PEMANFAATAN *SPARQL INFERENCING NOTATION (SPIN)* DALAM PENCARIAN BERBASIS SEMANTIK PADA DATA MAKANAN

The Utilization of SPARQL Inferencing Notation (SPIN) in Semantic Search Based on Food Data

Muhammad Briyan Wahyu Adityatama

Prodi S1 Teknik Informatika, Fakultas Teknik, Universitas Telkom
briyanwam@telkomuniversity.ac.id briyanwam@gmail.com

ABSTRAK

Semakin banyak informasi yang tersebar di internet menyebabkan semakin sulitnya pengguna dalam mencari informasi yang diinginkan dengan tepat, dikarenakan banyak mesin pencari yang belum menggunakan konsep semantik dalam memahami suatu kata atau kalimat pencarian. Indonesia memiliki beragam makanan khas daerah, dimana antara satu dan lainnya memiliki ciri khas masingmasing, selain itu jumlah tempat makan yang menyediakannya pun cukup banyak. Informasi ini jika tidak dikelola dengan baik, maka pengguna mengalami kesulitan dalam proses pencarian jika pada saat dibutuhkan, khusus para pencinta dan penikmat makanan khas daerah Indonesia. Oleh karena itu, dalam penelitian ini telah dikembangkan *prototype* aplikasi pencarian berbasis semantik pada data makanan dan tempat makan khas daerah indonesia menggunakan metode SPARQL *rules-based* dengan memanfaatkan SPARQL *Inferencing Notation* (SPIN). Setiap kalimat yang diinputkan pengguna diterjemahkan oleh proses *tokenization*, *stemming*, *filtering* dan dilanjutkan dengan representasi kalimat menggunakan *keywords* yang ada di *ontology* sehingga terbentuk SPARQL yang dapat dijalankan untuk melakukan *query*. Dari hasil proses menjalankan *query* memanfaatkan SPIN dapat disimpulkan rata-rata bahwa sistem dapat memberikan
hasil yang memuaskan (nilai *f-measure* rata-rata = 86,34 %) dalam menampilkan hasil pencarian dengan memanfaatkan SPIN dan menggunakan metode SPARQL *rules-based*.

Kata Kunci : SPARQL, SPIN, semantik, *query*, *prototype*, *rules-based*

ABSTRACT

The more information spread on the internet, the more difficult the user in searching the desired information properly, because many search engines have not used the concept of semantics in understanding a word or phrase search. Indonesia has a variety of regional specialties, which between one and the other has its own characteristics, in addition to the number of places to eat that provide it quite a lot. This information if not managed properly, then the user has difficulty in the search process if in times of need, especially the lovers and connoisseurs of typical Indonesian food. Therefore, in this research has been developed prototype semantic based search application on food data and eating place of indonesia region using SPARQL rule-based method by utilizing SPARQL Inferencing Notation (SPIN). Each sentence entered user translated by tokenization process, stemming, filtering and continued with the representation of sentences using keywords on the ontology to form SPARQL that can be run to perform queries. From the results of the process of running the query using SPIN can be concluded the average that the system can give satisfactory results (average f-measure value = 86.34%) by using SPIN and using the method SPARQL rules-based.

Keywords: SPARQL, SPIN, semantics, query, prototype, rules-based

1. PENDAHULUAN

1.1 Latar Belakang

Perkembangan Teknologi Informasi yang sangat cepat mendorong terjadinya perubahan di hampir semua bidang dalam kehidupan manusia, termasuk di bidang pencarian data. Hal tersebut dibuktikan dengan munculnya penyedia jasa *hosting*, baik yang berbayar maupun gratis mengakibatkan semakin banyak munculnya *website* baru dengan beragam informasi. Semakin banyak informasi, semakin banyak hal yang tidak kita perlukan yang juga beredar di internet, sehingga meski informasi semakin banyak tetapi tidak berbanding lurus dengan semakin mudahnya pengguna internet untuk mendapatkan informasi yang diinginkan. Yu Liyang [1] memberi pernyataan bahwa "Mesin pencari yang baik adalah yang mampu memisahkan hasil pencarian yang spesifik dan relevan dengan keinginan pengguna. Hasil pencarian yang tidak spesifik akan menghasilkan suatu daftar pencarian yang sangat banyak. Pengguna diharuskan melakukan penelusuran ulang dari daftar hasil pencarian tersebut untuk mencari yang sesuai dengan keinginannya". Salah satu kasus yang mengalami hal tersebut adalah kasus pencarian data tempat makan khas daerah. Sebagai contoh, jika pengguna ingin mencari penjual Dimsum di jalan Burangrang dengan mengetikkan "rumah makan penjual dimsum di jalan burangrang", maka mesin pencari akan menghasilkan daftar *web* yang berhubungan dengan Dimsum (meski tidak terletak di jalan Burangrang) dan berbagai hal tentang jalan Burangrang (meski tidak berhubungan dengan restoran dan Dimsum), meskipun ada diantara hasil yang muncul dan mengarah ke pertanyaan pengguna, akan tetapi banyak sekali hasil yang tidak

relevan ikut ditampilkan. Sehubungan dengan hal tersebut, Yu Liyang [1] memberikan pernyataan yang cukup keras dengan menyatakan bahwa *"Searching on the Web can be very frustrating"*. Tim Berners-Lee [2] mengatakan : "*The first step is putting data on the Web in a form that machines can naturally understand, or converting it to that form. This creates what I call a Semantic Web—a web of data that can be processed directly or indirectly by machines*." *Semantic web* akan membuat data yang ada di web tidak sekedar bisa dibaca oleh manusia, akan tetapi bisa juga dipahami oleh mesin, maka mesin akan mampu untuk memprosesnya sesuai kebutuhan. [3] Lei mengatakan bahwa salah satu tujuan penting dari *semantic web* adalah untuk membuat makna informasi eksplisit melalui semantik *mark-up*, sehingga memungkinkan akses yang lebih efektif untuk pengetahuan yang terkandung dalam lingkungan informasi heterogen, seperti *web*.

1.2 Perumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan di atas, perumusan masalahnya adalah sebagai berikut :

1. Bagaimana cara melakukan pencarian berbasis semantik pada data makanan ?
2. Bagaimana cara implementasi metode SPARQL based-rules pada pencarian berbasis semantik ?
3. Bagaimana cara menggunakan SPIN dalam pembuatan aplikasi pencarian berbasis semantik pada data makanan ?

1.3 Tujuan

Berdasarkan perumusan masalah di atas maka tujuan yang ingin dicapai adalah :

1. Menghasilkan hasil pencarian data makanan dan tempat makan khas daerah Indonesia yang tepat dan akurat sesuai dengan *keyword* yang diberikan.
2. Implementasi metode SPARQL *rules-based* dalam pencarian berbasis semantik pada data makanan menggunakan *SPARQL Inference Notation (SPIN)*.
3. Mampu menggunakan *SPARQL Inference Notation (SPIN)* dalam pembuatan aplikasi pencarian berbasis semantik.

