rdf-schema#label>	"1984 Morgan Hill earthquake"
4	<http://dbpedia.org/resource/1987_Santiago_de_Chuco_earthquake>	<http://www.w3.org/2000/01/rdf-schema#label>	"1987 Santiago de Chuco earthquake"
5	<http://dbpedia.org/resource/2000_Turkmenistan_earthquake>	<http://www.w3.org/2000/01/rdf-schema#label>	"2000 Turkmenistan earthquake"

Gambar 2.4 Hasil dari *query*

Apache Jena Fuseki menyediakan beberapa API seperti ditunjukkan pada Gambar 2.5. Beberapa fungsi API tersebut antara lain untuk *endpoint* pengunggahan file RDF (`/upload`), membaca data (`/get`), *query* SPARQL (`/query` atau `/sparql`), dan memperbarui data (`/update`).



Gambar 2.5 Daftar API Apache Jena Fuseki

2.10 Flask

Flask merupakan salah satu kerangka kerja aplikasi web Python yang paling populer. Flask memberikan kebebasan kepada pengembang untuk memilih alat dan pustaka yang ingin digunakan. Serta ada banyak ekstensi yang disediakan sehingga penambahan fungsionalitas baru menjadi lebih mudah [7]. Pada tugas akhir ini Flask digunakan untuk membuat aplikasi web.

2.11 Penelitian Terdahulu

Tabel 2.1 Penelitian terdahulu

Nama Penelitian	Deskripsi Penelitian	Kelebihan	Kekurangan
Mining Wikipedia Article Clusters for Geospatial Entities and Relationships [8]	*Paper* ini mencari entitas geospasial dan hubungan berdasarkan pencarian menggunakan SVM pada text artikel Wikipedia.	*Paper* ini dapat mendeteksi entitas geospasial dan hubungan menggunakan SVM. Sistem ini berhasil mengekstraksi entitas geospasial dengan F-measure 81% dan hubungan lokasi sebesar 54% dengan F-measure.	*Paper* ini hanya memperoleh entitas geospasial berdasarkan teks artikel Wikipedia saja.

Pada Tabel 2.1 menjelaskan tentang penelitian terdahulu beserta kelebihan dan kekurangannya. Kontribusi penelitian Tugas Akhir ini terhadap penelitian terdahulu adalah pencarian entitas geospasial tidak hanya dari teks deskripsi saja namun juga dari struktur metadata seperti DBpedia dan *infobox*. Implementasi penelitian Tugas Akhir ini hanya sampai mengekstraksi nama tempat pada deskripsi jadi sistem tidak dapat membuktikan bahwa nama tempat yang diekstraksi benar merupakan nama tempat pada artikel Wikipedia tersebut. Sehingga penelitian ini tidak sampai ke ranah *semantic* seperti pada penelitian terdahulu, melainkan tugas

akhir ini berdasarkan *raw text* di deskripsi dan *semistructure* pada kerangka DBpedia dan *infobox*.

2.12 *Confusion Matrix*

Confusion Matrix merupakan salah satu metode yang dapat digunakan untuk mengukur kinerja suatu metode klasifikasi. Pada dasarnya, *confusion matrix* mengandung informasi yang membandingkan hasil klasifikasi yang dilakukan oleh sistem dengan hasil klasifikasi yang seharusnya.

Pada pengukuran kinerja menggunakan *confusion matrix*, terdapat 4 (empat) istilah sebagai representasi hasil proses klasifikasi. Keempat istilah tersebut adalah *True Positive* (TP), *True Negative* (TN), *False Positive* (FP), dan *False Negative* (FN).

True Positive (TP) merupakan data positif yang diprediksi benar. Contohnya, kata yang merupakan nama tempat dan dari sistem yang dibuat memprediksi kata tersebut merupakan nama tempat. *True Negative* (TN) merupakan data negatif yang diprediksi benar. Contohnya, kata bukan merupakan nama tempat dan dari sistem yang dibuat memprediksi kata tersebut bukan merupakan nama tempat. *False Positive* (FP) merupakan data negatif namun diprediksi sebagai data positif. Contohnya, kata bukan merupakan nama tempat dan dari sistem yang dibuat memprediksi kata tersebut merupakan nama tempat. *False Negative* (FN) merupakan data positif namun diprediksi sebagai data negatif. Contohnya, kata merupakan nama tempat dan dari sistem yang dibuat memprediksi kata tersebut bukan merupakan nama tempat. Pada tugas akhir ini *True Negative* (TN) tidak digunakan disebabkan semua kata pada *raw text* selain entitas nama tempat benar-benar tidak dilabeli oleh sistem. Jika dilakukan pencarian kata-kata tersebut di artikel Wikipedia akan sangat banyak sehingga tidak memungkinkan melakukan pencarian nilai *True Negative* (TN).

		Actual Values	
		Positive (1)	Negative (0)
Predicted Values	Positive (1)	TP	FP
	Negative (0)	FN	TN

Gambar 2.6 *Confusion Matrix*

Metrik evaluasi *precision*, dan *recall* secara lebih rinci dijelaskan sebagaimana berikut:

a. *Precision*

Precision adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem. Persamaan (1) merupakan perhitungan *precision*.

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

b. *Recall*

Recall adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi [9]. Persamaan (2) merupakan perhitungan *recall*.

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

[Halaman ini sengaja dikosongkan]

BAB III
ANALISIS DAN PERANCANGAN

Pada bab ini akan dijabarkan analisis dan perancangan sistem dari tugas akhir yang meliputi tahap-tahap penyelesaian tugas akhir.

3.1 Analisis Permasalahan

Pokok permasalahan yang akan dibahas dan akan dicarikan solusinya dalam tugas akhir ini adalah mengekstraksi entitas geospasial dari artikel Wikipedia. Wikipedia merupakan ensiklopedia yang sangat banyak digunakan. Pada Wikipedia sendiri sudah tersedia jutaan artikel bertipe *Event*. Untuk mempermudah memodelkan informasi menjadi terstruktur dapat memanfaatkan metadata yang telah disediakan oleh DBpedia dan *infobox*. Namun, sejauh ini struktur metadata yang ada belum lengkap untuk menentukan nama tempat pada artikel Wikipedia bertipe *Event*.

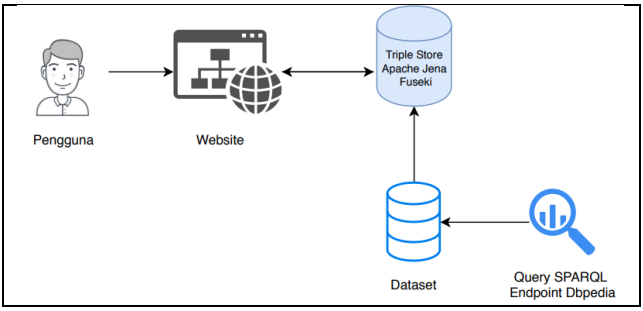
Pengekstraksian entitas geospasial pada artikel Wikipedia dibutuhkan untuk meningkatkan pencarian informasi dan penalaran mesin. Oleh karena itu, untuk meningkatkan hasil ekstraksi entitas geospasial tidak hanya memanfaatkan struktur metadata pada DBpedia dan *infobox* saja, melainkan juga dilakukan pengekstraksian dengan menerapkan NLP (*Natural Language Processing)* terhadap entitas geospasial pada *raw text* deskripsi utama artikel Wikipedia.

Dengan melihat permasalahan di atas, maka pada tugas akhir ini akan dibuat suatu aplikasi yang dapat mengenali entitas geospasial yang tersedia sehingga mesin lebih mudah mengenali entitas yang disebutkan dan dapat meningkatkan pencarian informasi dan penalaran mesin.

3.2 Deskripsi Umum Sistem

Pada tugas akhir ini akan dibangun suatu aplikasi yang dapat memperkaya tag mengenai nama tempat dari suatu *resource* metadata sehingga dapat dikenali sebagai nama tempat.

Arsitektur sistem pada tugas akhir ini dapat dilihat pada Gambar 3.1

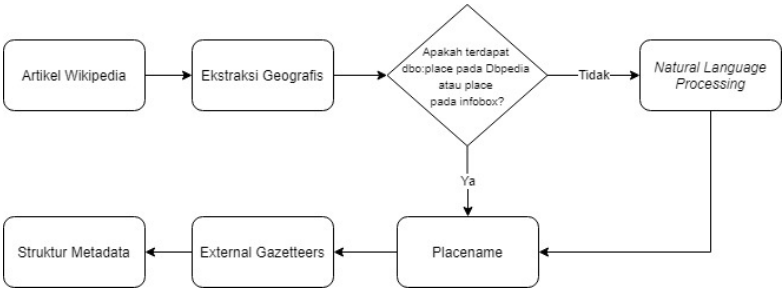


Gambar 3.1 Arsitektur Sistem

a. Pengguna
   Pengguna pada aplikasi ini berperan sebagai yang melakukan penginputan data dan pencarian data artikel.
b. Website
   Website disini berguna sebagai yang melakukan dua fungsi utama yaitu Input Data dan Cari Data. Kondisi sebelum input data adalah link yang akan dimasukkan belum tersedia pada *triple store* Apache Jena Fuseki. Pada Input Data dilakukan ekstraksi pada sistem kemudian hasilnya akan disimpan pada *triple store* Apache Jena Fuseki. Sedangkan pada Cari Data, dilakukan proses pencarian pasangan *triplets* yang telah tersimpan pada *triple store* Apache Jena Fuseki.
c. *Triple Store* Apache Jena Fuseki
   Triple Store Apache Jena Fuseki berfungsi sebagai penyimpanan hasil ekstraksi dan disimpan dengan pasangan *triplets*. Data yang ingin diekstraksi dapat diperoleh dari input pengguna pada *website* dan dataset.
d. Dataset
   Dataset yang digunakan pada penelitian ini diperoleh dari hasil *query* SPARQL Endpoint DBpedia.

e. Query SPARQL Endpoint DBpedia
   Query SPARQL Endpoint DBpedia dilakukan dengan memasukkan *query* yang menyeleksi artikel berupa tipe Event dan mengembalikan nilai *link* artikel DBpedia berupa tipe Event.

Arsitektur sistem perangkat lunak yang dirancang pada tugas akhir ini ditunjukkan oleh Gambar 3.2. *Input* data berupa link artikel DBpedia mengenai *event* yang ingin diolah. Selanjutnya sistem akan masuk ke proses ekstraksi nama tempat. Jika nama tempat sudah terdeteksi pada DBpedia maka *output* dari aplikasi ini adalah hasil *value* dari *property* penyimpan nama tempat seperti dbo:place, dbo:location, dbp:place dan dbp:location. Jika nama tempat tidak terdeteksi pada DBpedia, maka lanjut ke proses ekstraksi pada *infobox* Wikipedia. *Output* dari proses ini adalah *value* yang terdapat pada tag *place infobox* yang dapat diperoleh dengan menggunakan library *wptools*. Jika nama tempat belum terdeteksi juga pada DBpedia dan *infobox* maka akan dilakukan ekstraksi *raw text* artikel Wikipedia menggunakan Natural Language Processing. *Placename* yang diperoleh akan dilakukan pemeriksaan apakah tersedia pada GeoNames. *Placename* yang telah diverifikasi pada eksternal gazetteers akan diarahkan ke GeoNames untuk mendapatkan link id nama tempat pada GeoNames. Selain itu, nama tempat juga dapat diarahkan ke DBpedia untuk mendapatkan link nama tempat tersebut.

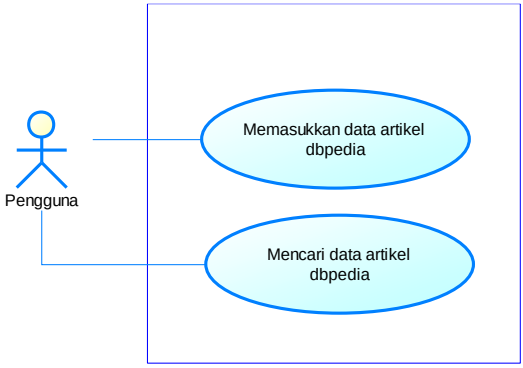


Gambar 3.2 Arsitektur sistem perangkat lunak

3.3 Kasus Penggunaan Sistem

Diagram kasus pengunaan dapat dilihat pada Gambar 3.3. Terdapat satu aktor yang terlibat dan berinteraksi langsung dengan sistem. Aktor yang berinteraksi dengan sistem yaitu pengguna yang diasumsikan tidak memahami bahasa pemrograman.

Selain aktor, pada diagram tersebut juga digambarkan dua kasus penggunaan, yaitu memasukkan link artikel DBpedia dan mencari data artikel. Daftar kasus penggunaan sistem dapat dilihat pada Tabel 3.1.



Gambar 3.3 Diagram kasus penggunaan

Tabel 3.1 Daftar kasus penggunaan

Kode	Nama Kasus Penggunaan
UC01	Memasukkan data artikel DBpedia
UC02	Mencari data artikel DBpedia

3.3.1 Memasukkan Data Artikel DBpedia

Pada kasus penggunaan ini, pengguna dapat memasukkan data artikel berupa link artikel DBpedia yang akan diekstraksi oleh sistem. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.2.

Tabel 3.2 Spesifikasi kasus penggunaan UC01

Kode Kasus Penggunaan	UC01
Nama Kasus Penggunaan	Memasukkan Data Artikel DBpedia
Aktor	Pengguna
Deskripsi	Pengguna dapat memasukkan data artikel berupa link artikel DBpedia pada sistem
Relasi	*Directed Association*
Kondisi Awal	Data artikel belum tersimpan pada *triple store*
Kondisi Akhir	Data artikel tersimpan pada *triple store*
Alur Normal	
Aktor	Sistem
2. Pengguna memilih pilihan “input data”.	1. Sistem menampilkan halaman utama. 3. Sistem menampilkan isian *input* data.

4. Pengguna memasukkan data artikel ke dalam isian *input* data. Data meliputi link artikel DBpedia.	5. Sistem memeriksa data yang dimasukkan oleh pengguna. A1. Data yang dimasukkan pengguna tidak sesuai. 6. Sistem melakukan proses ekstraksi terhadap link artikel sekaligus menyimpan data dan hasil ekstraksi ke dalam *triple store.* 7. Sistem menampilkan pesan input data berhasil.
Alur Alternatif	
A1. Data yang dimasukkan pengguna tidak sesuai	
Aktor	Sistem
A1.2. Pengguna mendapatkan pesan *error*. A1.3. Pengguna kembali ke alur nomor 4.	A1.1. Sistem menampilkan pesan *error* bahwa ada data yang tidak sesuai atau belum diisi
Eksepsi	
-	

3.3.2 Mencari Data Artikel DBpedia

Pada kasus penggunaan ini, pengguna dapat melihat hasil ekstraksi berdasarkan pencarian berdasarkan link artikel DBpedia. Spesifikasi kasus penggunaan dapat dilihat pada Tabel 3.3.

Tabel 3.3 Spesifikasi kasus penggunaan UC02

Kode Kasus Penggunaan	UC02
Nama Kasus Penggunaan	Mencari Data Artikel DBpedia
Aktor	Pengguna
Deskripsi	Pengguna dapat mencari data artikel yang telah diekstraksi
Relasi	*Directed Association*
Kondisi Awal	Data artikel sudah tersimpan pada *triple store*
Kondisi Akhir	Data hasil ekstraksi artikel ditampilkan pada sistem
Alur Normal	
Aktor	Sistem
	1. Sistem menampilkan halaman utama
2. Pengguna memilih pilihan “query data”	
	3. Sistem menampilkan isian *query* data.
4. Pengguna memasukkan link artikel DBpedia.	
	5. Sistem menampilkan hasil ekstraksi berdasarkan link artikel DBpedia.
6. Pengguna melihat hasil ekstraksi.	
Alur Alternatif	
-	

Eksepsi
-

3.4 Analisis dan Perancangan Data

Pada subbab ini akan dijelaskan mengenai perancangan data yang digunakan pada pembuatan tugas akhir. Data yang digunakan merupakan nama artikel bertipe *Event* yang diambil dari DBpedia.

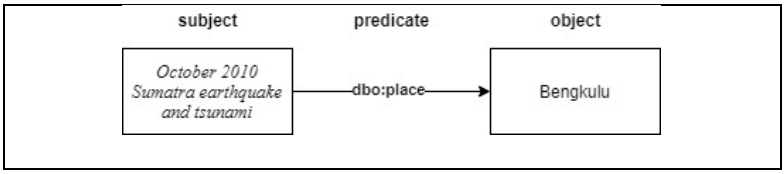
Data artikel disimpan ke dalam *triple store* Apache Jena Fuseki dalam bentuk pasangan *triplets* yang merupakan entitas data dengan susunan *subject-predicate-object*. Model data *triplets* ditunjukkan pada Tabel 3.4. Pasangan triplets pada triple store Apache Jena Fuseki ini terdiri dari link artikel DBpedia bertipe *Event* sebagai *subject*, *property* DBpedia yaitu *dbo:place* sebagai *predicate* dan hasil ekstraksi nama tempat sebagai *object*. Contoh data *triplets* ditunjukkan oleh Gambar 3.4.

Tabel 3.4 Model data *triplets*

Nama Atribut	Jenis Atribut	Keterangan
Subject	String (300)	Link artikel DBpedia
Predicate	String (10)	dbo:place
Object	String (100)	Hasil ekstraksi nama tempat

Pada Gambar 3.4, penelitian ini mengambil salah satu artikel sebagai contoh, yaitu *October 2010 Sumatra earthquake and tsunami*. Pada gambar ditunjukkan bahwa *October 2010 Sumatra earthquake* merupakan *subject* yang memiliki satu *predicate* yang terhubung dengan object. Misalnya pada gambar ditunjukkan bahwa artikel ini memiliki *predicate dbo:place* dengan *object* Bengkulu. Pasangan *triplets* tersebut dapat diartikan secara harfiah bahwa kejadian *October 2010 Sumatra earthquake and tsunami* terletak di Bengkulu. Dari contoh data *triplets* artikel tersebut

apabila disimpan ke *triple store* bentuk formatnya menjadi seperti yang ditunjukkan oleh Tabel 3.5.



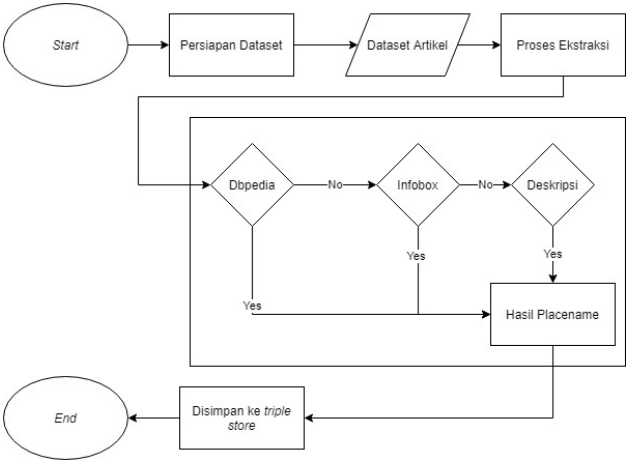
Gambar 3.4 Contoh data *triplets*

Tabel 3.5 Contoh format data *triplets* yang disimpan dalam *triple store*

subject	***predicate***	***object***
http://dbpedia.org/resource/October_2010_Sumatra_earthquake_and_tsunami	http://dbpedia.org/ontology/place	"Bengkulu"

3.5 Perancangan Proses

Perancangan proses merupakan penjelasan proses apa saja yang dilakukan untuk tugas akhir ini.



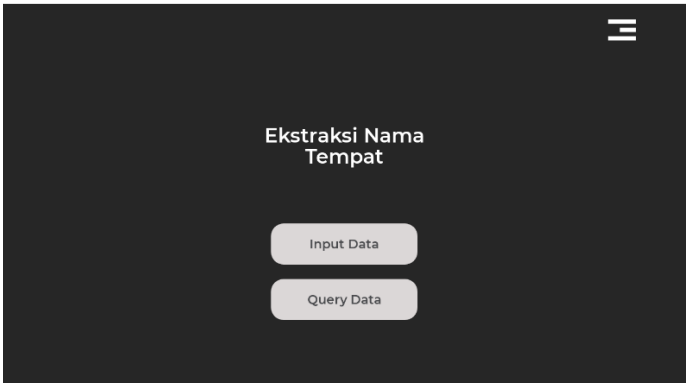
Gambar 3.5 Diagram alir seluruh tahapan

Pada diagram alir seluruh tahapan seperti ditunjukkan oleh Gambar 3.5 digambarkan bahwa tahap pertama dilakukan

persiapan dataset dengan cara *query* dari DBpedia yang difilter berdasarkan kategori Event. Kategori Event terdiri dari beberapa *subevents*. Pada penelitian ini, subevents yang diambil adalah *Natural Events* dan *Societal Events*. Pada *Natural Events* dan *Societal Events* ini sendiri juga memiliki beberapa subevents namun penelitian ini membatasi hanya mengambil subevents bertipe *earthquake* pada *Natural Events* dan *military conflict* pada *Societal Events* disebabkan data pada beberapa *subevents* yang lain tidak mencukupi dan tidak sesuai. Keluarannya berupa link artikel dan judul artikel yang diperoleh dari hasil *query* pada SPARQL DBpedia. Setelah mendapatkan dataset, dilakukan proses ekstraksi. Proses ekstraksi terdiri dari tiga tahap yang dikerjakan secara berurutan. Pada tahap pertama, dilakukan ekstraksi pada DBpedia dengan cara mengambil *value* pada properti yang menyimpan nama tempat, yaitu, dbo:place, dbo:location, dbp:place, dbp:location. Jika tidak ditemukan nama tempat pada tahap tersebut dilanjutkan ke tahap kedua yaitu ekstraksi pada *infobox*. Pada tahap ekstraksi *infobox* dilakukan ekstraksi dengan mengambil *value* pada tag *place*. Kemudian jika nama tempat masih belum ditemukan juga maka akan dilakukan pencarian pada *raw text* deskripsi utama artikel Wikipedia. Jika nama tempat sudah ditemukan, maka ditambahkan ke dalam *triple store* untuk memperkaya korpus data nama tempat yang telah dibuat. Pasangan triplets pada triple store Apache Jena Fuseki ini terdiri dari link artikel DBpedia sebagai *subject*, property DBpedia yaitu *dbo:place* sebagai *predicate* dan hasil ekstraksi nama tempat sebagai *object*.

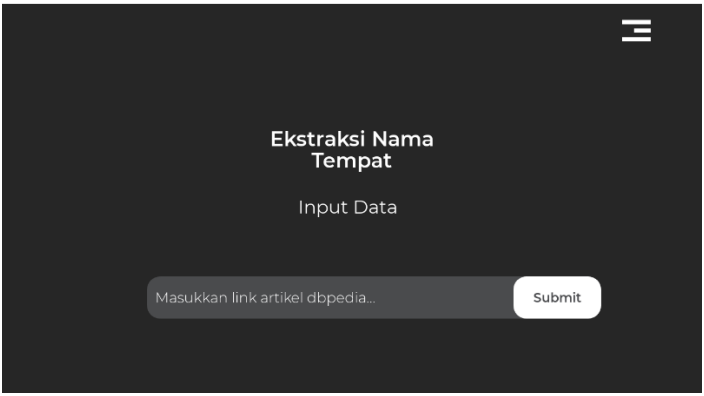
3.6 Perancangan Antarmuka Sistem

Antarmuka sistem pada tugas akhir ini digunakan untuk menampilkan demo simulasi hasil ekstraksi dengan melakukan *input* link artikel DBpedia. Selain itu, antarmuka sistem juga dapat menampilkan hasil *query* sesuai dengan link artikel DBpedia yang dimasukkan oleh pengguna.



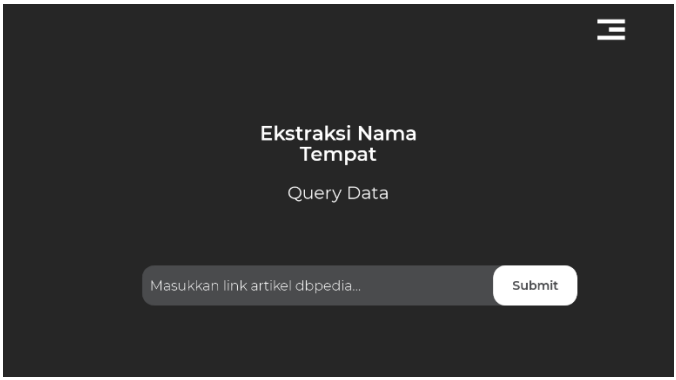
Gambar 3.6 Rancangan antarmuka halaman beranda aplikasi

Rancangan halaman depan atau beranda aplikasi seperti ditunjukkan pada Gambar 3.6, terdapat dua tombol yaitu *input* data dan *query* data. Apabila tombol input data diklik maka akan berpindah halaman input data artikel. Pada halaman yang ditunjukkan pada Gambar 3.7, pengguna dapat memasukkan link artikel DBpedia pada *form* tersebut. Setelah memasukkan link artikel DBpedia, pengguna dapat menekan tombol Submit agar sistem melakukan proses ekstraksi.



Gambar 3.7 Rancangan antarmuka halaman *input* data

Sedangkan apabila pada halaman beranda aplikasi, pengguna memilih tombol *query* data, maka akan berpindah ke halaman *query* data. Pada halaman yang ditunjukkan pada Gambar 3.8, pengguna dapat memasukkan kata kunci pencarian pada *form* berupa link DBpedia yang ingin dicari hasil ekstraksinya. Setelah memasukkan kata kunci yang sesuai, pengguna dapat menekan tombol Submit agar sistem memproses pencarian. Sistem akan menampilkan hasilnya seperti ditunjukkan oleh Gambar 3.9



Gambar 3.8 Rancangan antarmuka halaman *query* data

Gambar 3.9 Rancangan antarmuka halaman hasil ekstraksi

3.7 Evaluasi dan Uji Coba

Pada tahap evaluasi dan uji coba dilakukan evaluasi untuk proses ekstraksi, uji coba *query* spasial, uji coba *input* data, dan uji coba *query* data. Evaluasi proses ekstraksi dilakukan penelitian ini dengan menghitung nilai *precision* dan *recall* untuk setiap data uji. Sedangkan untuk uji coba *query* spasial, *input* data, dan *query* data dilakukan penelitian ini secara manual dengan cara uji coba langsung pada sistem.

[Halaman ini sengaja dikosongkan]

BAB IV
IMPLEMENTASI

Pada bab ini akan dijelaskan implementasi yang dilakukan selama tugas akhir. Bab ini juga akan merinci perangkat serta *tools* yang digunakan pada tugas akhir ini beserta langkah-langkah pengerjaannya.

4.1 Lingkungan Implementasi

Pada tugas akhir ini digunakan beberapa perangkat serta *tools* yang membantu dalam pengerjaan tugas akhir. Perangkat keras dan sistem operasi yang digunakan selama tugas akhir adalah:

- Processor Intel® Core™ i5-8250U CPU @ 1.60GHz
- Installed RAM 8.00 GB (7.88 GB usable)
- System Type 64-bit operating system, x64-based processor
- Windows Edition Windows 10 Home

Sedangkan untuk *tools* yang digunakan selama tugas akhir diuraikan pada Tabel 4.1.

Tabel 4.1 *Tools* yang digunakan pada tugas akhir

No.	*Tools*	Keterangan
1	Python	Bahasa Python digunakan untuk menangani ekstraksi data.
2	Pycharm, Sublime Text 3, Visual Studio Code	Aplikasi yang digunakan untuk penulisan program.
3	Ms. Excel dan Ms. Word	Aplikasi untuk mengolah data.
4	Wptools	Pustaka untuk memperoleh ekstraksi halaman, gambar, *infobox*, Wikidata dan lain-lain.
5	Geotext	Pustaka untuk mengekstraksi nama negara dan kota pada text.

No.	*Tools*	Keterangan
6	Flask	Kerangka kerja aplikasi web Python.
7	Apache Jena Fuseki	Triple store untuk menyimpan data.

4.2 Persiapan dan Pengambilan Data

Data artikel dari Wikipedia yang memuat artikel bertipe kejadian diambil dengan melakukan *query* pada *knowledge base* DBpedia. Proses tersebut dapat dilakukan dengan menggunakan Kode Sumber 4.1.

```
1.  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
    schema#>
2.  PREFIX  rdf: <http://www.w3.org/1999/02/22-rdf-
    syntax-ns#>
3.  PREFIX  dbo: <http://dbpedia.org/ontology/>
4.  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
5.
6.  SELECT ?evURI ?title ?Wikipedia_page
7.  WHERE {
8.    ?sub rdfs:subClassOf* dbo:Event.
9.    {
10.     ?sub rdfs:label "earthquake"@en .
11.   }
12.   UNION {
13.     ?sub rdfs:label "military conflict"@en .
14.   }
15.   ?evURI a ?sub
16.   OPTIONAL {
17.     ?evURI rdfs:label ? title.
18.     FILTER (lang(?title) = "en")
19.   }
20.   OPTIONAL {
21.     ?evURI foaf:name ? title.}
22.   OPTIONAL {
23.    ?evURI ^foaf:primaryTopic ?Wikipedia_page
24.   }}LIMIT 5000
```

Kode Sumber 4.1 Query pada DBpedia

4.3 Implementasi Proses

Implementasi proses dilakukan berdasarkan perancangan proses yang dijelaskan pada bab Analisis dan Perancangan Sistem.

4.3.1 Implementasi Ekstraksi DBpedia

Implementasi ekstraksi DBpedia dilakukan dengan menggunakan *query* pada SPARQL DBpedia. Data yang diperoleh dari hasil query disimpan dalam format csv. Data tersebut diolah sehingga menyimpan tiga data diantaranya *url*, *title* dan *Wikipedia_Page*. Contoh data dapat dilihat pada Tabel 4.2. Pada Kode Sumber 4.2, terlebih dahulu program membaca file csv. Data yang dibutuhkan dalam proses ekstraksi ini hanya data *url* sehingga kolom *url* diubah ke dalam bentuk list. Dengan memanfaatkan SPARQLWrapper yang berfungsi membantu dalam membuat *query uri* dan mengubah hasilnya menjadi format yang lebih mudah dikelola. Pada link artikel DBpedia memiliki beberapa properti untuk menyimpan lokasi kejadian, pada penelitian ini membatasi properti untuk menyimpan lokasi dalam dbo:place, dbp:place, dbo:location, dan dbp:location. Objek dari *query* tersebut berupa link DBpedia lokasi kejadian tersebut. Hasil *query* kemudian disimpan dalam format JSON. Jika dalam satu artikel memiliki beberapa lokasi maka lakukan perulangan terhadap hasil *query* dan simpan pada suatu array. Contoh hasil ekstraksi DBpedia dapat dilihat pada Gambar 4.1.