1.4 Metodologi Penyelesaian Masalah

Berikut adalah metodologi yang digunakan dalam penyelesaian masalah dalam Tugas Akhir ini :

1. Studi Literatur
2. Analisis dan Pengumpulan Data
3. Perancangan Desain Sistem
   a. Membuat desain ontologi berdasarkan model logis yang dibuat.
   b. Membuat analisis pola kalimat pencarian yang akan digunakan untuk melakukan *query* pada sistem sesuai SPARQL *rules-based*.
   c. Membuat *use case* dan *activity diagram* representasi kalimat pencarian.
   d. Membuat aturan (*rule*) dan inferensi yang diperlukan sesuai SPARQL *rules-based*.
   e. Membuat rancangan antar muka sistem.
4. Konstruksi Sistem dan Implementasi
   a. Membangun *ontology* menggunakan aplikasi Protégé.
   b. Membangun aplikasi pencarian dengan bahasa pemrograman Java menggunakan SPIN API sesuai dengan perancangan yang sudah dibuat.
   c. Implementasi dilakukan untuk melihat apakah metode dapat diterapkan dalam perancangan dan digunakan kemudian dalam pembuatan aplikasi pencarian berbasis semantik.
   d. Implementasi dilakukan untuk menghasilkan hasil pencarian berbasis semantik dengan tepat dan akurat.
5. Pengujian
6. Penyusunan Laporan Tugas Akhir

2. KAJIAN PUSTAKA

2.1 *Semantic Search*

Menurut Grimes [4] *semantic search* pada intinya pencarian yang dibuat cerdas, pencarian yang berusaha meningkatkan akurasi dengan menghilangkan ambiguitas melalui pemahaman konteks. Ide utama dari pendekatan *semantic search* berasal dari pandangan kognitif terhadap dunia dimana terdapat asumsi
bahwa arti dari suatu teks (kata) bergantung kepada relasi konseptual terhadap obyek dalam dunia nyata dari pada relasi linguistik yang terdapat dalam teks atau kamus. Komponen penting dalam model ini adalah keberadaan struktur *concept* untuk memetakan deskripsi objek informasi dengan *concept* yang terdapat dalam *query*. Struktur *concept* dapat bersifat umum atau domain spesifik dan dapat dibuat dengan pendekatan manual atau otomatis. Tipe utama dari struktur *concept* yang digunakan dalam pendekatan *semantic search* antara lain struktur *taxonomy*, *thesauri* dan *ontology*. Penerapan Pengolahan Bahasa Alami atau *Natural Language Processing* (NLP) pada sistem *Information Retrieval* (IR) dipercaya dapat memperbaiki kinerja IR dengan cara memperbaiki representasi tes dalam *indexing* dan process *searching* dibandingkan model yang sepenuhnya

berbasis *string matching* [5]. Hal ini berdasarkan premis bahwa pemrosesan secara linguistik dapat menguak berbagai aspek semantik dari isi dokumen yang tidak dapat dilakukan dengan semata-mata mencacah kata, sehingga akan menghasilkan
hasil yang lebih akurat. Mandala [6] juga mengatakan bahwa penerapan NLP dalam proses *indexing* dan pemrosesan *query* antara lain bertujuan menghilangkan ambiguitas kata yaitu dengan mencari relasi semantik dari kata yang diperoleh untuk menghasilkan sinonimnya. Pada penelitian ini, NLP diterapkan sebagai bagian dari *rule* untuk memahami *keyword* yang diinputkan pada kolom pencarian sebagai bagian dari upaya memahamkan mesin komputer sebagaimana pemahaman manusia. Liddy [7] menyatakaan bahwa NLP secara teoritis adalah pengembangan berbagai teknik komputasi untuk menganalisa dan menampilkan teks dalam bahasa alami pada satu atau lebih tingkat analisis linguistik untuk mencapai tujuan manusia dalam hal bahasa yaitu menyelesaikan berbagai tugas dan aplikasi. Secara umum NLP merupakan kemampuan komputer untuk memproses bahasa lisan atau tulisan yang digunakan oleh manusia dalam percakapan sehari-hari.

2.2 *Protocol and RDF Query Language (SPARQL)*

SPARQL merupakan bahasa *query* untuk RDF/OWL yang menyediakan fasilitas untuk mengekstrak informasi dalam bentuk URI, *blank node* dan *literal*, mengekstrak *subgraf* RDF dan membangun *graf* RDF baru berdasarkan pada informasi dari *graf* yang di *query* [12]. *Query* SPARQL didasarkan pencocokan pola *graf*. Pola *graf* yang paling sederhana adalah *triple pattern* yang mirip dengan RDF *triple*. Klausa yang digunakan dalam query SPARQL diantaranya : a) *Prefix*, sebuah metode untuk menyingkat alamat dataset yang akan digunakan pada *query* tersebut; b) *Select*, digunakan untuk menentukan *result* apa saja yang akan ditampilkan; c) *Where*, berisi *triple pattern* atau *graph pattern*. *Triple pattern* merupakan *triple* yang terdiri dari subjek, predikat dan objek. Sedangkan *Graph Pattern* merupakan kumpulan dari *triple pattern*, maka didalam klausa where setidaknya harus terdapat ketiga komponen yaitu subjek, predikat dan objek.

2.3 *SPARQL Inferencing Notation (SPIN)*

SPIN adalah kependekan dari *SPARQL Inferencing Notation*, merupakan produk dari TopBraid. SPIN sudah diterima sebagai W3C *Member Submission* dan secara *de-facto* telah menjadi standar industri untuk mewakili SPARQL *rule and constraint* pada model *Semantic Web*. Selain itu SPIN juga menyediakan
kemampuan *meta-modeling* yang memungkinkan pengguna untuk membuat sendiri SPARQL *Function* dan *query template*. SPIN dilengkapi dengan *library* siap pakai yang berisi *function* yang biasa dipakai dalam pengembangan *semantic web* [8]. Knublauch [9], mengatakan bahwa SPIN menggabungkan konsep-konsep dari bahasa berorientasi objek, bahasa *query*, dan *rule-based system* untuk menggambarkan perilaku objek pada *web* data. Salah satu ide dasar SPIN adalah menghubungkan *class definition* dengan *query* SPARQL untuk mengendalikan *constraint* dan *rule* yang merumuskan perilaku yang diharapkan dari kelas-kelas. SPARQL digunakan karena merupakan standar WC3 merupakan acuan paling terkenal untuk *semantic query* di seluruh Data RDF. SPARQL juga telah digunakan secara luas di antara RDF *query* engine dan *graph store*, selain itu SPARQL juga menyediakan ekspresivitas yang cukup baik untuk *query* dan *general computation of data*. Untuk memfasilitasi penyimpanan dan pemeliharaan, *query* SPARQL disimpan dalam RDF *triples*, dengan menggunakan SPIN SPARQL *Syntax*.