Tabel 4.2 Contoh data hasil query SPARQL Dbpedia

No.	url,title,Wikipedia_Page
1	http://dbpedia.org/resource/2009_Honduras_earthquake,2009 Honduras earthquake,http://en.wikipedia.org/wiki/2009_Honduras_earthquake
2	http://dbpedia.org/resource/1905_Calabria_earthquake,1905 Calabria earthquake,http://en.wikipedia.org/wiki/1905_Calabria_earthquake
3	http://dbpedia.org/resource/1984_Morgan_Hill_earthquake,1984 Morgan Hill earthquake,http://en.wikipedia.org/wiki/1984_Morgan_Hill_earthquake

```
1.  import pandas as pd
2.  from SPARQLWrapper import SPARQLWrapper, POST, JSON
3.  # membaca file csv
4.  df = pd.read_csv("predata/sparql-all.csv")
5.
6.  # mengubah dataframe ke dalam list
7.  lk = df['url'].tolist()
8.  place_db = []
9.  for x in range(len(nm)):
10.     sparql = SPARQLWrapper("http://dbpedia.org/sparql")
11.     sparql.setQuery("""
12.       PREFIX dbo: <http://dbpedia.org/ontology/>
13.       PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
14.       PREFIX dbr: <http://dbpedia.org/resource/>
15.       SELECT ?place
16.       WHERE {
17.           { <"""+lk[x]+"""> dbo:place ?place. }
18.           UNION
19.           { <"""+lk[x]+"""> dbp:place ?place. }
20.           UNION
21.           { <"""+lk[x]+"""> dbo:location ?place.}
22.           UNION
23.           { <"""+lk[x]+"""> dbp:location ?place.}
24.        }""")
```

```
25.
26.     sparql.setReturnFormat(JSON)
27.     results = sparql.query().convert()
28.     result = []
29.     for result in results["results"]["bindings"]:

30.         place_db.append([(lk[x], result["place"][
    "value"])])
```

Kode Sumber 4.2 Implementasi Ekstraksi DBpedia

```
[[('http://dbpedia.org/resource/Iberian_War',
'http://dbpedia.org/resource/Upper_Mesopotamia')],
[('http://dbpedia.org/resource/Iberian_War',
'http://dbpedia.org/resource/Transcaucasus')],
[('http://dbpedia.org/resource/Iberian_War',
'http://dbpedia.org/resource/Caucasian_Iberia')]]
```

Gambar 4.1 Hasil Ekstraksi DBpedia pada Artikel *Iberian War*

4.3.2 Implementasi Ekstraksi *Infobox*

Implementasi ekstraksi pada *infobox* dilakukan dengan menggunakan bantuan pustaka *wptools*. Data yang dibutuhkan dalam proses ekstraksi ini adalah judul artikel yang diperoleh dari hasil *query*. Proses query dapat dilihat pada Kode Sumber 4.3. Cara implementasi ekstraksi *infobox* dapat dilihat pada Kode Sumber 4.4. Pada kode sumber tersebut, terlebih dahulu masukkan judul artikel yang ingin diekstraksi. Selanjutnya untuk mendapatkan data API:Parse lakukan *command* get_parse() untuk memparsing konten halaman dan mendapatkan output. Pada hasil parsing ini bisa dilihat artikel yang memiliki *infobox*. Data pada *infobox* dapat diambil dengan melakukan perintah `page.data['infobox']`. *Infobox* memiliki beberapa tag diantaranya *place*. Untuk mendapatkan hasil tag pada place maka dapat lakukan perintah `page.data['infobox']['place']`. Data pada tag place itu kemudian dijadikan *nameplace* oleh sistem. Keseluruhan proses ekstraksi dapat dilihat pada Kode Sumber 4.5.

```
1.  from SPARQLWrapper import SPARQLWrapper, POST, JS
    ON
2.  sparql = SPARQLWrapper("http://dbpedia.org/sparql
    ")
3.  sparql.setQuery("""
4.      PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
    schema#>
5.      PREFIX foaf: <http://xmlns.com/foaf/0.1/>
6.      SELECT ?title
7.      WHERE {
8.      <""" + lk[x] + """> rdfs:label ?title.
9.      FILTER(lang(?title) = "en")
10.     OPTIONAL
11.     { <""" + lk[x] + """> foaf:name ?title. }
12.     }
13. """)
14. sparql.setReturnFormat(JSON)
15. results = sparql.query().convert()
16. title = []
17. for result in results["results"]["bindings"]:
18.     title = result['title']['value']
```

Kode Sumber 4.3 Query judul artikel

```
>>> page = wptools.page(title)
>>> page.get_parse()
en.wikipedia.org (parse) Iberian War
en.wikipedia.org (imageinfo) File:Roman-Persian
Frontier in Late ...
Iberian War (en) data
{
  image: <list(1)> {'kind': 'parse-image',
'file': 'File:Roman-Per...
  infobox: <dict(13)> conflict, partof, image,
image_size, caption...
  pageid: 5196520
```

```
  parsetree: <str(16289)>
<root><template><title>About</title><par...
  requests: <list(2)> parse, imageinfo
  title: Iberian War
  wikibase: Q2326066
  wikidata_url:
https://www.wikidata.org/wiki/Q2326066
  wikitext: <str(11451)> {{About|2=the conflict
between Napoleon's...
}
Out[28]: <wptools.page.WPToolsPage at
0x23a66af9208>
>>> page.data['infobox']
{'conflict': 'Iberian War',
 'partof': 'the [[Roman-Persian Wars]]',
 'image': 'Roman-Persian Frontier in Late
Antiquity.svg',
 'image_size': '280px',
 'caption': 'The Roman-Persian frontier in the
4th to 7th centuries',
 'date': '526–532',
 'place': '[[Caucasian Iberia|Iberia]],
[[Transcaucasus]], [[Upper Mesopotamia]]',
 'territory': '[[Sasanian Empire|Sasanians]]
retained Iberia<br>[[Byzantine
Empire|Byzantines]] retained [[Lazica]]',
 'result': 'Persian victory. Treaty of [[Eternal
Peace (532)|Eternal Peace]]<br>[[Byzantine
Empire|Byzantines]] paid tribute of 11,000 lbs
(5000 kg) gold'}
>>> page.data['infobox']['place']
```

```
'[[Caucasian Iberia|Iberia]], [[Transcaucasus]],
[[Upper Mesopotamia]]'
```

Kode Sumber 4.4 Contoh ekstraksi *infobox* artikel *Iberian War*

```
1.  from SPARQLWrapper import SPARQLWrapper, POST, JS
    ON
2.  sparql = SPARQLWrapper("http://dbpedia.org/sparql
    ")
3.  sparql.setQuery("""
4.      PREFIX rdfs: <http://www.w3.org/2000/01/rdf-
    schema#>
5.      PREFIX foaf: <http://xmlns.com/foaf/0.1/>
6.      SELECT ?title
7.      WHERE {
8.      <""" + lk[x] + """> rdfs:label ?title.
9.      FILTER(lang(?title) = "en")
10.     OPTIONAL
11.     { <""" + lk[x] + """> foaf:name ?title. }
12.     }
13. """)
14. sparql.setReturnFormat(JSON)
15. results = sparql.query().convert()
16. title = []
17. for result in results["results"]["bindings"]:
18.     title = result['title']['value']
19. page = wptools.page(title)
20. page.get_parse()
21. # mengambil value pada tag place di infobox
22. text_if = page.data['infobox']['place']
23. # menghapus tanda kurung siku
24. res = re.findall(r'\[\[.*?\]\]', text_if)
25. a = str(res).replace("'[[", "").replace("']]
    ", "")
26. b = str(a).replace("[", "").replace("]", "")
27. place = b.split(", ")
28. for i in place:
29.     # menyimpan nama tempat pada place_db
30.     place_db.append([(lk[x], i)])
```

Kode Sumber 4.5 Keseluruhan Implementasi Ekstraksi *Infobox*

4.3.3 Implementasi Ekstraksi Deskripsi Artikel

Implementasi ekstraksi deskripsi pada artikel kejadian Wikipedia dilakukan dengan menggunakan bantuan pustaka GeoText. Implementasi ekstraksi deskripsi artikel Wikipedia pada sistem dapat dilihat pada Kode Sumber 4.6. Pada kode sumber tersebut, data yang dibutuhkan dalam proses ekstraksi ini adalah judul artikel yang diperoleh dari hasil *query*. Proses query dapat dilihat pada Kode Sumber 4.3. Pada Kode Sumber 4.6, terlebih dahulu masukkan judul artikel yang ingin diekstraksi. Lakukan get_query() pada *wptools* untuk mengambil informasi tentang wiki dan data yang tersimpan di dalamnya, seperti wikitext, tautan dan kategori atau token. `data['extext']` menyimpan data deskripsi utama Wikipedia yang kemudian hasilnya disimpan untuk mencari entitas nama tempat. Pencarian entitas nama tempat dibantu dengan pustaka GeoText untuk mendapatkan nama negara dan kota. Sesuai dengan batasan masalah penelitian ini, nama tempat berupa *landmark* tidak termasuk dalam entitas nama tempat. Sehingga nama negara dan nama kota dapat dijadikan sebagai entitas nama tempat. Contoh *output* hasil ekstraksi deskripsi ditunjukkan oleh Gambar 4.2.

```
1.  import wptools
2.  from geotext import GeoText
3.
4.  page = wptools.page(title)
5.  page.get_query()
6.  # mencari deskripsi artikel Wikipedia
7.  text = page.data['extext']
8.  # mencari entitas negara dan kota pada deskripsi
9.  places_cand = GeoText(text)
10. # menyimpan data negara pada deskripsi
11. a = places_cand.countries
12. # menyimpan data kota pada deskripsi
13. b = places_cand.cities
14. # menggabungkan data negara dan kota dalam place,
     dan menggunakan set agar tidak ada data sama yan
    g berulang
```

```
place = set(a+b)
for i in place:
# menyimpan nama tempat pada place_db
  place_db.append([(url_title, i)])
```

Kode Sumber 4.6 Implementasi Ektraksi Deskripsi Artikel

```
[[('http://dbpedia.org/resource/2000_Turkmenistan_earthquake', 'Ashgabat')],
[('http://dbpedia.org/resource/2000_Turkmenistan_earthquake', 'Balkanabat')],
[('http://dbpedia.org/resource/2000_Turkmenistan_earthquake', 'Turkmenistan')]]
```

Gambar 4.2 Hasil Ekstraksi Deskripsi Artikel *2000 Turkmenistan earthquake*

4.3.4 Implementasi Pencarian Nama Tempat pada Geonames dan DBpedia

Untuk mencari Geonames dapat dilihat pada Kode Sumber 4.7. Dengan menggunakan Geonames API, sistem dapat melakukan pencarian berdasarkan nama tempat yang telah disimpan. Pertama lakukan *request* terhadap halaman web, halaman tersebut disimpan dalam variabel *url*. Metode get () bertujuan mengirimkan permintaan GET ke url yang ditentukan. Params merupakan daftar *tuple* untuk dikirim sebagai string *query*. Pada daftar *tuple* ini terdapat *username* yang dapat diperoleh setelah melakukan *sign in* pada geonames.org. *name_equals* berupa nama tempat yang ingin dicari id geonamesnya. *featureClass* dibatasi dengan karakter P dan A di mana P merupakan *city, village,...* dan A merupakan *country, state, region,....* Untuk mengambil hasil yang pertama muncul dari pencarian dapat menggunakan maxRows = 1. Karena data disimpan dalam string JSON, maka data dapat diparsing menggunakan metode json.loads(). Untuk mendapatkan id geonames dapat memanggil array ‘geonameId’. Contoh output pencarian nama tempat pada ID Geonames ditunjukkan oleh

Gambar 4.3. Sedangkan untuk mencari DBpedia dengan cara mengubah spasi menjadi *underscore* agar mendapat link DBpedia nama tempat tersebut. Contoh output pencarian nama tempat pada DBpedia ditunjukkan oleh Gambar 4.4.

```
1.  from geotext import GeoText
2.  import requests
3.  import json
4.
5.  # mencari geonames
6.  url = 'http://api.geonames.org/searchJSON?'
7.  for m in place:
8.      params = {'username': "pii.rizvi",
9.                'name_equals': m,
10.               'featureClass': ['P', 'A'],
11.               'maxRows': "1"}
12.
13.     e = requests.get(url, params=params)
14.     if e:
15.         pretty_json = json.loads(e.text)
16.     for item in pretty_json["geonames"]:
17.         place_db.append([(title, "https://www.geo
    names.org/{}".format(item['geonameId']))])
18. # mencari DBpedia
19. for n in place:
20.     n2 = n.replace(" ", "_")
21.     place_db.append([(title, "http://dbpedia.org/
    resource/{}".format(n2))])
22.
```

Kode Sumber 4.7 Mencari geonames dan DBpedia

```
[[('http://dbpedia.org/resource/2000_Turkmenistan_earthquake
', 'https://www.geonames.org/162183')],
[('http://dbpedia.org/resource/2000_Turkmenistan_earthquake'
, 'https://www.geonames.org/161616')],
[('http://dbpedia.org/resource/2000_Turkmenistan_earthquake'
, 'https://www.geonames.org/1218197')]]
```

Gambar 4.3 Hasil Ekstraksi *2000 Turkmenistan earthquake* dengan ID Geonames

```
[[('http://dbpedia.org/resource/2000_Turkmenistan_earthquake
', 'http://dbpedia.org/resource/Ashgabat')],
[('http://dbpedia.org/resource/2000_Turkmenistan_earthquake'
, 'http://dbpedia.org/resource/Balkanabat')],
[('http://dbpedia.org/resource/2000_Turkmenistan_earthquake'
, 'http://dbpedia.org/resource/Turkmenistan')]]
```

Gambar 4.4 Hasil Ekstraksi *2000 Turkmenistan earthquake* dengan DBpedia

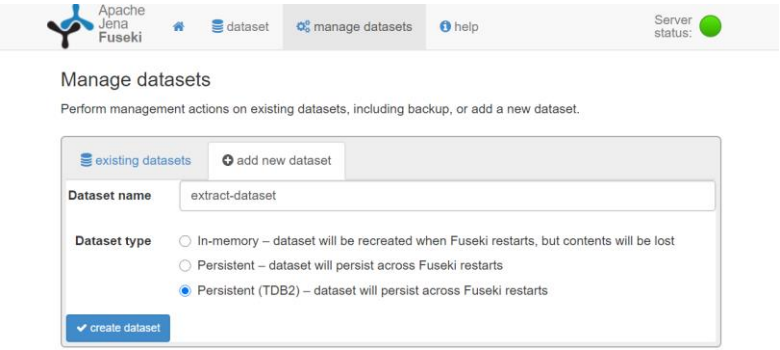
4.3.5 Implementasi Pembuatan *Dataset* pada Apache Jena Fuseki

4.3.5.1 Pembuatan *Dataset* Baru pada Apache Jena Fuseki

Cara membuat *dataset* baru pada Apache Jena Fuseki dapat dilihat pada Gambar 4.5. Kode Sumber 4.8 menunjukkan cara untuk menjalankan sintaks *query* SPARQL pada *tab query* Apache Jena Fuseki. Sintaks *query* SPARQL tersebut bertujuan untuk menampilkan semua pasangan *triplets* (*subject-predicate-object*) yang ada di dalam *dataset*.

```
PREFIX dbo: <http://dbpedia.org/ontology/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX dbr: <http://dbpedia.org/resource/>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT DISTINCT *
WHERE { ?subject ?property ?object }
```

Kode Sumber 4.8 *Query* SPARQL untuk menampilkan semua pasangan *triplets*



Gambar 4.5 Pembuatan *dataset* baru pada Apache Jena Fuseki

4.3.5.2 Penyimpanan Data pada Apache Jena Fuseki

Untuk menambah korpus data artikel pada *dataset* Apache Jena Fuseki, penelitian ini menggunakan *script* Python untuk menambah data secara otomatis. Data artikel yang akan dimasukkan ke Apache Jena Fuseki berupa link artikel DBpedia dan judul artikel sebanyak 5000 data artikel yang disimpan dalam format csv. Cara memasukkan data link artikel dan judul ke Apache Jena Fuseki dapat dilihat pada Kode Sumber 4.9. Sedangkan, cara memasukkan data hasil ekstraksi ke Apache Jena Fuseki dapat dilihat pada Kode Sumber 4.10. Data hasil ekstraksi dikonversi dulu ke dalam format sintaks SPARQL. Sintaks SPARQL tersebut kemudian digunakan sebagai *query insert* data pada Apache Jena Fuseki. Total data yang berhasil tersimpan pada Apache Jena Fuseki sebanyak 32.272 pasang *triplets*. Contoh pasangan *triplets* hasil ekstraksi yang tersimpan dalam Apache Jena Fuseki ditunjukkan oleh Gambar 4.6.

Pada Kode Sumber 4.9 dilakukan pembacaan data terhadap semua data artikel yang telah diambil dari *query* SPARQL DBpedia dan disimpan ke dalam *file* berformat CSV. *File* tersebut memiliki 3 kolom, yaitu: *url*, *title*, dan *Wikipedia_page*. Kolom *url* digunakan untuk menyimpan *link* artikel DBpedia. Sedangkan

kolom *title*, untuk menyimpan judul artikel. Sementara untuk kolom *Wikipedia_page* digunakan untuk menyimpan link artikel Wikipedia. Data yang dibutuhkan dalam proses ini hanya *url* dan *title* saja sehingga kedua kolom tersebut akan diubah menjadi bentuk *list* terlebih dahulu untuk memudahkan proses *insert* data. *List* ini kemudian dikonversi menjadi sintaks *query insert* data. *Query* ini digunakan untuk memasukkan nama artikel pada *triple store* Apache Jena Fuseki. Sintaks *query* menggunakan SPARQL tersebut selanjutnya dikirim ke *endpoint* Apache Jena Fuseki secara berulang sehingga semua data akhirnya dapat tersimpan ke *triple store* Apache Jena Fuseki.

Sementara pada Kode Sumber 4.10, awalnya dilakukan pembacaan data artikel yang tersimpan dalam *file* berformat CSV. Pada proses ini hanya membutuhkan kolom *url* saja sehingga kolom tersebut diubah menjadi bentuk *list* terlebih dahulu untuk memudahkan proses *insert* data. *List url* kemudian masuk ke proses pertama dalam ekstraksi nama tempat yaitu menggunakan *query* SPARQL DBpedia. Pencarian nama tempat pada DBpedia dapat diperoleh dari value dari propert dbo:place, dbo:location, dbp:place, dan dbp:location. Dalam proses *query* SPARQL ini juga dapat digunakan untuk mencari id geonames dan link DBpedia nama tempat yang telah ditemukan. Jika nama tempat belum ditemukan maka lanjut ke proses kedua yaitu ekstraksi *infobox* melalui tag *place*. Untuk mengambil *value* pada tag *place* dapat menggunakan pustaka *wptools*. Jika nama tempat belum ditemukan juga maka lanjut ke proses ketiga yaitu proses ekstraksi melalui deskripsi utama Wikipedia. Proses pengambilan value deskripsi utama juga dapat menggunakan pustaka *wptools*. Untuk proses mencari entitas nama tempat pada deskripsi utama membutuhkan pustaka GeoText yang dapat mengekstraksi nama negara dan kota pada teks. Nama negara dan kota ini kemudian digabungkan dan dijadikan nama tempat. Semua nama tempat yang diperoleh dari proses pertama sampai ketiga disimpan dalam satu

variabel bernama place_db dan dijadikan sebagai hasil ekstraksi. Hasil ekstraksi selanjutnya dikonversi menjadi sintaks *query insert* data. *Query* ini digunakan untuk memasukkan entitas pada *triple store* Apache Jena Fuseki. Sintaks *query* menggunakan SPARQL tersebut selanjutnya dikirim ke *endpoint* Apache Jena Fuseki secara berulang sehingga semua hasil ekstraksi akhirnya dapat tersimpan ke *triple store* Apache Jena Fuseki.

```
1.  import pandas as pd
2.  from SPARQLWrapper import SPARQLWrapper, POST, JS
    ON
3.
4.  # membaca file csv
5.  df = pd.read_csv("predata/sparql-all.csv")
6.
7.  # mengubah dataframe menjadi list
8.  tl = df['title'].tolist()
9.  lk = df['url'].tolist()
10.
11. # insert data link artikel dan nama artikel ke je
    na fuseki
12. for x in range(len(lk)):
13.     sparql = SPARQLWrapper("http://localhost:3030
    /extract-dataset/update")
14.     sparql.setMethod(POST)
15.     sparql.setQuery("""
16.        PREFIX rdfs: <http://www.w3.org/2000/01/rd
    f-schema#>
17.        INSERT DATA { <"""+lk[x]+ """> rdfs:label
    '""" + tl[x]+ """'};
18.     """)
19.     sparql.setReturnFormat(JSON)
20.     results = sparql.query().convert()
21.     print(results)
```

Kode Sumber 4.9 Memasukkan data nama artikel ke Apache Jena Fuseki

```
1.  import pandas as pd
2.  import wptools
```

```
3.  from SPARQLWrapper import SPARQLWrapper, POST, JS
    ON
4.  from geotext import GeoText
5.  import requests
6.  import json
7.  import re
8.
9.  # membaca file csv
10. df = pd.read_csv("predata/sparql-all.csv")
11. # mengubah dataframe ke dalam list
12. lk = df['url'].tolist()
13.
14. def call_geo_db(title, place):
15.     # mencari geonames
16.     url = 'http://api.geonames.org/searchJSON?'
17.     for m in place:
18.         params = {'username': "pii.rizvi",
19.                   'name_equals': m,
20.                   'featureClass': ['P', 'A'],
21.                   'maxRows': "1"}
22.
23.         e = requests.get(url, params=params)
24.         if e:
25.             pretty_json = json.loads(e.text)
26.         for item in pretty_json["geonames"]:
27.             place_db.append([(title, "https://www
    .geonames.org/{}".format(item['geonameId']))])
28.     # mencari DBpedia
29.     for n in place:
30.         n2 = n.replace(" ", "_")
31.         place_db.append([(title, "http://dbpedia.
    org/resource/{}".format(n2))])
32.
33. def call_desc(url_title, title):
34.     page = wptools.page(title)
35.     try:
36.         page.get_query()
37.         # mencari deskripsi artikel Wikipedia
38.         text = page.data['extext']
39.         # mencari entitas negara dan kota pada de
    skripsi
40.         places_cand = GeoText(text)
```

```
41.         # menyimpan data negara pada deskripsi
42.         a = places_cand.countries
43.         # menyimpan data kota pada deskripsi
44.         b = places_cand.cities
45.         # menggabungkan data negara dan kota dala
    m place, dan menggunakan set agar tidak ada data
    yang berulang
46.         place = set(a+b)
47.         for i in place:
48.         # menyimpan nama tempat pada place_db
49.             place_db.append([(url_title, i)])
50.         call_geo_db(url_title, place)
51.     except:
52.         pass
53.
54. place_db = []
55. for x in range(len(lk)):
56.     sparql = SPARQLWrapper("http://dbpedia.org/sp
    arql")
57.     sparql.setQuery("""
58.         PREFIX dbo: <http://dbpedia.org/ontology/
    >
59.         PREFIX rdf: <http://www.w3.org/1999/02/22
    -rdf-syntax-ns#>
60.         PREFIX dbr: <http://dbpedia.org/resource/
    >
61.         PREFIX rdfs: <http://www.w3.org/2000/01/r
    df-schema#>
62.         PREFIX foaf: <http://xmlns.com/foaf/0.1/>

63.         SELECT ?place ?name ?geonames
64.         WHERE {
65.             { <""" + lk[x] + """> dbo:place ?plac
    e. }
66.             UNION
67.             { <""" + lk[x] + """> dbp:place ?plac
    e. }
68.             UNION
69.             { <""" + lk[x] + """> dbo:location ?p
    lace. }
70.             UNION
```

```
71.             { <""" + lk[x] + """> dbp:location ?p
    lace. }
72.             ?place rdfs:label ?name.
73.             FILTER(lang(?name) = "en")
74.             OPTIONAL{
75.              ?place (owl:sameAs|^owl:sameAs) ?geo
    names.
76.              FILTER(regex(str(?geonames), "sws.ge
    onames.org" ) )
77.             }
78.         }
79.     """)
80.
81.     sparql.setReturnFormat(JSON)
82.     results = sparql.query().convert()
83.     result = []
84.
85.     for result in results["results"]["bindings"]:

86.     # menyimpan nama tempat pada place_db
87.         place_db.append([(lk[x], result["place"][
    "value"])])
88.         place_db.append([(lk[x], result["name"]["
    value"])])
89.         if 'geonames' in result:
90.             place_db.append([(lk[x], result["geon
    ames"]["value"])])
91.
92.     if result==[]:
93.         sparql = SPARQLWrapper("http://dbpedia.or
    g/sparql")
94.         sparql.setQuery("""
95.             PREFIX rdfs: <http://www.w3.org/2000/
    01/rdf-schema#>
96.             PREFIX foaf: <http://xmlns.com/foaf/0
    .1/>
97.             SELECT ?title
98.             WHERE {
99.             <""" + lk[x] + """> rdfs:label ?title
    .
100.                 FILTER(lang(?title) = "en")
101.                 OPTIONAL
```

```
102.                    { <""" + lk[x] + """> foaf:name ?title. }
103.                    }
104.                """)
105.                sparql.setReturnFormat(JSON)
106.                results = sparql.query().convert()

107.                title = []
108.                for result in results["results"]["bindings"]:
109.                    title = result['title']['value']
110.                page = wptools.page(title)
111.                try:
112.                    page.get_parse()
113.                    # mengambil value pada tag place di infobox
114.                    text_if = page.data['infobox']['place']
115.                    # menghapus tanda kurung siku

116.                    res = re.findall(r'\[\[.*?\]\]', text_if)
117.                    a = str(res).replace("'[[", "").replace("']]", "")
118.                    b = str(a).replace("[", "").replace("]", "")
119.                    inf = b.split(", ")
120.
121.                    for i in inf:
122.                        # menyimpan nama tempat pada place_db
123.                        place_db.append([(lk[x], i)])
124.                    call_geo_db(lk[x], inf)
125.                except:
126.                    call_desc(lk[x], title)
127.
128.        for x in range(len(place_db)):
129.            ent_ttl = []
130.            ent_plc = []
131.            for i in range(len(place_db[x])):
```

```
132.                if "http" in place_db[x][i][1]:
133.                    ent_plc.append("<"+place_db[x]
    [i][1]+">")
134.                ent_ttl.append(place_db[x][i][0])

135.                ent_plc.append('"'"'+place_db[x][
    i][1]+'"')
136.                insert = ""
137.                for i in range(len(ent_ttl)):
138.                    text = ("INSERT DATA { <"+ent_
    ttl[i]+"> dbo:place "+ent_plc[i]+" };")
139.                    insert += text
140.
141.        # insert data hasil ekstraksi ke jena fuse
    ki
142.                sparql = SPARQLWrapper("http://loc
    alhost:3030/extract-dataset/update")
143.                sparql.setMethod(POST)
144.                sparql.setQuery("""
145.                    PREFIX dbo: <http://dbpedia.or
    g/ontology/>
146.                    PREFIX rdf: <http://www.w3.org
    /1999/02/22-rdf-syntax-ns#>
147.                    PREFIX dbr: <http://dbpedia.or
    g/resource/>
148.                    PREFIX xsd: <http://www.w3.org
    /2001/XMLSchema#>
149.                    """+insert+"""
150.                """)
151.
152.                sparql.setReturnFormat(JSON)
153.                results = sparql.query().convert()

154.                print(results)
```

Kode Sumber 4.10 Memasukkan data hasil ekstraksi ke Apache Jena Fuseki

	subject	predicate	object
1	<http://dbpedia.org/resource/2009_Honduras_earthquake>	<http://www.w3.org/2000/01/rdf-schema#label>	"2009 Honduras earthquake"
2	<http://dbpedia.org/resource/2009_Honduras_earthquake>	<http://dbpedia.org/ontology/place>	"La Ceiba"
3	<http://dbpedia.org/resource/2009_Honduras_earthquake>	<http://dbpedia.org/ontology/place>	<https://www.geonames.org/3608248>
4	<http://dbpedia.org/resource/2009_Honduras_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/La_Ceiba>
5	<http://dbpedia.org/resource/1905_Calabria_earthquake>	<http://www.w3.org/2000/01/rdf-schema#label>	"1905 Calabria earthquake"
6	<http://dbpedia.org/resource/1905_Calabria_earthquake>	<http://dbpedia.org/ontology/place>	"Italy"
7	<http://dbpedia.org/resource/1905_Calabria_earthquake>	<http://dbpedia.org/ontology/place>	<https://www.geonames.org/3175395>
8	<http://dbpedia.org/resource/1905_Calabria_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Italy>

Gambar 4.6 Contoh *triplets* pada Apache Jena Fuseki

4.3.6 Implementasi *Insert* Data ke Apache Jena Fuseki

Link *url* artikel DBpedia digunakan sebagai masukan untuk *insert* data ke dalam *triple store* Apache Jena Fuseki. Link *url* akan dijadikan sebagai nama *instance*. *Instance* artikel tersebut memiliki properti rdfs:label berisi nama artikel dan properti dbo:place yang berisi hasil ekstraksi nama tempat. Implementasi *insert* data ke dalam Apache Jena Fuseki dapat dilihat pada Kode Sumber 4.11. Pada kode sumber tersebut, awalnya lakukan ekstraksi nama tempat pada SPARQL DBpedia kemudian pada *infobox* dan terakhir pada deskripsi utama halaman Wikipedia. Hasil ekstraksi selanjutnya dikonversi menjadi sintaks *query insert* data. Sintaks *insert* data menggunakan SPARQL tersebut kemudian dikirim ke *endpoint* Apache Jena Fuseki sehingga hasil ekstraksi dapat tersimpan ke *triple store* Apache Jena Fuseki. Contoh data *input* ditunjukkan oleh Gambar 4.7 dan contoh *output*

hasil *insert* data ke Apache Jena Fuseki ditunjukkan oleh Gambar 4.8.

```
1.  from SPARQLWrapper import SPARQLWrapper, POST, JS
    ON
2.  import wptools
3.  from geotext import GeoText
4.  import requests
5.  import json
6.  import re
7.
8.  def call_geo_db(title, place):
9.      # mencari geonames
10.     url = 'http://api.geonames.org/searchJSON?'
11.     for m in place:
12.         params = {'username': "pii.rizvi",
13.                   'name_equals': m,
14.                   'featureClass': ['P', 'A'],
15.                   'maxRows': "1"}
16.
17.         e = requests.get(url, params=params)
18.         if e:
19.             pretty_json = json.loads(e.text)
20.         for item in pretty_json["geonames"]:
21.             place_db.append([(title, "https://www
    .geonames.org/{}".format(item['geonameId']))])
22.     # mencari DBpedia
23.     for n in place:
24.         n2 = n.replace(" ", "_")
25.         place_db.append([(title, "http://dbpedia.
    org/resource/{}".format(n2))])
26.
27. def call_desc(url_title, title):
28.     page = wptools.page(title)
29.     try:
30.         page.get_query()
31.         # mencari deskripsi artikel Wikipedia
32.         text = page.data['extext']
33.         # mencari entitas negara dan kota pada de
    skripsi
34.         places_cand = GeoText(text)
35.         # menyimpan data negara pada deskripsi
```