2.4 Ontologi OWL dan RDF

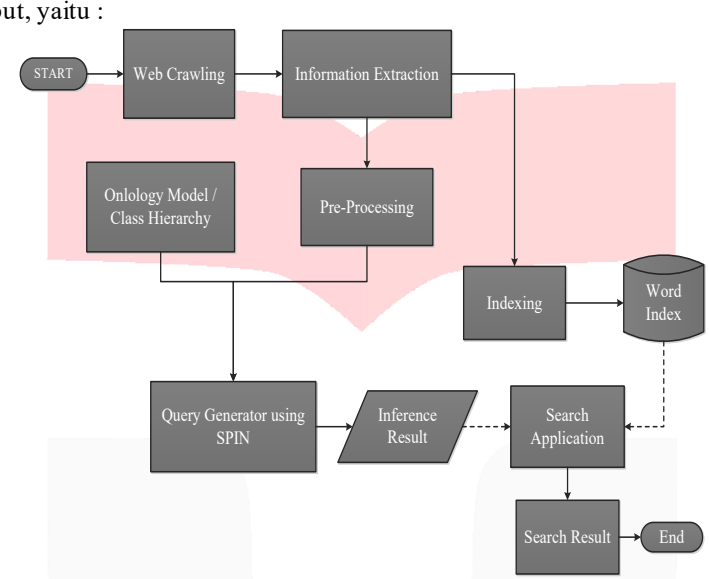
Agar data bisa dibaca dan dipahami oleh mesin maka penyimpanan data harus dipersiapkan dalam format khusus dengan pemodelan ontologi basis pengetahuan yang berbentuk OWL (*Web Ontologi Language)* atau RDF (*Resource Description Framework*). Pencarian dengan data yang berbentuk OWL atau RDF dilakukan dengan SPARQL, akan tetapi terkadang SPARQL tidak selalu memenuhi kebutuhan spesifik suatu pencarian [10], sebagai contoh: apabila dimiliki suatu RDF dengan relasi “Restoran XYZ menyediakan Dimsum”, kemudian dilakukan pencarian dengan kata kunci : “restoran yang menjual
makanan *Chinese*”, karena “*Chinese*” bukan nama makanan, melainkan jenis makanan, sedangkan antara “*Dimsum*” dengan “*Chinese*” tidak ada ekuivalensinya maka data tidak akan ditemukan. Untuk menyelesaikan hal tersebut perlu dibuat *triple* misalnya : “*Dimsum*” *rdfs:SubClassOf* “makanan *Chinese*” maka dengan menggunakan teknik *inference*, data restoran penjual *Dimsum* yang notabene makanan chinese bisa ditemukan. Terdapat berbagai cara untuk melakukan *inference* terhadap OWL atau RDF tetapi pada penelitian ini akan digunakan SPARQL *Inference Notation* (SPIN) dari *Top Braid* untuk
melakukan *inference*. SPIN adalah kumpulan RDF *vocabularies* yang dibangun diatas SPARQL yang merupakan standar W3C dan mempunyai kemampuan untuk mendefinisikan *function*, *template*, *constraint checking* dan melakukan *inference* pada semantik *web* [9]. Dengan SPIN akan dikembangkan suatu aplikasi pencarian semantik pada data restoran yang menghasilkan *search result* yang sesuai keinginan pengguna. Penelitian-penelitian yang telah dilakukan dengan menggunakan teknologi SPIN dalam semantik web sudah cukup banyak, berikut ini beberapa yang relevan dengan penelitian kali ini : Fürber dan Hepp [12], melihat bahwa kualitas data merupakan faktor kunci dalam bidang sistem informasi terutama berkenaan dengan data

yang harus dipisahkan dan tidak disertakan dalam proses bisnis, serta kualitas keputusan yang didasarkan pada output dari sistem informasi. Perkembangan saat ini yang memanfaatkan sematik *web* untuk pengelolaan data menjadikan hal yang penting untuk melakukan penelitian dalam mengidentifikasi kualitas data pada semantik *web* menggunakan SPARQL dan SPIN.
He, dkk [13], membuat penelitian dengan judul "*A Multimodal Restaurant Finder for Semantic Web*". Penelitian ini menghasilkan aplikasi pencarian restoran berbasis *semantic web service* dengan penginputan berbasis *gesture recognition* (dengan klik *mouse*) dan *speech recognition*. Process pencarian dalam basis pengetahuan menggunakan *clustering technique* KSOM (*Kohonen Self – Organizing Map*) Neural Network.

3. METODOLOGI DAN DESAIN SISTEM

3.1 Arsitektur Sistem

Arsitektur Sistem yang sesuai untuk pencarian berbasis *semantic* dengan pemanfaatan SPIN untuk mencari data makanan dan tempat makan khas daerah Indonesia yang ditunjukkan oleh Gambar 3-1 Komponen-komponen dalam arsitektur tersebut, yaitu :



Gambar Error! No text of specified style in document.**-1 Arsitektur Sistem**

3.1.1 *Web Crawling*

Web crawling adalah sebuah teknik suatu program atau *script otomat* yang relatif *simple*, yang dengan metode tertentu melakukan *scan* atau "*crawl*" ke semua halaman-halaman Internet untuk membuat *index* dari data yang dicarinya. Nama lain untuk *web crawl* adalah *web spider, web robot, bot, crawl* dan *automatic indexer. Web crawl* dapat digunakan untuk beragam tujuan. Penggunaan yang paling umum adalah yang terkait dengan *search engine*. *Search engine* menggunakan *web crawl* untuk mengumpulkan informasi mengenai apa yang ada di halaman-halaman *web* publik. Tujuan utamanya adalah mengumpukan data sehingga ketika pengguna Internet mengetikkan kata pencarian di komputernya, *search engine* dapat dengan segera menampilkan *website* yang relevan. Pada aplikasi pencarian semantik untuk mencari data makanan dan tempat makan khas daerah, tahapan *Web crawl* digunakan sebagai *dataset* yang di dalamnya berisi kalimat pembentuk kata kunci dilengkapi alamat URL halaman situs (*web*) dimana informasi itu berasal pertama kali. Tersedia banyak program / *script otomat* untuk melakukan *web crawling* di internet, namun pada penelitian ini akan digunakan program Twitter4J API yang memanfaatkan *database* Twitter untuk mencari semua data yang berkaitan dengan kata kunci yang diberikan dengan ruang lingkup data makanan dan tempat makan khas daerah Indonesia. Query yang digunakan dalam aplikasi adalah sebagai berikut : *"filter:links 'makanan & tempat makan bandung' since:2016-01-01"* Pemilihan *database* Twitter yang akan digunakan sebagai *dataset* karena dianggap praktis yaitu tidak berbayar (*free*) dan memiliki pencarian kata yang lebih mengarah ke pengalaman pengguna (*user*) internet yang pernah mengalami langsung atau hanya sekedar menyampaikan pengalaman dan informasi tentang makanan dan tempat makan khas daerah Indonesia, sehingga hasilnya diharapkan lebih mewakili kebutuhan pencarian yang akan digunakan, jika dibandingkan menggunakan Google API / Azure API yang harus membayar lisensi penggunaan.

3.1.2 *Information Extraction*

Dari hasil *Web Crawling* kemudian dilakukan proses pemisahan menggunakan fungsi *Regex*, untuk mendapatkan alamat URL dalam format standar Twitter (yaitu dengan ekstensi t.co) yang disertakan dalam setiap kalimat yang ditemukan yang mengandung kata atau topik mengenai data makanan dan tempat makan khas daerah Indonesia. Fungsi *Regex* atau biasa disebut *Regular Expression* adalah fungsi konstruksi bahasa untuk

mencocokkan teks berdasarkan pola tertentu, terutama untuk kasus-kasus kompleks, contoh misalnya mencari teks berawalan karakter tertentu, memiliki perulangan dari suatu teks, dan lain sebagainya. Fungsi Regex pun dapat digunakan untuk mencari alamat URL (dengan format ekstensi t.co) dan memisahkannya untuk kemudian akan digunakan sebagai acuan untuk mencari apapun informasi yang ada di dalam konten situs sesuai dengan URL yang bersangkutan. Dalam penelitian ini Fungsi *Regex* digunakan untuk mendapatkan URL tersebut dari hasil *web crawling*. Untuk membaca konten teks yang ada dalam sebuah situs sesuai URL yang ditunjukkan, akan digunakan Jsoup API. Jsoup API memiliki fungsi HTML *Parser* yang dapat digunakan untuk memisahkan konten teks dari konten lainnya dalam sebuah situs HTML. Dalam penelitian ini agar dapat dilakukan ekstraksi informasi dari dalam konten situs yang dimaksud, akan digunakan Jsoup API, sehingga didapat semua konten teks dari situs tersebut yang kemudian digunakan sebagai *dataset* sebagai acuan dalam pencarian pada proses selanjutnya. Tahapan *Information Extraction* adalah sebagai berikut :

1. *Get shortened URL*, menggunakan kelas Java *Matcher* (*java.util.regex.Matcher*). *Matcher* digunakan untuk mencari ekspresi reguler yang sama dalam berbagai teks. Pola *Regex* yang digunakan dalam aplikasi adalah sebagai berikut : *"((https?|ftp|gopher|telnet|file):((//)|(\\\\))+[\\w\\d:#@%/;$()~_?\\+=\\\\\\.&]*)"*

Dengan pola *Regex* di atas, kemudian dilanjutkan dengan penggunaan kelas Java *Pattern* (*java.util.regex.Pattern*). Kelas ini digunakan menggunakan metode *Pattern.match()* untuk memeriksa apakah teks (*String*) cocok dengan *Regex* (ekspresi reguler) yang diberikan.