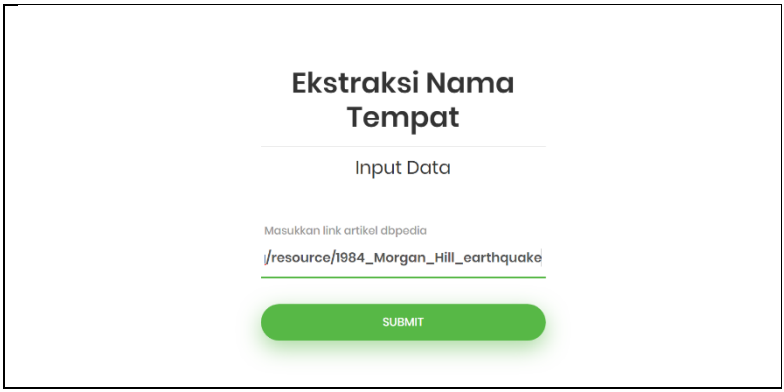
```
36.         a = places_cand.countries
37.         # menyimpan data kota pada deskripsi
38.         b = places_cand.cities
39.         # menggabungkan data negara dan kota dala
    m place, dan menggunakan set agar tidak ada data
    sama yang berulang
40.         place = set(a+b)
41.         for i in place:
42.         # menyimpan nama tempat pada place_db
43.             place_db.append([(url_title, i)])
44.         call_geo_db(url_title, place)
45.     except:
46.         pass
47.
48. place_db = []
49. def insert(url_title):
50.     sparql = SPARQLWrapper("http://dbpedia.org/sp
    arql")
51.     sparql.setQuery("""
52.         PREFIX dbo: <http://dbpedia.org/ontology/
    >
53.         PREFIX rdf: <http://www.w3.org/1999/02/22
    -rdf-syntax-ns#>
54.         PREFIX dbr: <http://dbpedia.org/resource/
    >
55.         PREFIX rdfs: <http://www.w3.org/2000/01/r
    df-schema#>
56.         PREFIX foaf: <http://xmlns.com/foaf/0.1/>

57.         SELECT ?place ?name ?geonames
58.         WHERE {
59.             { <""" + url_title + """> dbo:place ?
    place. }
60.                 UNION
61.                 { <""" + url_title + """> dbp:pla
    ce ?place. }
62.                 UNION
63.                 { <""" + url_title + """> dbo:loc
    ation ?place. }
64.                 UNION
65.                 { <""" + url_title + """> dbp:loc
    ation ?place. }
```

```
66.                 ?place rdfs:label ?name.
67.                 FILTER(lang(?name) = "en")
68.                 OPTIONAL{
69.                  ?place (owl:sameAs|^owl:sameAs) ?geonames.
70.                  FILTER(regex(str(?geonames), "sws.geonames.org" ) )
71.                 }
72.             }
73.     """)
74.
75.     sparql.setReturnFormat(JSON)
76.     results = sparql.query().convert()
77.     result = []
78.     for result in results["results"]["bindings"]:
79.         # menyimpan nama tempat pada place_db
80.         place_db.append([(url_title, result["place"]["value"])])
81.         place_db.append([(url_title, result["name"]["value"])])
82.         if 'geonames' in result:
83.             place_db.append([(url_title, result["geonames"]["value"])])
84.
85.     if result==[]:
86.         sparql = SPARQLWrapper("http://dbpedia.org/sparql")
87.         sparql.setQuery("""
88.             PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
89.             PREFIX foaf: <http://xmlns.com/foaf/0.1/>
90.             SELECT ?title
91.             WHERE {
92.             <""" + url_title + """> rdfs:label ?title.
93.             FILTER(lang(?title) = "en")
94.             OPTIONAL
95.             { <""" + url_title + """> foaf:name ?title. }
96.             }
```

```
97.         """)
98.         sparql.setReturnFormat(JSON)
99.         results = sparql.query().convert()
100.                title = []
101.                for result in results["results"]["
    bindings"]:
102.                    title = result['title']['value
    ']
103.                page = wptools.page(title)
104.                try:
105.                    page.get_parse()
106.                    # mengambil value pada tag pla
    ce di infobox
107.                    text_if = page.data['infobox']
    ['place']
108.                    # menghapus tanda kurung siku

109.                    res = re.findall(r'\[\[.*?\]\]
    ', text_if)
110.                    a = str(res).replace("'[[", ""
    ).replace("']]", "")
111.                    b = str(a).replace("[", "").re
    place("]", "")
112.                    inf = b.split(", ")
113.                    for i in inf:
114.                        # menyimpan nama tempat pa
    da place_db
115.                        place_db.append([(url_titl
    e, i)])
116.                    call_geo_db(lk[x], inf)
117.                except:
118.                    call_desc(url_title, title)
119.
120.            for x in range(len(place_db)):
121.                ent_ttl = []
122.                ent_plc = []
123.                for i in range(len(place_db[x])):

124.                    if "http" in place_db[x][i][1]
    :
125.                        ent_plc.append("<" + place
    _db[x][i][1] + ">")
```

```
126.                    ent_ttl.append(place_db[x][i][
    0])
127.                    ent_plc.append('"' + place_db[
    x][i][1] + '"')
128.                    insert = ""
129.                    for i in range(len(ent_ttl)):

130.                        text = ("INSERT DATA { <"
    + ent_ttl[i] + "> dbo:place " + ent_plc[i] + "};"
    )
131.                        insert += text
132.
133.                    sparql = SPARQLWrapper("http:/
    /localhost:3030/extract-dataset/update")
134.                    sparql.setMethod(POST)
135.                    sparql.setQuery("""
136.                          PREFIX dbo: <http://dbpe
    dia.org/ontology/>
137.                          PREFIX rdf: <http://www.
    w3.org/1999/02/22-rdf-syntax-ns#>
138.                          PREFIX dbr: <http://dbpe
    dia.org/resource/>
139.                          PREFIX xsd: <http://www.
    w3.org/2001/XMLSchema#>
140.                          """ + insert + """
141.                      """)
142.
143.                    sparql.setReturnFormat(JSON)
144.                    results = sparql.query().conve
    rt()
145.                    print(results)
```

Kode Sumber 4.11 Implementasi *insert* data ke Apache Jena Fuseki



Gambar 4.7 Contoh data *input*

	subject	predicate	object
1	<http://dbpedia.org/resource/1984_Morgan_Hill_earthquake>	<http://dbpedia.org/ontology/place>	"Santa Clara"
2	<http://dbpedia.org/resource/1984_Morgan_Hill_earthquake>	<http://dbpedia.org/ontology/place>	"Morgan Hill"
3	<http://dbpedia.org/resource/1984_Morgan_Hill_earthquake>	<http://dbpedia.org/ontology/place>	"Hamilton"
4	<http://dbpedia.org/resource/1984_Morgan_Hill_earthquake>	<http://dbpedia.org/ontology/place>	<https://www.geonames.org/3537906>
5	<http://dbpedia.org/resource/1984_Morgan_Hill_earthquake>	<http://dbpedia.org/ontology/place>	<https://www.geonames.org/5374764>
6	<http://dbpedia.org/resource/1984_Morgan_Hill_earthquake>	<http://dbpedia.org/ontology/place>	<https://www.geonames.org/2190324>
7	<http://dbpedia.org/resource/1984_Morgan_Hill_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Santa_Clara>
8	<http://dbpedia.org/resource/1984_Morgan_Hill_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Morgan_Hill>
9	<http://dbpedia.org/resource/1984_Morgan_Hill_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Hamilton>

Gambar 4.8 Contoh hasil *insert* data ke Apache Jena Fuseki

4.3.7 Implementasi *Query* dari Apache Jena Fuseki

Pengguna dapat melakukan *query* atau pencarian sama seperti *input* data yaitu berdasarkan link artikel DBpedia. Setelah pengguna memasukkan link artikel yang ingin dicari, dilakukan

query ke Apache Jena Fuseki. Implementasi *query* dari Apache Jena Fuseki dapat dilihat pada Kode Sumber 4.12. Pada kode sumber tersebut, setiap *subject* pada *triple store* Apache Jena Fuseki dilakukan filter terhadap *string keyword* yang ingin dicari. *Query* pada Apache Jena Fuseki diimplementasikan dengan menggunakan *statement select* menggunakan SPARQL. Hasil pencarian yang didapatkan dari Apache Jena Fuseki mengembalikan nilai *subject*, *predicate* dan *object*. Kemudian hasil ekstraksi link artikel yang benar-benar cocok dengan kata kunci akan ditampilkan. Contoh kata kunci pencarian ditunjukkan oleh Gambar 4.9 dan contoh *output* hasil *query* dari Apache Jena Fuseki ditunjukkan oleh Gambar 4.10.

```
1.  import requests
2.
3.  def query(keyword):
4.      result_list = []
5.      # query ke jena fuseki secara berulang untuk
    mengecek setiap kata kunci pencarian
6.      str_query = 'PREFIX rdf: <http://www.w3.org/1
    999/02/22-rdf-syntax-ns#> \
7.          PREFIX dbr: <http://dbpedia.org/resource/
    > \
8.          PREFIX xsd: <http://www.w3.org/2001/XMLSc
    hema#> \
9.          PREFIX rdfs: <http://www.w3.org/2000/01/r
    df-schema#> \
10.         \
11.         SELECT DISTINCT ?subject ?predicate ?obje
    ct \
12.         WHERE { \
13.             ?subject ?predicate ?object . \
14.             FILTER (?subject IN (<' +keyword +'>)
     && ?predicate IN (<http://dbpedia.org/ontology/p
    lace>)) \
15.         }'
16.
```

```
17.     response = requests.post('http://localhost:3030/extract-dataset/query', data={'query':str_query})
18.
19.     resp_json = response.json()
20.     subject = [i for i in [result["subject"]["value"] for result in resp_json["results"]["bindings"]]]
21.     predicate = [i for i in [result["predicate"]["value"] for result in resp_json["results"]["bindings"]]]
22.     object = [i for i in [result["object"]["value"] for result in resp_json["results"]["bindings"]]]
23.   return(subject, predicate, object)
```

Kode Sumber 4.12 Implementasi *query* dari Apache Jena Fuseki

Gambar 4.9 Contoh kata kunci pencarian

Hasil Ekstraksi

Subject	Predicate	Object
http://dbpedia.org/resource/1984_Morgan_Hill_earthquake	http://dbpedia.org/ontology/place	Santa Clara
http://dbpedia.org/resource/1984_Morgan_Hill_earthquake	http://dbpedia.org/ontology/place	Morgan Hill
http://dbpedia.org/resource/1984_Morgan_Hill_earthquake	http://dbpedia.org/ontology/place	Hamilton
http://dbpedia.org/resource/1984_Morgan_Hill_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/3537906
http://dbpedia.org/resource/1984_Morgan_Hill_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/5374784
http://dbpedia.org/resource/1984_Morgan_Hill_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/2190324
http://dbpedia.org/resource/1984_Morgan_Hill_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Santa_Clara
http://dbpedia.org/resource/1984_Morgan_Hill_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Morgan_Hill
http://dbpedia.org/resource/1984_Morgan_Hill_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Hamilton

Gambar 4.10 Contoh *output* hasil *query* dari Apache Jena Fuseki

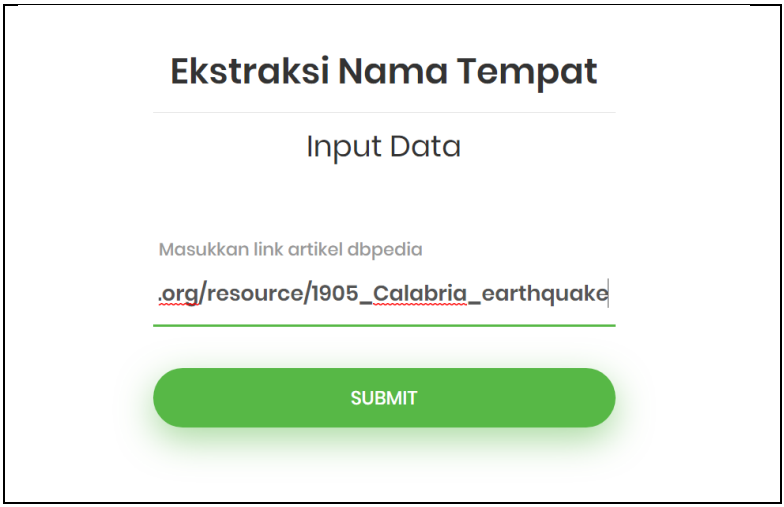
4.4 Implementasi Antarmuka Sistem

Implementasi antarmuka sistem dibuat untuk mempermudah uji coba dan evaluasi. Implementasi antarmuka sistem dibuat dengan menggunakan kerangka kerja Flask dan *triple store* Apache Jena Fuseki. Pada halaman depan atau beranda aplikasi seperti ditunjukkan pada Gambar 4.11, terdapat dua tombol yang memiliki fungsi berbeda, yang pertama untuk *input* data dan yang kedua untuk *query* data.

Gambar 4.11 Halaman beranda aplikasi

Pada halaman *input* data artikel, pengguna dapat memasukkan link artikel DBpedia pada *form* seperti ditunjukkan pada Gambar 4.12.

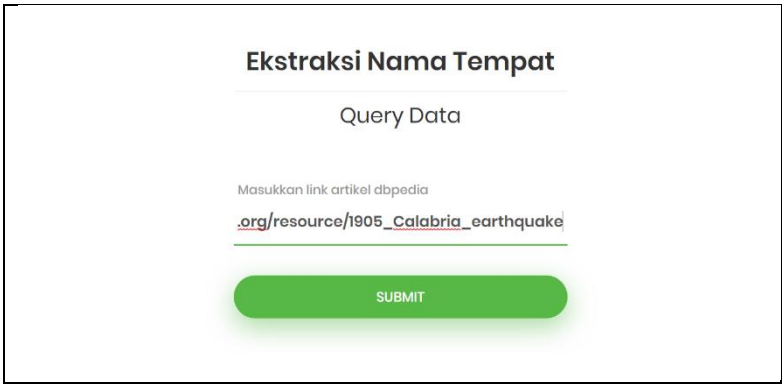
Setelah memasukkan link artikel DBpedia, pengguna harus menekan tombol Submit agar sistem melakukan proses ekstraksi serta menyimpan hasil esktraksi pada *triple store* Apache Jena Fuseki. Apabila proses penyimpanan pada *triple store* berhasil maka akan muncul pesan seperti ditunjukkan pada Gambar 4.13.



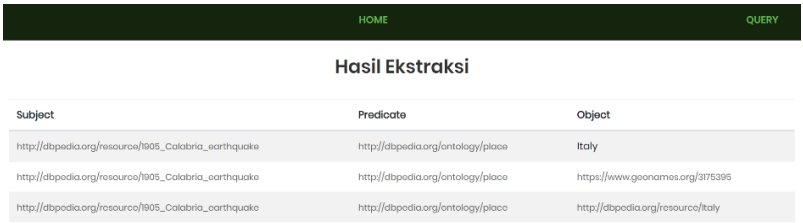
Gambar 4.12 Halaman *input* data

Sedangkan pada halaman *query* data, pengguna dapat memasukkan kata kunci pencarian pada *form* seperti ditunjukkan pada Gambar 4.14. Kata kunci pencarian sama dengan *input* data yaitu link artikel DBpedia.

Gambar 4.13 Pesan insert data pada Apache Jena Fuseki berhasil

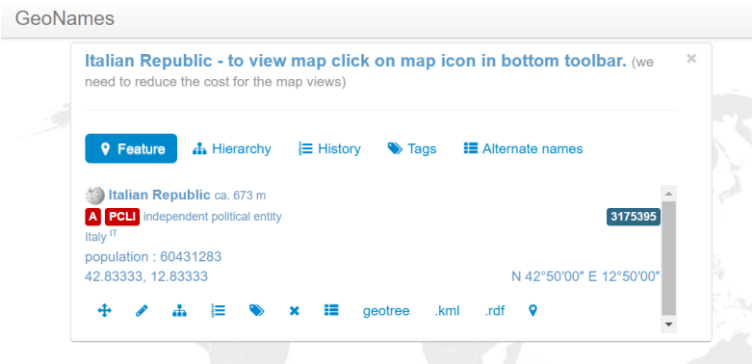


Gambar 4.14 Halaman *query* data



Gambar 4.15 Halaman hasil *query* data

Setelah memasukkan kata kunci berupa link artikel DBpedia, pengguna harus menekan tombol Submit agar sistem memproses pencarian. Halaman hasil pencarian artikel yang sesuai dengan kata kunci dapat dilihat pada Gambar 4.15. Hasil pencarian berupa pasangan *triplets* dari hasil ekstraksi nama tempat artikel yang jika diklik dapat merujuk ke link DBpedia atau Geonames pada nama tempat tersebut seperti ditunjukkan pada Gambar 4.16 apabila menekan link geonames dan Gambar 4.17 apabila menekan link DBpedia.



Gambar 4.16 Halaman Geonames Italy

Gambar 4.17 Halaman DBpedia Italy

[Halaman ini sengaja dikosongkan]

BAB V
UJI COBA DAN EVALUASI

Pada bab ini, akan dijabarkan hasil uji coba beserta evaluasi dari sistem yang telah dibuat. Pengujian dilakukan untuk mengetahui kinerja sistem dengan lingkungan uji coba yang telah ditentukan.

5.1. Lingkungan Uji Coba

Lingkungan uji coba pada pengerjaan tugas akhir ini dilakukan pada perangkat keras dan sistem operasi sebagai berikut:

- Processor : Intel® Core™ i5-8250U CPU @ 1.60GHz
- Installed RAM : 8,00 GB (7,88 GB usable)
- System Type : 64-bit operating system, x64-based processor
- Windows Edition : Windows 10 Home

5.2. Data Uji Coba

Data uji coba yang digunakan untuk evaluasi *query* spasial dilakukan terhadap beberapa objek yang tersimpan dalam *triple store* Apache Jena Fuseki. Data uji coba pada evaluasi hasil ekstraksi juga menggunakan data yang sudah tersimpan di Apache Jena Fuseki. Untuk evaluasi *input* data, menggunakan data masukan baru berupa link artikel DBpedia. Untuk evaluasi *query* data, juga menggunakan kata kunci pencarian berupa link artikel DBpedia.

5.3. Skenario Uji Coba

Subbab ini akan menjelaskan skenario uji yang telah dilakukan. Terdapat 4 skenario uji coba yang dilakukan. Berikut merupakan penjelasan setiap skenario pengujian:

1. Skenario Pengujian 1

   Dalam skenario ini akan dilakukan pengujian terhadap *query* spasial sistem pada Apache Jena Fuseki dan SPARQL Endpoint DBpedia.

2. Skenario Pengujian 2
   Dalam skenario ini akan dilakukan pengujian terhadap hasil ekstraksi pada Apache Jena Fuseki. Evaluasi *query* dilakukan dengan menghitung nilai *precision* dan *recall*, untuk setiap data uji.
3. Skenario Pengujian 3
   Dalam skenario ini akan dilakukan pengujian dengan data masukan baru berupa link artikel DBpedia ke dalam sistem.
4. Skenario Pengujian 4
   Dalam skenario ini akan dilakukan pengujian dengan melakukan *query* link artikel DBpedia. Pengujian dilakukan dengan memasukkan kata kunci link artikel DBpedia.

5.3.1 Skenario Pengujian 1

Pada skenario ini dilakukan evaluasi *query* spasial secara manual. Uji coba *query* spasial dilakukan terhadap beberapa objek yang tersimpan dalam *triple store* Apache Jena Fuseki. Terdapat 3 data uji yang digunakan untuk evaluasi *query* spasial seperti ditunjukkan pada Tabel 5.1.

Tabel 5.1 Data uji evaluasi *query* spasial

Nama	Data Uji
Data Uji 1	Indonesia
Data Uji 2	Australia
Data Uji 3	Malaysia

Query sparql pada Apache Jena Fuseki dan SPARQL Endpoint DBpedia pada Data Uji 1 ditunjukkan pada Gambar 5.1 dan hasil query dapat dilihat pada Gambar 5.2 dan Gambar 5.3. *Query* sparql pada Apache Jena Fuseki dan SPARQL Endpoint DBpedia pada Data Uji 2 ditunjukkan pada Gambar 5.4 dan hasil query dapat dilihat pada Gambar 5.5 dan Gambar 5.6. *Query* sparql pada Apache Jena Fuseki dan SPARQL Endpoint DBpedia pada

Data Uji 3 ditunjukkan pada Gambar 5.7 dan hasil query dapat dilihat pada Gambar 5.8 dan Gambar 5.9.

```
1 prefix owl: <http://www.w3.org/2002/07/owl#>
2 prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3
4 select distinct ?subject ?predicate ?object
5 where {
6    ?subject ?predicate ?object.
7   FILTER (?object IN (<http://dbpedia.org/resource/Indonesia>) &&
8         ?predicate IN (<http://dbpedia.org/ontology/place>))}
```

Gambar 5.1 Query SPARQL Data Uji 1

Showing 1 to 36 of 36 entries Search: Show All entries

	subject	predicate	object
1	<http://dbpedia.org/resource/2005_Nias–Simeulue_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
2	<http://dbpedia.org/resource/1996_Biak_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
3	<http://dbpedia.org/resource/1982_Flores_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
4	<http://dbpedia.org/resource/March_2007_Sumatra_earthquakes>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
5	<http://dbpedia.org/resource/2004_Nabire_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
6	<http://dbpedia.org/resource/2009_Andaman_Islands_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
7	<http://dbpedia.org/resource/2006_Pangandaran_earthquake_and_tsunami>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
8	<http://dbpedia.org/resource/2004_Indian_Ocean_earthquake_and_	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>

Gambar 5.2 Hasil Query Apache Jena Fuseki Data Uji 1

subject	predicate	object
http://dbpedia.org/resource/Aceh_War	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/2003–04_Indonesian_offensive_in_Aceh	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Papua_conflict	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/1990–98_Indonesian_military_operations_in_Aceh	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Darul_Islam_Rebellion	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Indonesian_National_Revolution	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Operation_Rimau	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Battle_of_Bantam	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Operation_Sumatra_Assist	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Battle_of_Biak	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Insurgency_in_Aceh	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Dutch_intervention_in_Bali_(1849)	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Dutch_intervention_in_Bali_(1906)	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Mapenduma_hostage_crisis	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Operation_Kraai	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Operation_Product	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/Operation_Tinombala	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia

Gambar 5.3 Hasil Query SPARQL Endpoint DBpedia Data Uji 1

Dari kedua hasil *query* pada Data Uji 1 dapat dilihat bahwa hasil *query* pada data yang tersimpan pada Apache Jena Fuseki terdapat 36 *entries*. Hasil uji coba menggunakan Data Uji 1 dapat dilihat pada Lampiran 1. Data tersebut jauh lebih banyak dibandingkan hasil *query* SPARQL Endpoint DBpedia yang hanya terdapat 17 *entries*. Hal itu disebabkan pada data Apache Jena Fuseki, sistem sudah dapat mengekstrak nama tempat jauh lebih banyak dengan menggunakan proses ekstraksi melalui DBpedia, *Infobox* dan deskripsi utama Wikipedia.

```
1  prefix owl: <http://www.w3.org/2002/07/owl#>
2  prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3
4  select distinct ?subject ?predicate ?object
5  where {
6     ?subject ?predicate ?object.
7    FILTER (?object IN (<http://dbpedia.org/resource/Australia>) &&
8          ?predicate IN (<http://dbpedia.org/ontology/place>))}
```

Gambar 5.4 Query SPARQL Data Uji 2

Showing 1 to 27 of 27 entries Search: Show All entries

	subject	predicate	object
1	<http://dbpedia.org/resource/2006_Pangandaran_earthquake_and_tsunami>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
2	<http://dbpedia.org/resource/2009_Papua_earthquakes>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
3	<http://dbpedia.org/resource/1960_Valdivia_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
4	<http://dbpedia.org/resource/Glacial_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
5	<http://dbpedia.org/resource/1922_Vallenar_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
6	<http://dbpedia.org/resource/2013_Solomon_Islands_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
7	<http://dbpedia.org/resource/1977_Sumba_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
8	<http://dbpedia.org/resource/2016_Sumatra_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
9	<http://dbpedia.org/resource/2010_Kalgoorlie-Boulder_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>

Gambar 5.5 Hasil Query Apache Jena Fuseki Data Uji 2

subject	predicate	object
http://dbpedia.org/resource/Castle_Hill_convict_rebellion	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Australia
http://dbpedia.org/resource/Eureka_Rebellion	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Australia
http://dbpedia.org/resource/United_States_Exploring_Expedition	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Australia
http://dbpedia.org/resource/Battle_of_Richmond_Hill	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Australia
http://dbpedia.org/resource/Convincing_Ground_massacre	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Australia
http://dbpedia.org/resource/Australian_frontier_wars	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Australia
http://dbpedia.org/resource/Hawkesbury_and_Nepean_Wars	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Australia

Gambar 5.6 Hasil Query SPARQL Endpoint DBpedia Data Uji 2

Dari kedua hasil *query* pada Data Uji 2 dapat dilihat bahwa hasil *query* pada data yang tersimpan pada Apache Jena Fuseki terdapat 27 *entries*. Hasil uji coba menggunakan Data Uji 2 dapat dilihat pada Lampiran 2. Data tersebut jauh lebih banyak dibandingkan hasil *query* SPARQL Endpoint DBpedia yang hanya terdapat 8 *entries*. Hal itu disebabkan pada data Apache Jena Fuseki, sistem sudah dapat mengekstrak nama tempat jauh lebih

banyak dengan menggunakan proses ekstraksi melalui DBpedia, *Infobox* dan deskripsi utama Wikipedia.

```
1  prefix owl: <http://www.w3.org/2002/07/owl#>
2  prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>
3
4  select distinct ?subject ?predicate ?object
5  where {
6     ?subject ?predicate ?object.
7    FILTER (?object IN (<http://dbpedia.org/resource/Malaysia>) &&
8          ?predicate IN (<http://dbpedia.org/ontology/place>))}
```

Gambar 5.7 Query SPARQL Data Uji 3

Showing 1 to 10 of 10 entries Search: Show All entries

	subject	predicate	object
1	<http://dbpedia.org/resource/2011_Aceh_Singkil_Regency_earthquakes>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
2	<http://dbpedia.org/resource/1976_Sabah_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
3	<http://dbpedia.org/resource/2015_Sabah_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
4	<http://dbpedia.org/resource/May_2010_Northern_Sumatra_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
5	<http://dbpedia.org/resource/Operation_Lalang>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
6	<http://dbpedia.org/resource/Landing_at_Labis>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
7	<http://dbpedia.org/resource/Floods_in_Malaysia>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
8	<http://dbpedia.org/resource/Malayan–Portuguese_war>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
9	<http://dbpedia.org/resource/2013_Southeast_Asian_haze>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
10	<http://dbpedia.org/resource/Battle_of_Malacca_(1641)>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>

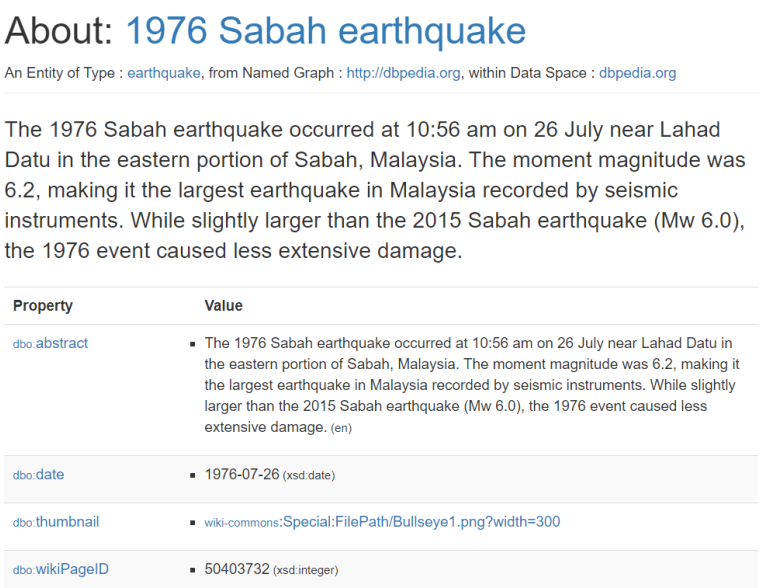
Gambar 5.8 Hasil Query Apache Jena Fuseki Data Uji 3

subject	predicate	object
http://dbpedia.org/resource/Landing_at_Labis	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/Malayan–Portuguese_war	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/Battle_of_Malacca_(1641)	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/Action_of_11_January_1944	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/Action_of_14_February_1944	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/Communist_insurgency_in_Sarawak	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/Battle_of_Labuan	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/13_May_Incident	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/Al-Ma'unah	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/Action_of_17_July_1944	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/Pudu_Prison_siege	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/Cross_border_attacks_in_Sabah	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/MT_Orkim_Harmony_hijacking	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/Jabidah_massacre	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia
http://dbpedia.org/resource/Action_of_13_November_1943	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Malaysia

Gambar 5.9 Hasil Query SPARQL Endpoint DBpedia Data Uji 3

Dari kedua hasil *query* pada Data Uji 3 dapat dilihat bahwa hasil *query* pada data yang tersimpan pada Apache Jena Fuseki terdapat 10 *entries*. Hasil uji coba menggunakan Data Uji 3 dapat dilihat pada Lampiran 3. Data tersebut berbeda seperti Data Uji sebelumnya yang jauh lebih banyak, kali ini data jauh lebih sedikit jika dibandingkan dengan hasil *query* SPARQL Endpoint DBpedia yang terdapat 16 *entries*. Hal itu disebabkan pada data Apache Jena Fuseki, meskipun sistem sudah dapat mengekstrak nama tempat jauh lebih banyak namun ada banyak data pada DBpedia yang belum dimasukkan dalam Apache Jena Fuseki disebabkan data terlalu banyak dan tidak bisa dimasukkan semua.

Untuk melihat hasil bahwa sistem tetap bekerja dengan baik, dapat dilihat pada Gambar 5.10, data artikel http://dbpedia.org/resource/1976_Sabah_earthquake tidak memiliki *property place* pada DBpedia sehingga pada *query* SPARQL DBpedia diatas tidak ada artikel tersebut. Namun, jika diekstraksi menggunakan *raw text* deskripsi utama Wikipedia terdapat nama tempat “Malaysia” sehingga sistem menandai “Malaysia” sebagai nama tempat dan Apache Jena Fuseki dapat mengenali artikel tersebut sebagai kejadian yang berlokasi di Malaysia.



About: 1976 Sabah earthquake

An Entity of Type : earthquake, from Named Graph : http://dbpedia.org, within Data Space : dbpedia.org

The 1976 Sabah earthquake occurred at 10:56 am on 26 July near Lahad Datu in the eastern portion of Sabah, Malaysia. The moment magnitude was 6.2, making it the largest earthquake in Malaysia recorded by seismic instruments. While slightly larger than the 2015 Sabah earthquake (Mw 6.0), the 1976 event caused less extensive damage.