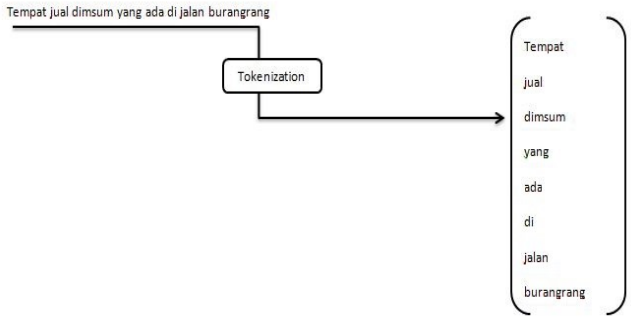
2. *HTML Parsing from shortened URL,* setelah semua *shortened URL* didapat dari hasil *web crawling*, tahap berikutnya adalah melihat isi konten HTML yang ditunjukkan oleh *shortened URL* dan memisahkan konten teks dengan konten lainnya dengan menggunakan pustaka Jsoup API.

3.1.3 *Pre-Processing*

Proses ini dibutuhkan agar mendapatkan hasil pencarian yang baik dan merupakan pra-syarat ke tahap *Query Generator* dengan menggunakan SPIN. Selain itu hasil *pre-processing* juga dibutuhkan untuk membangun *word index* dalam proses *indexing*. Ada 3 proses yang dilakukan dalam tahap *pre-processing* sebelum dilanjutkan ke tahap berikutnya, yaitu Proses *Tokenization*, Proses *Stemming* dan Proses *Filtering*, penjelasan detailnya sebagai berikut.

3.1.3.1 Proses *Tokenization*

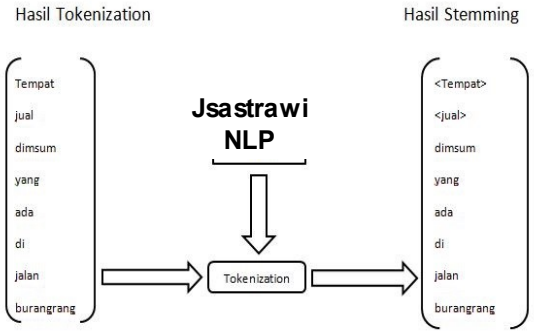
Proses *tokenization* / tokenisasi adalah proses pemecahan kalimat pencarian yang berbentuk *string* berdasarkan tiap kata yang menyusunnya. Pemisahan antar kata diidentifikasikan dengan adanya karakter spasi, sehingga proses tokenisasi dilakukan dengan memanfaatkan karakter spasi pada kalimat untuk memisahkan setiap kata. Setelah melalui proses tokenisasi maka kalimat pencarian akan menjadi sekumpulan kata yang diletakkan ke dalam sebuah array yang setiap elemenya terdiri dari kata dari kalimat tersebut. Proses tokenisasi akan di ilustrasikan pada Gambar 3-2.



Gambar Error! No text of specified style in document.**-2 Ilustrasi Tokenisasi Pada Kalimat**

3.1.3.2 Proses *Stemming*

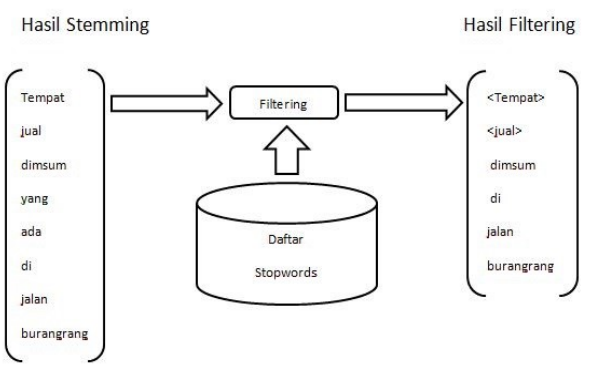
Proses *stemming* adalah proses pembentukan kata dasar. Setiap Kata yang diperoleh dari tahap *tokenization* dikenai proses *stemming*. *Stemming* bermanfaat mereduksi kata untuk menghindari ketidakcocokan dikarenakan adanya kata-kata yang berbeda namun memiliki makna dasar yang sama, kata-kata tersebut direduksi menjadi satu bentuk. Pada penelitian proses *stemming* akan menggunakan NLP (*Natural Language Processing*). Proses *Stemming* akan di ilustrasikan pada Gambar 3-3



Gambar Error! No text of specified style in document.**-3 Ilustrasi *stemming* pada kata-kata hasil tokenisasi.**

3.1.3.3 Proses *Filtering*

Proses *Filtering* adalah proses pembuangan *stopword* dan dimaksudkan untuk menghilangkan kata yang tidak memiliki arti atau tidak relevan. Kata- kata yang perlu dihapus dalam suatu kalimat disimpan tersendiri dalam ontologi. Apabila dalam kalimat terdapat kata yang masuk dalam daftar *stopword* maka kata tersebut tidak akan diproses lebih lanjut. Daftar *Stopword* dapat dilihat pada halaman lampiran. Sebagai contoh untuk proses penghilangan stopword dapat dilihat pada Gambar 3-4



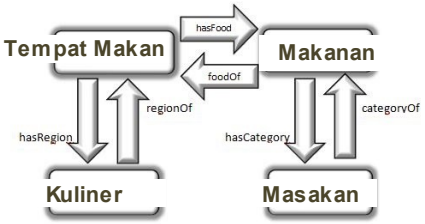
Gambar Error! No text of specified style in document.**-4 Ilustrasi penghilangan Stopword**

3.1.4 Perancangan Ontologi

Perancangan ontologi makanan dilakukan dengan mengikuti langkah-langkah seperti disarankan oleh Noy, dkk [14], yang diawali dengan penentuan domain dan konsep, kemudian pembentukan *class* dan relasinya, dilanjutkan dengan penentuan properti tiap *class*, penentuan *constraint* dan terakhir adalah pembentukan *instance*. Domain untuk penelitian ini adalah data restoran yang ada di wilayah Bandung. Sehubungan dengan kebutuhan pencarian berbasis semantik yang dilengkapi dengan *inference,* maka digunakan ontologi yang sesuai untuk kebutuhan tersebut. Ontologi makanan memiliki empat kelas yaitu :

a. *Tempat Makan* (informasi utama mengenai restoran seperti nama maupun alamat restoran),
b. *Makanan* (informasi utama mengenai nama- nama makanan yang menjadi trend di suatu daerah, dan yang pada umumnya dijual di tempat makan),
c. *Kuliner* (informasi mengenai area di daerah sebagai penunjuk lokasi restoran) dan
d. *Masakan* (informasi mengenai kategori atau jenis masakan, seperti misalnya masakan daerah, masakan tradisional, *sea food*, dan lain- lain).