Property	Value
dbo:abstract	The 1976 Sabah earthquake occurred at 10:56 am on 26 July near Lahad Datu in the eastern portion of Sabah, Malaysia. The moment magnitude was 6.2, making it the largest earthquake in Malaysia recorded by seismic instruments. While slightly larger than the 2015 Sabah earthquake (Mw 6.0), the 1976 event caused less extensive damage. (en)
dbo:date	1976-07-26 (xsd:date)
dbo:thumbnail	wiki-commons:Special:FilePath/Bullseye1.png?width=300
dbo:wikiPageID	50403732 (xsd:integer)

Gambar 5.10 Contoh data artikel

5.3.2 Skenario Pengujian 2

Pada skenario ini dilakukan evaluasi hasil ekstraksi. Evaluasi dilakukan dengan menghitung nilai *precision* dan *recall*, untuk setiap data uji. Terdapat 5 data uji yang digunakan untuk evaluasi hasil ekstraksi seperti ditunjukkan pada Tabel 5.2.

Tabel 5.2 Data uji evaluasi hasil ekstraksi

Nama	Data Uji	Keterangan
Data Uji 4	http://dbpedia.org/resource/2005_Nias–Simeulue_earthquake	Melalui proses *raw text*
Data Uji 5	http://dbpedia.org/resource/1933_Long_Beach_earthquake	Melalui proses *raw text*

Nama	Data Uji	Keterangan
Data Uji 6	http://dbpedia.org/resource/2003_Bingöl_earthquake	Melalui proses *raw text*
Data Uji 7	http://dbpedia.org/resource/Basmachi_movement	Melalui proses DBpedia
Data Uji 8	http://dbpedia.org/resource/Simba_Rebellion	Melalui proses *infobox*

Pengujian dilakukan dengan menghitung nilai *True Positive* (TP), *False Positive* (FP), dan *False Negative* (FN). Untuk pengujian menggunakan Data Uji 4 dapat dilihat pada Gambar 5.11, sistem mendapatkan enam data yang dideteksi sebagai nama tempat (Sibolga, Indonesia, Bangkok, Thailand, Jakarta dan March). Ke enam nama tempat tersebut kemudian dicek kembali secara manual, ternyata dari enam nama tempat tersebut ada satu yang bukan merupakan nama tempat (March) sehingga ini ditandai sebagai *False Negative* (FN). Dan dari deskripsi *raw text* diperoleh ada tiga nama tempat (Sumatra, Nias, Simeulue) yang tidak ditandai sebagai nama tempat oleh sistem, sehingga ketiga nama tempat tersebut ditandai sebagai *False Positive* (FP). Hasil evaluasi ekstraksi menggunakan Data Uji 4 dapat dilihat pada Tabel 5.3.

Hasil Ekstraksi

Subject	Predicate	Object
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	March
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/2643071
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/March
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	Sibolga
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	Thailand
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	Indonesia
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	Jakarta
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	Bangkok
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/1213855
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/1605651
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/1643084
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/1642911
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/1609350
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Sibolga
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Thailand
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Indonesia
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Jakarta
http://dbpedia.org/resource/2005_Nias-Simeulue_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Bangkok

Gambar 5.11 Hasil ekstraksi Data Uji 4

Tabel 5.3 Hasil evaluasi ekstraksi menggunakan Data Uji 4

Klasifikasi	**Nilai**	**Pengecekan Manual**	**Sistem**
TP	5	- Sibolga - Indonesia - Bangkok - Thailand - Jakarta	- Sibolga - Indonesia - Bangkok - Thailand - Jakarta - March
FN	1	March	
FP	3	- Sumatra - Nias - Simeulue	
Precision	**0,83**	**83%**	
Recall	**0,625**	**63%**	

Dari kasus tersebut maka dapat simpulkan bahwa sistem memiliki *precision* sebesar 83% dan *recall* sebesar 63% yang didapatkan dari perhitungan diatas.

Untuk pengujian menggunakan Data Uji 5 dapat dilihat pada Gambar 5.12, sistem mendapatkan empat data yang dideteksi sebagai nama tempat (Los Angeles, Long Beach, March, Newport). Ke empat nama tempat tersebut kemudian dicek kembali secara manual, ternyata dari empat nama tempat tersebut ada dua yang bukan merupakan nama tempat (March, Newport) sehingga nama tempat ini ditandai sebagai *False Negative* (FN). Dan dari deskripsi *raw text* diperoleh ada satu nama tempat (California) yang tidak ditandai sebagai nama tempat oleh sistem, sehingga nama tempat ini ditandai sebagai *False Positive* (FP). Hasil evaluasi ekstraksi menggunakan Data Uji 5 dapat dilihat pada Tabel 5.4.

Hasil Ekstraksi

Subject	Predicate	Object
http://dbpedia.org/resource/1933_Long_Beach_earthquake	http://dbpedia.org/ontology/place	Los Angeles
http://dbpedia.org/resource/1933_Long_Beach_earthquake	http://dbpedia.org/ontology/place	Newport
http://dbpedia.org/resource/1933_Long_Beach_earthquake	http://dbpedia.org/ontology/place	March
http://dbpedia.org/resource/1933_Long_Beach_earthquake	http://dbpedia.org/ontology/place	Long Beach
http://dbpedia.org/resource/1933_Long_Beach_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/5368381
http://dbpedia.org/resource/1933_Long_Beach_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/1495577
http://dbpedia.org/resource/1933_Long_Beach_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/2643071
http://dbpedia.org/resource/1933_Long_Beach_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/5367929
http://dbpedia.org/resource/1933_Long_Beach_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Los_Angeles
http://dbpedia.org/resource/1933_Long_Beach_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Newport
http://dbpedia.org/resource/1933_Long_Beach_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/March
http://dbpedia.org/resource/1933_Long_Beach_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Long_Beach

Gambar 5.12 Hasil ekstraksi Data Uji 5

Tabel 5.4 Hasil evaluasi ekstraksi menggunakan Data Uji 5

Klasifikasi	**Nilai**	**Pengecekan Manual**	**Sistem**
TP	2	- Los Angeles - Long Beach	- Los Angeles - Long Beach - March - Newport
FN	2	- March - Newport	
FP	1	California	
Precision	**0,5**	**50%**	
Recall	**0,67**	**67%**	

Dari kasus tersebut maka dapat simpulkan bahwa sistem memiliki *precision* sebesar 50% dan *recall* sebesar 67% yang didapatkan dari perhitungan diatas.

Untuk pengujian menggunakan Data Uji 6 dapat dilihat pada Gambar 5.13, sistem mendapatkan dua data yang dideteksi sebagai nama tempat (Bingol, Turkey). Ke dua nama tempat tersebut kemudian dicek kembali secara manual dan diperoleh keduanya merupakan nama tempat. Dan dari deskripsi *raw text* tidak terdapat nama tempat lain. Hasil evaluasi ekstraksi menggunakan Data Uji 6 dapat dilihat pada Tabel 5.5.

Hasil Ekstraksi

Subject	Predicate	Object
http://dbpedia.org/resource/2003_Bingöl_earthquake	http://dbpedia.org/ontology/place	Turkey
http://dbpedia.org/resource/2003_Bingöl_earthquake	http://dbpedia.org/ontology/place	Bingöl
http://dbpedia.org/resource/2003_Bingöl_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/298795
http://dbpedia.org/resource/2003_Bingöl_earthquake	http://dbpedia.org/ontology/place	https://www.geonames.org/321082
http://dbpedia.org/resource/2003_Bingöl_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Turkey
http://dbpedia.org/resource/2003_Bingöl_earthquake	http://dbpedia.org/ontology/place	http://dbpedia.org/resource/Bingöl

Gambar 5.13 Hasil ekstraksi Data Uji 6

Tabel 5.5 Hasil evaluasi ekstraksi menggunakan Data Uji 6

Klasifikasi	**Nilai**	**Pengecekan Manual**	**Sistem**
TP	2	- Bingol - Turkey	- Bingol - Turkey
FN	0	-	
FP	0	-	
Precision	**1**	**100%**	
Recall	**1**	**100%**	

Dari kasus tersebut maka dapat simpulkan bahwa sistem memiliki *precision* sebesar 100% dan *recall* sebesar 100% yang didapatkan dari perhitungan diatas.

Untuk pengujian menggunakan Data Uji 7, disebabkan proses ekstraksi diperoleh dari proses ekstraksi DBpedia, dapat disimpulkan bahwa *precision* sebesar 100% dan *recall* sebesar 100% karena sistem berhasil mengekstraksi berdasarkan property dengan baik.

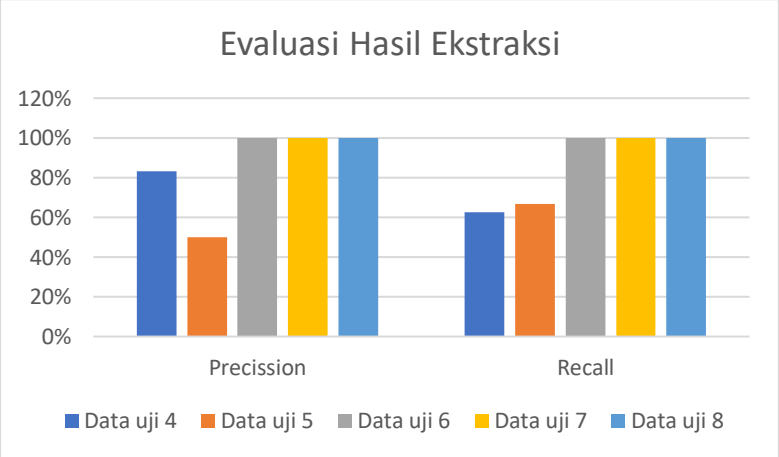
Untuk pengujian menggunakan Data Uji 8, hampir sama seperti Data Uji 7 disebabkan proses ekstraksi diperoleh dari proses ekstraksi *infobox*, dapat disimpulkan bahwa *precision* sebesar 100% dan *recall* sebesar 100% karena sistem berhasil mengekstraksi berdasarkan tag place dengan baik.

Hasil evaluasi ekstraksi menggunakan semua Data Uji dapat dilihat pada Tabel 5.6. Grafik evaluasi hasil ekstraksi menggunakan semua data uji ditunjukkan oleh Gambar 5.14.

Tabel 5.6 Hasil evaluasi ekstraksi menggunakan semua Data Uji

Data	***Precision***	***Recall***
http://dbpedia.org/resource/2005_Nias–Simeulue_earthquake	83%	63%

http://dbpedia.org/resource/1933_Long_Beach_earthquake	50%	67%
http://dbpedia.org/resource/2003_Bingöl_earthquake	100%	100%
http://dbpedia.org/resource/Basmachi_movement	100%	100%
http://dbpedia.org/resource/Simba_Rebellion	100%	100%



Gambar 5.14 Grafik evaluasi hasil ekstraksi menggunakan semua data uji

5.3.3 Skenario Pengujian 3

Pada skenario ini dilakukan uji coba *input* link artikel DBpedia menggunakan data masukan baru ke dalam sistem. Data masukan baru merupakan data uji coba di luar *dataset* yang sudah ada. Kondisi sistem sebelum dilakukan pengujian terdapat 32.272 pasang *triplets* yang tersimpan dalam *triple store* Apache Jena Fuseki seperti yang telah dijelaskan pada subbab 4.3.5.2.

Data baru yang akan diuji terdapat pada Tabel 5.7. Hasil uji coba *input* Data Uji 9 yang tersimpan pada Apache Jena Fuseki

ditunjukkan oleh Gambar 5.15. Hasil uji coba *input* data menggunakan Data Uji 10 yang tersimpan pada Apache Jena Fuseki ditunjukkan oleh Gambar 5.16. Hasil uji coba *input* data menggunakan Data Uji 11 yang tersimpan pada Apache Jena Fuseki ditunjukkan oleh Gambar 5.17.

Tabel 5.7 Data uji coba untuk *input* data

Nama	Data
Data Uji 9	Link artikel DBpedia: http://dbpedia.org/resource/First_Matabele_War
Data Uji 10	Link artikel DBpedia: http://dbpedia.org/resource/Miao_Rebellion_(1735–36)
Data Uji 11	Link artikel DBpedia: http://dbpedia.org/resource/Saltpeter_War_(Mexico)

	subject	predicate	object
1	<http://dbpedia.org/resource/First_Matabele_War>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Matabeleland>
2	<http://dbpedia.org/resource/First_Matabele_War>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Mashonaland>
3	<http://dbpedia.org/resource/First_Matabele_War>	<http://dbpedia.org/ontology/place>	"Matabeleland"
4	<http://dbpedia.org/resource/First_Matabele_War>	<http://dbpedia.org/ontology/place>	"Mashonaland"

Gambar 5.15 Hasil *input* Data Uji 9 yang tersimpan pada Apache Jena Fuseki

	subject	predicate	object
1	<http://dbpedia.org/resource/Miao_Rebellion_(1735–36)>	<http://dbpedia.org/ontology/place>	"China"
2	<http://dbpedia.org/resource/Miao_Rebellion_(1735–36)>	<http://dbpedia.org/ontology/place>	<https://www.geonames.org/1814991>
3	<http://dbpedia.org/resource/Miao_Rebellion_(1735–36)>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/China>
4	<http://dbpedia.org/resource/Miao_Rebellion_(1735–36)>	<http://dbpedia.org/ontology/place>	<https://www.geonames.org/1809445>
5	<http://dbpedia.org/resource/Miao_Rebellion_(1735–36)>	<http://dbpedia.org/ontology/place>	"Guizhou"
6	<http://dbpedia.org/resource/Miao_Rebellion_(1735–36)>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Guizhou>

Gambar 5.16 Hasil *input* Data Uji 10 yang tersimpan pada Apache Jena Fuseki

	subject	predicate	object
1	<http://dbpedia.org/resource/Saltpeter_War_(Mexico)>	<http://dbpedia.org/ontology/place>	"Colima"
2	<http://dbpedia.org/resource/Saltpeter_War_(Mexico)>	<http://dbpedia.org/ontology/place>	<https://www.geonames.org/4013516>
3	<http://dbpedia.org/resource/Saltpeter_War_(Mexico)>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Colima>
4	<http://dbpedia.org/resource/Saltpeter_War_(Mexico)>	<http://dbpedia.org/ontology/place>	"Sayula"
5	<http://dbpedia.org/resource/Saltpeter_War_(Mexico)>	<http://dbpedia.org/ontology/place>	<https://www.geonames.org/8582182>
6	<http://dbpedia.org/resource/Saltpeter_War_(Mexico)>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Sayula>

Gambar 5.17 Hasil *input* Data Uji 11 yang tersimpan pada Apache Jena Fuseki

Pada saat uji coba *input* data menggunakan Data Uji 9 hasil ekstraksi diperoleh dari proses ekstraksi melalui DBpedia, namun pada data uji ini nama tempat tersebut tidak memiliki id Geonames. Pada Data Uji 10 hasil diperoleh dari hasil ekstraksi melalui *infobox*. Sedangkan pada saat uji coba *input* data menggunakan Data Uji 11, diperoleh dari hasil ekstraksi melalui proses deskripsi

utama. *Value* yang terdiri dari *string*, link DBpedia dan link Geonames sudah terdeteksi meskipun nama tempat belum tentu benar dikarenakan proses hanya mengambil nama negara dan kota yang terdapat pada *raw text* deskripsi utama. Keseluruhan hasil ekstraksi tersebut berhasil di-*insert* ke *triple store* Apache Jena Fuseki.

5.3.4 Skenario Pengujian 4

Pada skenario ini dilakukan uji coba *query* dengan memasukkan kata kunci berupa link artikel DBpedia. Uji coba *query* dilakukan terhadap 32.272 pasang *triplets* yang tersimpan dalam *triple store* Apache Jena Fuseki seperti yang telah dijelaskan pada subbab 4.3.5.2. Contoh data kata kunci yang akan diuji terdapat pada Tabel 5.8. Hasil uji coba *query* dapat dilihat pada Tabel 5.9. Uji coba *query* berhasil menampilkan hasil ekstraksi yang sesuai dengan kata kunci pencarian.