Diagram Relasi antar kelas (*class diagram*) digunakan untuk menunjukkan interaksi antar kelas dalam sistem. Relasi akan mengijinkan suatu kelas mengetahui atribut, relasi dan hubungan yang ada pada kelas lainnya. Relasi antar kelas pada ontologi *Makanan* ditunjukkan oleh Gambar 3-5.

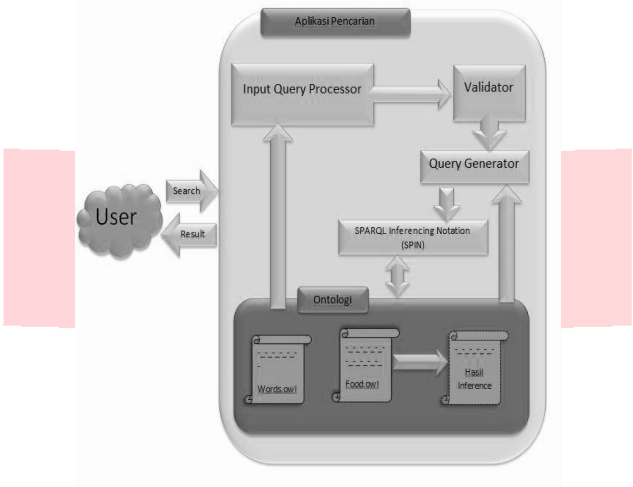


Gambar Error! No text of specified style in document.**-5 Diagram Relasi Antar Kelas**

Pada Gambar 3-5 terlihat bahwa kelas *Tempat Makan* memiliki relasi *hasFood* dengan kelas *Makanan* yang berarti sebuah restoran memiliki berbagai makanan. Sebaliknya kelas *Makanan* memiliki relasi *foodOf* dengan kelas *Tempat Makan* yang berarti sebuah makanan di kelas *Makanan* dimiliki satu atau lebih restoran. Kelas *Tempat Makan* juga memiliki relasi *hasRegion* dengan kelas *Kuliner* yang berarti setiap restoran memiliki satu atau lebih daerah kuliner. Sebaliknya kelas *Kuliner* memiliki relasi *regionOf* dengan kelas *Tempat Makan* yang berarti suatu daerah kuliner di kelas *Kuliner* dimiliki satu atau lebih Tempat Makan. Kelas *Makanan* memiliki relasi *hasCatagory* dengan kelas *Masakan* yang berarti setiap makanan memiliki satu atau lebih jenis atau kategori masakan. Sebaliknya kelas *Masakan* memiliki relasi *catagoryOf* dengan kelas *Makanan* yang berarti suatu kategori masakan di kelas *Catagory* dimiliki satu atau lebih makanan.

3.1.5 Query Generator (SPIN)

Pada Gambar 3-6 berikut akan ditunjukan alur proses *query generator* SPIN.



Gambar Error! No text of specified style in document.**-6 Proses Query Generator Menggunakan SPIN**

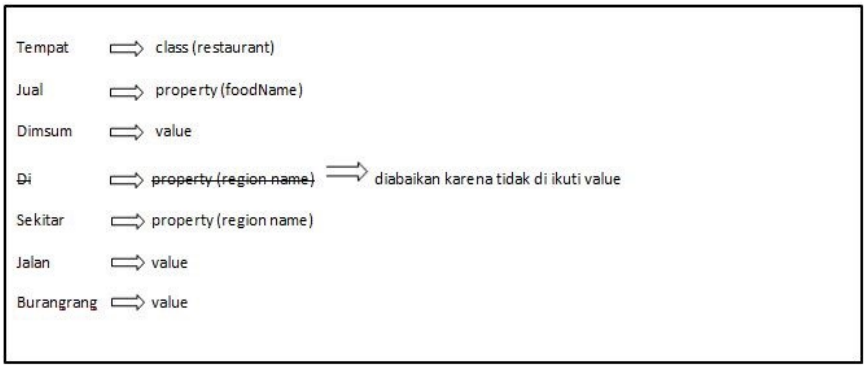
Penjelasan Proses Query Generator SPIN pada gambar 3-6 adalah sebagai berikut :

1) *User,* pengguna yang memberi masukan ke aplikasi pencarian berupa kalimat pencarian, kemudian diterima oleh *Input Query Processor.*
2) *Input Query Processor,* bertugas untuk melakukan penyesuaian kalimat dan *parsing* terhadap isian kata kunci dari pengguna, melakukan proses stemming dan filtering dengan mengacu pada ontologi word.owl.
3) *Validator,* bertugas untuk melakukan pengecekan hasil pemrosesan kalimat pencarian yang diisikan oleh *user*. Validasi dilakukan dengan mengacu pada suatu aturan yang telah ditentukan sebelumnya.
4) *Query Generator,* bertugas untuk membangun sebuah *query* berdasar data hasil *pemrosesan* yang valid dengan mengacu pada data *Triple* dan *Inference* yang terdapat di dalam ontologi.
5) *Ontology Foods.owl* dan *Words.owl, Foods.owl* adalah bagian yang menyimpan basis pengetahuan mengenai data makanan dalam bentuk *Triple*, sedangkan *Words.owl* adalah ontologi bantuan yang digunakan untuk menyimpan data *keyword, stopword* dan *directionword.*
6) Hasil *Inference,* berisikan hasil *inference* yang mengacu pada data *triple* yang terdapat di dalam *ontology Restaurants.owl.*
7) *SPARQL Inferencing Notation (SPIN),* bertugas untuk melakukan *query* dan Inferensi sesuai hasil dari *Query Generator* dan dilakukan terhadap isi *Ontology* OWL/RDF. Hasil *Query* yang diperoleh adalah merupakan hasil pencarian yang kemudian dikirimkan ke *user.*

3.1.6 Proses Representasi Kalimat

Proses representasi kalimat adalah proses untuk mencari makna atau maksud dari kalimat pencarian yang diisikan oleh pengguna. Proses dilakukan setelah proses *filtering*, sehingga hanya kata yang relevan saja yang digunakan untuk melakukan representasi dan sekaligus melakukan validasi kalimat. Proses validasi kalimat dilakukan dalam dua langkah, langkah pertama menghasilkan kalimat yang valid, maka dilakukan pengecekan untuk kata kedua, kata kedua harus berupa *class* atau sinonim dari *class*. Pencarian dilakukan dengan menggunakan daftar *Keywords* dari ontologi bantuan. Jika kata tidak ditemukan di dalam sinonim kata dalam daftar tersebut yang memiliki tipe *class* maka kalimat dianggap tidak valid. Apabila proses validasi ini menghasilkan kalimat yang valid maka dilanjutkan dengan mencari pola kalimat. Pada langkah kedua proses validasi kalimat di atas, akan menemukan *class* yang digunakan sebagai inti kalimat pencarian. Isi *class* ada empat yaitu : *makanan, tempat makan, kuliner dan masakan*. *Class Kuliner* yang ditemukan diartikan bahwa pengguna ingin menanyakan lokasi yang berkaitan dengan kuliner suatu daerah, informasi mengenai lokasi ini juga diberikan apabila pengguna membuat pertanyaan dengan inti kalimat pencarian berisi *class Tempat Makan* sehingga untuk *class Kuliner* akan dianggap sama dengan *class Tempat Makan*. Berdasarkan hal tersebut maka

kemudian dapat dicari padanan dari tiap kata mulai kata yang kedua (kata kedua adalah *class*). Pencarian mengacu ke daftar *keywords* dengan tipe *property* dan *forclass* berisi *class* yang ditemukan di kata kedua. Pencarian ini akan menghasilkan pola : CLASS – PROPERTY – VALUE – PROPERTY – VALUE – VALUE - ... dan seterusnya. Apabila ditemukan pola dengan *property* yang berurutan tanpa diselingi *value*, maka yang akan digunakan adalah *property* yang terakhir, yang langsung berhubungan dengan *value*. Pada Gambar 3-7 berikut akan ditampilkan hasil representasi kalimat.



Gambar Error! No text of specified style in document.**-7 Hasil Representasi Kalimat**

Sebagai contoh kalimat pencarian : "Tempat jual dimsum yang ada di jalan burangrang", setelah proses *filtering* akan menghasilkan bentuk : "Tempat jual dimsum sekitar jalan burangrang" dengan hasil representasi ditunjukkan Gambar 3-7.

SPARQL *Filter* dibuat berdasar pola yang ditemukan pada representasi kalimat sebelumnya dan dibentuk dengan format *property-value*. Sebagai contoh hasil representasi yang ditunjukkan Gambar 3-7 akan menghasilkan format seperti terlihat di Tabel 3-1.

Hasil Representasi	jual	dimsum	sekitar	jalan	burangrang
Pattern	proprety	value	property	value	value
Setences	foodName	dimsum	regionName	jalan	burangrang

Tabel Error! No text of specified style in document.**-1 Format Hasil Representasi**

Berdasar *Pattern* dan *Sentence* yang ada pada Tabel 3-1 akan dapat dibentuk SPARQL Filter yang dapat dilihat di Gambar 3-8.

```
FILTER( (regex(?foodName,'Dimsum','i') &&
        regex(?regionName,'jalan burangrang','i'))
```

Gambar Error! No text of specified style in document.**-8 Contoh SPARQL *Filter***

Salah satu kelebihan dari SPIN adalah kemudahan dalam melakukan *inference* dengan menggunakan SPARQL CONSTRUCT. Terdapat beberapa SPARQL CONSTRUCT yang dibuat, diantaranya adalah SPARQL CONSTRUCT untuk melakukan inferensi transitif, sebagai contoh adalah hubungan antara *class Restaurant* dengan *class Catagory*, Terdapat relasi *hasFood* antara *class Restaurant* dengan *class Food* dan relasi *hasCatagory* antara *class Food* dengan *class Catagory*. Sehingga bisa dilakukan suatu inferensi antara *class Restaurant* dengan *class Category* yang menghasilkan *spin:RuleProperty = foodCatagory*. *Script* SPARQL CONSTRUCT diletakkan di dalam OWL menggunakan *spin:rule*. Pada saat proses *Inferencing* dijalankan maka SPARQL CONSTRUCT akan menghasilkan *triple* baru, dan kemudian *triple* baru tersebut bisa diakses menggunakan perintah SPARQL biasa. SPARQL CONSTRUCT yang dibuat dapat dilihat pada Gambar 3-9.

```
//sparql construct untuk food_category
CONSTRUCT {
    ?s :foodCategory ?fc .
}
WHERE {
    ?s :hasFood ?f . ?f :hasCategory ?c . ?c :categoryName ?fc .
}
//sparql_construct untuk price_category
CONSTRUCT {
    ?sbj :priceCategory ?pc .
}
WHERE {
    ?sbj :lowestPrice ?h .
    BIND (IF((?h < 20000), "MURAH", "MAHAL") AS ?pc) .
```

Gambar Error! No text of specified style in document.**-9 Ilustrasi SPARQL Construct**

SPARQL CONSTRUCT juga digunakan untuk membuat rule pada *owl:inverseOf.* Seperti misalnya, properti *hasFood* yang merelasikan *class Restaurant* dengan *class Food* maka dibuat juga properti *foodOf* yang merupakan *inverse* dari *hasFood* dan merelasikan *class Food* dengan *class Restaurant.* Bentuk *script* nya dapat dilihat pada Gambar 3-10.

```
CONSTRUCT {
        ?r :foodOf ?s .
}
WHERE {
        ?p1 owl:inverseOf ?p2 .
        FILTER (?p2 IN (:hasFood)) .
        ?s ?p2 ?r .
}
```

Gambar Error! No text of specified style in document.**-10 Ilustrasi Script**

Hal yang sama juga dilakukan untuk properti *hasCatagory, catagoryOf,* dan properti *hasRegion, regionOf.*

3.1.7 *Indexing*

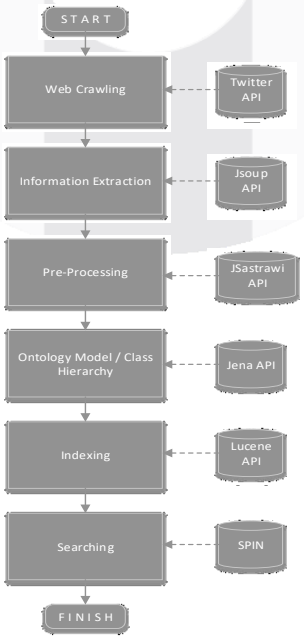
Proses *indexing* adalah proses yang dibutuhkan dalam pembuatan *search engine* (mesin pencari), yang mana pada proses ini akan dilakukan proses pengenaan indeks pada hasil ekstrasi dari tahap sebelumnya yang telah melalui tahapan *pre-processing*. Pada tahap pengenaan indeks pada setiap kata yang telah berhasil di tokenisasi akan dapat mempermudah pencarian sesuai dengan kata kunci yang diberikan. Pada tahap indexing ini akan digunakan NLP (*Natural Language Processing*) *Tools* dari *developer Lucene.*

3.2 Perancangan Sistem

Pada bagian ini akan dijelaskan mengenai bagian – bagian dalam perancangan sistem yang akan dilakukan dalam penelitian ini.

3.2.1 Gambaran Umum Sistem

Perancangan sistem yang dibangun berdasarkan studi literatur dan penelitian-penelitian terkait sesuai dengan kebutuhan. Penelitian dibuat dengan mengikuti sistematika yang terukur, mulai dari penentuan tahapan sesuai dengan metode yang akan digunakan, memperhatikan kelayakan penelitian, serta kemudahan pengguna dan pihak lain yang memanfaatkan hasil penelitian ini agar dapat digunakan secara mudah, praktis, namun memberikan hasil yang memadai. Perancangan sistem mengacu pada arsitektur sistem yang telah dibuat di tahap sebelumnya, dimana dimulai dari tahap *web crawling* yang digunakan selanjutnya untuk membentuk *indexing*, kemudian di tahap lainnya adalah perancangan ontologi yang akan digunakan pada tahapan *query generator* dengan menggunakan SPIN, yang mana kedua tahap ini akan digunakan pada tahap akhir pencarian (mesin pencari) sesuai kata kunci, dimana diharapkan hasilnya sesuai seperti yang ditentukan pada tujuan penelitian. Gambar 3-11 adalah blok diagram alir (*flowchart*) sistem yang akan dibangun untuk memenuhi kebutuhan penelitian untuk mencari data tempat makan dan makanan khas daerah Indonesia seperti yang dapat dilihat pada gambar 3-11.