Tabel 5.8 Data uji coba untuk *query* label

Nama	Data
Data Uji 12	http://dbpedia.org/resource/Battle_of_Aksu
Data Uji 13	http://dbpedia.org/resource/Battle_of_Lade
Data Uji 14	http://dbpedia.org/resource/2009_West_Java_earthquake
Data Uji 15	http://dbpedia.org/resource/2009_Yunnan_earthquake
Data Uji 16	http://dbpedia.org/resource/2009_Karonga_earthquakes
Data Uji 17	http://dbpedia.org/resource/Battle_of_Kursk

Tabel 5.9 Hasil uji coba *query*

Nama	Kata Kunci Pencarian	Hasil Pencarian
Data Uji 12	http://dbpedia.org/resource/Battle_of_Aksu	Xinjiang Aksu, Xinjiang http://dbpedia.org/resource/Xinjiang http://sws.geonames.org/1529047/ http://dbpedia.org/resource/Aksu,_Xinjiang http://sws.geonames.org/1529660
Data Uji 13	http://dbpedia.org/resource/Battle_of_Lade	Miletus http://dbpedia.org/resource/Miletus http://sws.geonames.org/322215
Data Uji 14	http://dbpedia.org/resource/2009_West_Java_earthquake	Java Bandung http://dbpedia.org/resource/Java http://sws.geonames.org/1642673 http://dbpedia.org/resource/Bandung http://sws.geonames.org/1650357 http://sws.geonames.org/6559695
Data Uji 15	http://dbpedia.org/resource/2009_Yunnan_earthquake	China https://www.geonames.org/1814991 http://dbpedia.org/resource/China
Data Uji 16	http://dbpedia.org/resource/2009_Karonga_earthquakes	Karonga Malawi https://www.geonames.org/235715 https://www.geonames.org/927384 http://dbpedia.org/resource/Karonga http://dbpedia.org/resource/Malawi

Nama	Kata Kunci Pencarian	Hasil Pencarian
Data Uji 17	http://dbpedia.org/resource/Battle_of_Kursk	Russian SFSR Soviet Union Kursk, Russia http://dbpedia.org/resource/Russian_SFSR http://dbpedia.org/resource/Soviet_Union http://dbpedia.org/resource/Kursk,_Russia

5.4. Evaluasi

Pada Skenario Pengujian 1 diperoleh bahwa sistem berhasil menampilkan hasil *query* spasial di *triple store* Apache Jena Fuseki lebih banyak dibandingkan dengan SPARQL Endpoint DBpedia. Hal ini disebabkan banyaknya proses ekstraksi yang dilakukan pada sistem untuk mendapatkan entitas nama tempat. Meskipun ada beberapa *query* spasial pada SPARQL Endpoint DBpedia yang memiliki event lebih banyak dibandingkan *triple store* Apache Jena Fuseki, sistem tetap berjalan dengan baik. Dan apabila event-event tersebut diinputkan pada sistem dan berhasil disimpan di *triple store*, sistem tetap dapat menampilkan hasil *query* yang lebih banyak dibandingkan SPARQL Endpoint DBpedia.

Pada Skenario Pengujian 2 diperoleh hasil terbaik saat menggunakan Data Uji 6, Data Uji 7, dan Data Uji 8 dengan nilai *precision* 100% dan *recall* 100% disebabkan sistem mampu mengekstraksi entitas nama tempat melaui proses DBpedia, *infobox* dan *raw text*.

Pada Skenario Pengujian 3 diperoleh bahwa sistem berhasil mengekstraksi nama tempat. Sistem juga berhasil menyimpan hasil ekstraksi dalam bentuk pasangan *triplets* pada Apache Jena Fuseki. Sistem otomatis menambahkan data masukan baru ke *triple store*.

Pada Skenario Pengujian 4 diperoleh bahwa sistem berhasil menampilkan data artikel yang sesuai dengan kata kunci pencarian yang dimasukkan pengguna ke dalam sistem.

BAB VI
KESIMPULAN DAN SARAN

Pada bab ini akan diberikan kesimpulan yang diperoleh selama pengerjaan tugas akhir dan saran mengenai pengembangan yang dapat dilakukan terhadap tugas akhir ini di masa yang akan datang.

6.1 Kesimpulan

Dari hasil pengamatan selama proses perancangan, implementasi, dan pengujian yang dilakukan, dapat diambil kesimpulan sebagai berikut:

1. Ekstraksi entitas geospasial pada artikel Wikipedia dengan memanfaatkan struktur metadata dilakukan dengan cara mengekstraksi pada struktur metadata DBpedia dan *infobox*. Ekstraksi menggunakan DBpedia diperoleh dari hasil pada properti penyimpan nama tempat. Sedangkan ekstraksi menggunakan *infobox* diperoleh dari hasil pada tag *place*.
2. Ekstraksi entitas geospasial pada artikel Wikipedia dengan menerapkan NLP (*Natural Language Processing*) dilakukan dengan cara mengekstraksi nama tempat dari *raw text* deskripsi utama Wikipedia. Proses ini menggunakan bantuan pustaka GeoText untuk mendapatkan entitas nama tempat.
3. Metadata artikel dimodelkan dalam bentuk pasangan *triplets* pada *triple store* Apache Jena Fuseki. Metadata tersebut dapat digunakan untuk membandingkan hasil *query* spasial pada SPARQL Apache Jena Fuseki dengan hasil *query* spasial SPARQL Endpoint DBpedia. Hasil ekstraksi yang diperoleh dapat dibuktikan pada Skenario Pengujian 1.
4. Pada Skenario Pengujian 2 diperoleh pada Data Uji 4 mendapatkan hasil *precision* 83% dan *recall* 63% dan Data Uji 5 mendapatkan hasil *precision* 50% dan *recall* 67%. Hal ini disebabkan sistem tidak mengekstraksi nama tempat secara benar pada *raw text*. Namun saat menggunakan Data

Uji 6, Data Uji 7, dan Data Uji 8 mendapatkan hasil terbaik dengan nilai *precision* 100% dan *recall* 100% disebabkan sistem mampu mengekstraksi entitas nama tempat melaui proses DBpedia, *infobox* dan *raw text*.

6.2 Saran

Berikut merupakan beberapa saran untuk pengembangan sistem di masa yang akan datang. Saran-saran ini didasarkan pada hasil perancangan implementasi, dan pengujian yang telah dilakukan.

1. Menambah jumlah data event yang digunakan untuk menyimpan data pada sistem yang dibuat mampu mengekstraksi lebih banyak entitas yang beragam.
2. Memperluas batasan dan ruang lingkup dataset dan juga menambah value pada hasil ekstraksi dengan tidak hanya menggunakan *string*, geonames dan DBpedia saja agar korpus data artikel yang dibuat menjadi lebih lengkap.

DAFTAR PUSTAKA

[1] "What Is Metadata,". [Online]. Available: http://whatis.techtarget.com/definition/metadata. [Accessed Desember 2019].

[2] J. Lehmann, R. Isele, M. Jakob, “DBpedia – A Large-scale, Multilingual Knowledge Base Extracted from Wikipedia,” University of Leipzig, Institute of Computer Science, Brox IT-Solutions GmbH, An der Breiten Wiese 9, D-30625 Hannover, Germany, Neofonie GmbH, Robert-Koch-Platz 4, D-10115 Berlin, Germany, 2015.

[3] "Owl-Web Ontology Language Features," 2003. [Online]. Available: http://www.w3.org/TR/owl-features/. [Accessed Desember 2019].

[4] "Geonames," [Online]. Available: http://www.geonames.org/. [Accessed Desember 2019]

[5] "Linked Data," [Online]. Available: https://www.w3.org/DesignIssues/LinkedData.html. [Accessed Desember 2019].

[6] “Apache Jena Fuseki,” [Online]. Available: https://jena.apache.org/documentation/fuseki2/. [Accessed Juni 2020].

[7] “Flask PyPI.”[Online]. Available:https://pypi.org/project/Flask [Accessed: 17-Dec-2019].

[8] J. Witmer and J. Kalita, "Mining Wikipedia Article Clusters for Geospatial Entities and Relationships," Department of Computer Science, University of Colorado, Colorado

Springs, 2009.

[9] "Precision and Recall,". [Online]. Available: https://en.wikipedia.org/wiki/Precision_and_recall [Accessed Desember 2019].

LAMPIRAN

1. Hasil uji coba menggunakan Data Uji 1

Berikut hasil *query* spasial dari Apache Jena Fuseki sebagai hasil uji coba menggunakan Data Uji 1.

Showing 1 to 36 of 36 entries Search: Show All entries

	subject	predicate	object
1	<http://dbpedia.org/resource/2005_Nias–Simeulue_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
2	<http://dbpedia.org/resource/1996_Biak_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
3	<http://dbpedia.org/resource/1982_Flores_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
4	<http://dbpedia.org/resource/March_2007_Sumatra_earthquakes>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
5	<http://dbpedia.org/resource/2004_Nabire_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
6	<http://dbpedia.org/resource/2009_Andaman_Islands_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
7	<http://dbpedia.org/resource/2006_Pangandaran_earthquake_and_tsunami>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
8	<http://dbpedia.org/resource/2004_Indian_Ocean_earthquake_and_	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
9	<http://dbpedia.org/resource/2009_Papua_earthquakes>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
10	<http://dbpedia.org/resource/1994_Java_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
11	<http://dbpedia.org/resource/2009_Talaud_Islands_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
12	<http://dbpedia.org/resource/2008_Sulawesi_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
13	<http://dbpedia.org/resource/1965_Ceram_Sea_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
14	<http://dbpedia.org/resource/Effect_of_the_2004_Indian_Ocean_earthquake_on_Sri_Lanka>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
15	<http://dbpedia.org/resource/2009_Sumatra_earthquakes>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
16	<http://dbpedia.org/resource/April_2010_Sumatra_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
17	<http://dbpedia.org/resource/2000_Enggano_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>

18	<http://dbpedia.org/resource/1992_Flores_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
19	<http://dbpedia.org/resource/1931_Southwest_Sumatra_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
20	<http://dbpedia.org/resource/1968_Sulawesi_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
21	<http://dbpedia.org/resource/1989_West_Papua_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
22	<http://dbpedia.org/resource/1629_Banda_Sea_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
23	<http://dbpedia.org/resource/1977_Sumba_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
24	<http://dbpedia.org/resource/2016_Sumatra_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
25	<http://dbpedia.org/resource/September_2007_Sumatra_earthquakes>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
26	<http://dbpedia.org/resource/1976_Papua_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
27	<http://dbpedia.org/resource/2010_Papua_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
28	<http://dbpedia.org/resource/1981_Irian_Jaya_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
29	<http://dbpedia.org/resource/May_2010_Northern_Sumatra_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
30	<http://dbpedia.org/resource/1994_Liwa_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
31	<http://dbpedia.org/resource/Aceh_War>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
32	<http://dbpedia.org/resource/2003–04_Indonesian_offensive_in_Aceh>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
33	<http://dbpedia.org/resource/Madiun_Affair>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
34	<http://dbpedia.org/resource/2013_Southeast_Asian_haze>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
35	<http://dbpedia.org/resource/Papua_conflict>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>
36	<http://dbpedia.org/resource/Aftermath_of_World_War_II>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Indonesia>

2. Hasil uji coba menggunakan Data Uji 2

Berikut hasil *query* spasial dari Apache Jena Fuseki sebagai hasil uji coba menggunakan Data Uji 2.

Showing 1 to 27 of 27 entries Search: Show All entries

	subject	predicate	object
1	<http://dbpedia.org/resource/2006_Pangandaran_earthquake_and_tsunami>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
2	<http://dbpedia.org/resource/2009_Papua_earthquakes>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
3	<http://dbpedia.org/resource/1960_Valdivia_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
4	<http://dbpedia.org/resource/Glacial_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
5	<http://dbpedia.org/resource/1922_Vallenar_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
6	<http://dbpedia.org/resource/2013_Solomon_Islands_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
7	<http://dbpedia.org/resource/1977_Sumba_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
8	<http://dbpedia.org/resource/2016_Sumatra_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
9	<http://dbpedia.org/resource/2010_Kalgoorlie-Boulder_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
10	<http://dbpedia.org/resource/1868_Arica_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
11	<http://dbpedia.org/resource/Vexatious_litigation>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
12	<http://dbpedia.org/resource/Day_of_Mourning_(Australia)>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
13	<http://dbpedia.org/resource/Empire_Air_Mail_Scheme>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
14	<http://dbpedia.org/resource/Bold_Alligator>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
15	<http://dbpedia.org/resource/Eager_Lion>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
16	<http://dbpedia.org/resource/Kimberley_Plan>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
17	<http://dbpedia.org/resource/Kingdom_Under_Fire:_Circle_of_Doom>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
18	<http://dbpedia.org/resource/Wool_classing>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
19	<http://dbpedia.org/resource/Hook_turn>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
20	<http://dbpedia.org/resource/Zimbabwean_cricket_crisis>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>

21	<http://dbpedia.org/resource/Bendigo_Petition>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
22	<http://dbpedia.org/resource/Forest_Creek_Monster_Meeting>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
23	<http://dbpedia.org/resource/Operation_Mazurka>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
24	<http://dbpedia.org/resource/Castle_Hill_convict_rebellion>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
25	<http://dbpedia.org/resource/Gurindji_strike>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
26	<http://dbpedia.org/resource/St_Kilda_Road_robberies>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>
27	<http://dbpedia.org/resource/European_Potato_Failure>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Australia>

3. Hasil uji coba menggunakan Data Uji 3

Berikut hasil *query* spasial dari Apache Jena Fuseki sebagai hasil uji coba menggunakan Data Uji 3.

Showing 1 to 10 of 10 entries Search: Show All entries

	subject	predicate	object
1	<http://dbpedia.org/resource/2011_Aceh_Singkil_Regency_earthquakes>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
2	<http://dbpedia.org/resource/1976_Sabah_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
3	<http://dbpedia.org/resource/2015_Sabah_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
4	<http://dbpedia.org/resource/May_2010_Northern_Sumatra_earthquake>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
5	<http://dbpedia.org/resource/Operation_Lalang>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
6	<http://dbpedia.org/resource/Landing_at_Labis>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
7	<http://dbpedia.org/resource/Floods_in_Malaysia>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
8	<http://dbpedia.org/resource/Malayan–Portuguese_war>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
9	<http://dbpedia.org/resource/2013_Southeast_Asian_haze>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>
10	<http://dbpedia.org/resource/Battle_of_Malacca_(1641)>	<http://dbpedia.org/ontology/place>	<http://dbpedia.org/resource/Malaysia>

BIODATA PENULIS

Rizvi Sofbrina, lahir pada tanggal 12 Januari 1999 di Banda Aceh. Penulis telah menempuh pendidikan di SMA Negeri 10 Fajar Harapan Banda Aceh. Penulis merupakan seorang mahasiswa yang sedang menempuh pendidikan di Departemen Informatika Institut Teknologi Sepuluh Nopember. Penulis aktif dalam organisasi mahasiswa, diantaranya adalah menjabat sebagai Staff Departemen Kewirausahaan Himpunan Mahasiswa Teknik Computer-Informatika (HMTC) ITS periode 2017/2018, Staff Departemen Entrepreneurship BEM FTIf ITS periode 2017/2018, Staff Annisa KMI ITS periode 2017/2018. Penulis juga aktif dalam kepanitiaan Schematics, sebagai staff dalam Biro National Seminar of Technology selama dua periode (2017 dan 2018). Penulis melaksanakan kerja praktik di PT. Pelindo Daya Sejahtera pada periode Juni 2019 – Agustus 2019. Dalam menyelesaikan pendidikan sarjana, penulis mengambil rumpun mata kuliah (RMK) Manajemen Informasi (MI). Untuk komunikasi penulis dapat dihubungi melalui surel rizvisfbrina@gmail.com.