Gambar Error! No text of specified style in document.**-11 Diagram Alir Sistem**

3.2.2 Bahan Penelitian

Dataset dibutuhkan sebagai sumber referensi pencarian yang digunakan dalam proses mencari konten *web* / Twitter yang sesuai dengan kata kunci yang diberikan. *Dataset* ini dibentuk dengan diawali proses *web crawling*, yaitu mendapatkan semua konten Twitter yang menyertakan *shortened URL* sesuai dengan kata kunci yang digunakan. Hasil konten Twitter yang telah berhasil didapatkan kemudian dilakukan proses pemisahan dengan menggunakan fungsi *Regex* untuk mendapatkan alamat *shortened URL*. Alamat *shortened URL* digunakan untuk mengambil semua konten teks dan dari konten *web*. Semua konten teks yang berhasil dipisahkan kemudian akan disusun menjadi sebuah dataset yang akan digunakan pada tahap berikutnya. Untuk mendapatkan akses ke Twitter melalui Twitter API, perlu dilakukan proses pendaftaran sebagai *developer* dengan mengakses halaman https://dev.twitter.com. Setelah proses pendaftaran berhasil dilakukan maka akan diperoleh konfigurasi *access key*. Gambar 3-12 adalah contoh aplikatif dari hasil proses *web crawling* dengan menggunakan Twitter API, seperti yang telah diterapkan pada aplikasi untuk mencari semua tempat makan dan makanan khas daerah Indonesia di daerah kuliner Kota Bandung.

Gambar Error! No text of specified style in document.**-12 Hasil Proses *Web Crawling***

Dari hasil *Web Crawling* di atas kemudian dilakukan proses pemisahan *link shortened URL* yang ada pada setiap baris konten web / *tweet* yang ditemukan. Hasil dari pemisahan *shortened URL* akan digunakan untuk mengakses konten *web* dan mengambil semua konten teks yang nantinya akan digunakan sebagai dataset atau bahan penelitian, proses ini dinamakan ekstraksi informasi. Pengambilan konten teks dari konten web sesuai *shortened URL* yang telah berhasil dipisahkan melalui proses *Regex* dilakukan dengan menggunakan Jsoup API, yang mana API ini memiliki fungsi *HTML Cleaner* yang dapat mengambil konten teks pada standar format halaman web berekstensi HTML. Pada tahap ini semua yang teridentifikasi sebagai konten teks akan diambil tanpa melihat apakah teks tersebut ada terkait dengan kata kunci baik secara terminologi maupun semantik, karena proses ini akan dilakukan di tahap berikutnya, yaitu proses *pre-processing*.

Hasil dari proses ekstraksi informasi pada aplikasi setelah melalui proses *web crawling* dapat dilihat pada Gambar 3-13.

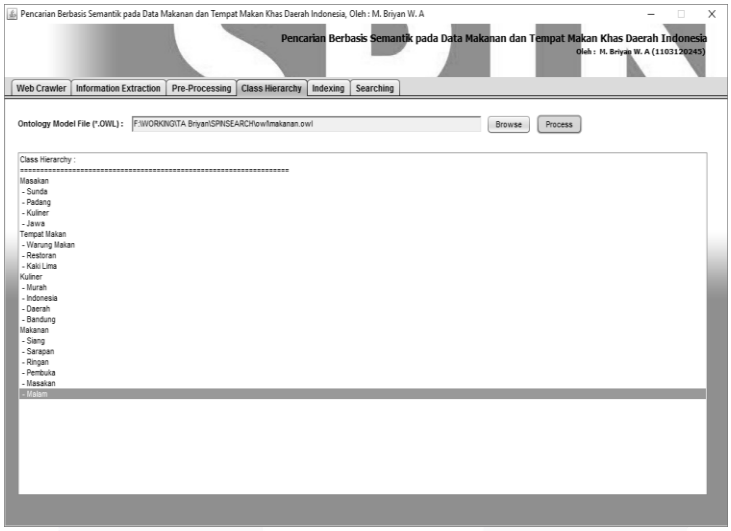
Gambar Error! No text of specified style in document.**-13 Hasil Proses Information Extraction**

Hasil dari proses ekstraksi informasi berupa kumpulan teks dengan berbagai macam arti dan makna yang didapat dari hasil *web crawling* yang hanya melibatkan konten yang berhubungan dengan tempat makanan dan makanan khas daerah yang berhasil didapatkan menggunakan Twitter API. Agar hasil ekstraksi informasi dapat digunakan sebagai *dataset*, tahap berikutnya adalah dengan melakukan *pre-processing*. Pada tahap *pre-processing* akan dilakukan proses tokenisasi, *stemming* dan *filtering*. Pada tahapan tokenisasi, semua konten teks akan dipisahkan

menjadi kata per kata dengan memisahkan berdasarkan jarak spasi yang ditemukan. Hasil dari tokenisasi kemudian akan dilakukan proses *stemming,* yang mana setiap kata hasil tokenisasi yang ditemukan akan dicari bentuk dasarnya, yaitu mendapatkan kata baku dan kata dasar dengan menggunakan JSastrawi API. Hasil akhir dari proses *stemming* kemudian dilakukan proses *filtering* dengan menggunakan daftar *stopword* yang telah dibuat. Proses *filtering* dilakukan untuk membuang semua kata yang tidak dibutuhkan dari hasil proses *stemming*, yang mana kata-kata ini dianggap tidak dapat digunakan pada tahap selanjutnya. Semua hasil *pre-processing* selanjutnya adalah *dataset* yang akan digunakan pada tahap *indexing* dan *query generator*.

3.2.3 *Ontology Model / Class Hierarchy*

Ontologi dibutuhkan sebagai *input* untuk proses *Query Generator*. Pada tahap ini file Ontologi yang akan digunakan (makanan.owl) akan dibaca dengan menggunakan OWL API dan kemudian didapatkan modelnya agar dapat diperolah *class hierarchy* dari ontologi ini dengan menggunakan Jena API. *Class Hierarchy* akan digunakan pada proses *query generator* untuk memperoleh keterhubungan antara notasi yang ada pada ontologi. Hasil dari pembacaan *class hierarchy* tersebut dapat dilihat pada Gambar 3-14 di bawah ini.

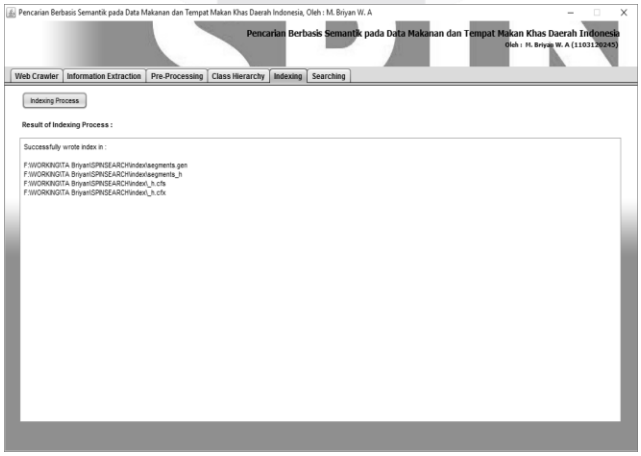


Gambar Error! No text of specified style in document.**-14 Hasil Proses Class Hierarchy**

Hasil di atas di peroleh dari pembacaan file ontologi. Didalam file ontologi akan dibaca semua kelas notasi yang ada dalam hierarki termasuk semua bagian sub di bawahnya secara lengkap.

3.2.4 *Indexing*

Tahap selanjutnya adalah proses *indexing* yaitu proses pencatatan konten web yang telah diperoleh dari hasil ekstraksi informasi yang dikelompokkan berdasarkan kata-kata yang diperoleh dari hasil *pre-processing* dan dibuat dalam bentuk daftar indeks untuk memudahkan pencarian berdasarkan hasil *query* yang digenerasi oleh SPIN. Proses *indexing* ini akan menggunakan Lucene API yang memiliki fasilitas untuk membuat indeks kata setiap konten yang akan didaftarkan ke dalam *database* kata yang simpan ke dalam sebuah file-file indeks pada perangkat penyimpanan di komputer, dalam hal ini *hard disk*. Hasil dari proses *indexing* tersebut dapat dilihat pada Gambar 3-15 di bawah ini.



Gambar Error! No text of specified style in document.**-15 Hasil Proses Indexing**

3.2.5 *Searching* (Pencarian)

Setelah semua proses di atas telah dilakukan dan diakhiri dengan terbentuknya file *indexing*, maka tahap selanjutnya adalah fitur *searching* (pencarian kata) berdasarkan kata kunci (*keyword*) yang dimasukkan. Dalam tahap ini, pada saat *user* memasukkan *keyword* untuk melakukan pencarian, maka sebelum mencari ke semua file hasil *indexing*, akan dilakukan pencarian semua kata yang memiliki keterkaitan secara ontologi dengan *keyword* yang dimasukkan. Proses ini akan dilakukan dengan menggunakan SPIN API untuk membentuk *query* sesuai SPARQL *rule based*, dimana SPIN API akan mencari semua kata yang terkait dengan *keyword* melalui *query* yang telah dibentuk melalui langkah inferensi dengan membandingkan notasi didalam ontologi, sehingga diperoleh hasil kata-kata yang dianggap berkaitan dengan *keyword*. Setelah kata-kata tersebut ditemukan, maka untuk menemukan konten-konten web / Twitter yang terkait dengan *keyword*, maka akan dilakukan pencarian dengan teknik *indexing* ke dalam file-file indeks yang telah terbentuk. Hasil pencarian tersebut dapat dilihat pada Gambar 3-16.

Gambar Error! No text of specified style in document.**-16 Hasil Proses Searching**

Dari hasil di atas dapat dilihat bahwa dengan memasukkan *keyword* "nasi goreng" diperoleh 7 hasil konten twitter yang berkaitan dengan *keyword* lengkap dengan *shortened URL* yang mana kontennya berisi tentang "nasi goreng" atau minimal terdapat kata "nasi goreng" dalam konten web sesuai dengan *shortened URL* yang dimaksud.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Berdasarkan dari hasil rumusan masalah, evaluasi sistem, dan analisis dengan menggunakan SPIN untuk mendapatkan hasil data tempat makan dan makanan khas daerah Indonesia dengan menggunakan kata kunci (*keyword*) tertentu oleh *user*, yaitu :

1. Penelitian ini berhasil menggunakan SPIN dilengkapi dengan penggunaan teknik *indexing* untuk memperoleh hasil pencarian data tempat makan dan makanan daerah indonesia yang ditunjukkan oleh hasil konten web / twitter yang ditunjukkan oleh *shortened URL* yang ditemukan.
2. Hasil evaluasi sistem dengan menggunakan kata kunci seperti pada contoh tabel 4-1, diperoleh hasil *f-measure* dan dari hasil *f-measure* tersebut dapat disimpulkan rata-rata bahwa sistem dapat memberikan hasil yang memuaskan (nilai *f-measure* rata-rata = 86,34 %) atau dapat disimpulkan tepat dan akurat yang dapat menampilkan hasil pencarian dengan memanfaatkan SPIN dan menggunakan metode SPARQL *rules-based*.
3. Dari hasil penelitian SPIN dapat dipakai dalam pembuatan aplikasi pencarian berbasis semantik, hal ini dapat disimpulkan dari berhasilnya didapatkan hasil pengujian yang memuaskan, dan dapat digunakan dalam pembuatan aplikasi pencarian berbasis desktop dengan menggunakan bahasa pemrograman Java.

5.2 Saran

Adapun saran yang ingin disampaikan untuk mengembangkan penelitian ini ialah sebagai berikut :

1. Menggunakan tools lain selain SPIN yang sama-sama dapat digunakan dalam aplikasi pencarian semantik, agar diperoleh hasil perbandingan yang aktual untuk mengukur tingkat manfaat penggunaan SPIN dalam penelitian
2. Menggunakan aplikasi berbasis *web / mobile* untuk memanfaatkan SPIN dalam aplikasi pencarian, namun tetap melihat ketersediaan dukungan dari *developer* yang mengembangkan SPIN.

DAFTAR PUSTAKA

[1] Yu, L., 2011, Developer Guide for Semantic Web, Springer-Verlag Berlin Heidelberg, Germany.

[2] Berner-Lee, T. dan Fischetti, M. , 1999, Weaving The Web, Harper San Francisco, USA

[3] Lei, Y., Uren, V., Motta, E., 2006, SemSearch: A Search Engine for the Semantic Web, 15th International Conference, EKAW 2006, Poděbrady, Czech Republic, October 2-6, 2006. Proceedings, pp 238- 245, Springer

[4] Grimes, Seth (January 21, 2010). "Breakthrough Analysis: Two + Nine Types of Semantic Search".

[5] Strzalkowski, T., Carballo, J.P., Karlgen, J., Hulth, A., Tapanainen, P., dan Lahtinen, T., 1999, Natural Language Information Retrieval, http://trec.nist.gov/pubs/trec8/papers/ge8adhoc2.pdf

[6] Mandala, R., 1999, Temu Kembali Informasi dengan Bantuan Analisis Linguistik, Proceeding of Information Processing and Management.

[7] Liddy, E.D., 2001, Natural Language Processing, In Enclopedia of Library and Information Science, Marcel Decker Inc, NY.

[8] Knublauch,H., 2009, SPIN SPARQL Inferencing Notation, www.spinrdf.org .

[9] Knublauch,H., Hendler, J.,A, Idehen, K., 2011, SPIN - Overview and Motivation, *http://www.w3.org/Submission/spin- overview/* .

[10] Zou,Y., Finin, T., dan Chen, H., 2005, *F- OWL: an Inference Engine for the Semantic Web, Proceedings of the Third International Workshop Formal Approaches to Agent-Based Systems (FAABS)*, Volume 3228, Springer-verlag , Greenbelt, MD, USA, p 238-248.

[11] Badra, F., Paul Servant, F. dan Passant, A, 2011, A Semantic Web Representation of a Product Range Specication based on Constraint Satisfaction Problem in the Automotive Industry, Proceedings of the 1st International Workshop on Ontology and Semantic Web for Manufacturing, Heraklion, Crete, Greece

[12] Fürber, C., Hepp, M., 2010, Using SPARQL and SPIN for Data Quality Management on the Semantic Web, Proceedings Business Information Systems 13th International Conference, Berlin, Germany.

[13] He, Y., Quan, T.T. dan Hui, S.C. (2006), *A Multimodal Restaurant Finder For Semantic Web*, In Addendum Contributions of International IEEE Conference on Computer Sciences (RIVF'06), Vietnam.

[14] Noy, N.F dan McGunness, D.L, 2001, Ontology Development 101: A Guide to creating your First Ontology, http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy- mcguinness.pdf , diakses tanggal 29 November 2011

[15] Bernes-Lee, T., Hendler, J., dan Lassila, O., 2001, The Semantis Web. American Scientific, USA
Tala, Z, 2003, A Study of Stemming Effect in Information Retrieval in Bahasa Indonesia, Institute for Technologies For Mapping Representation Of Ontologies, Springer, New York