JURNAL TEKNOLOGI TECHNOSCIENTIA ISSN: 1979-8415
Vol. 6 No. 2 Februari 2014

PEMANFAATAN SPARQL INFERENCING NOTATION (SPIN) DALAM PROTOTIPE PENCARIAN DATA RESTORAN BERBASIS SEMANTIK

Cosmas Haryawan[1]

[1]Prodi Sistem Informasi, STMIK AKAKOM Yogyakarta

Masuk: 3 Desember 2013, revisi masuk: 17 Januari 2014, diterima: 2 Februari 2014

ABSTRACT

Today, more and more information circulating on the internet that cause more difficulty in finding users desired information because many search engine have not used in understanding the concept of semantic search phrase. Application developed in this research is prototype of semantic search on restaurant data using SPARQL Inferencing Notation (SPIN). Search sentence must follow a rule that must be preceded by direction word (*tampil, cari, sebut*) and followed by a synonym of class which want to find (*restoran, makanan, kategori, lokasi*). Every search phrase which entered by user will be translated with tokenization process, stemming, stopword removal (filtering process) and followed by the representation of search sentence using keywords in support ontology (words.owl) thus formed run SPARQL to query data located on restaurant.owl ontology. The test result of this research show that the search application which built have an ability to handle wide variety of pattern of question and got the ratio of recall and precision 1:1, thus meaning that this application has a high effectiveness and efficiency in search result.

Keywords : ontology, owl, restaurant, semantic search, SPARQL, SPIN

INTISARI

Dewasa ini, semakin banyak informasi yang beredar di internet yang menyebabkan semakin sulitnya pengguna dalam mencari informasi yang diinginkan dikarenakan banyak mesin pencari yang belum menggunakan konsep semantik dalam memahami kalimat pencarian. Aplikasi yang dikembangkan dalam penelitian ini berupa prototipe untuk pencarian semantik pada data restoran dengan memanfaatkan SPARQL *Inferencing Notation* (SPIN). Kalimat pencarian harus mengikuti suatu aturan yaitu harus diawali dengan kata perintah (tampil, cari, sebut) dan diikuti dengan sinonim dari *class* yang dicari (restoran, makanan, kategori, lokasi). Setiap kalimat pencarian yang diisikan pengguna diterjemahkan dengan melakukan proses *tokenization, stemming*, penghilangan *stopword (*proses *filtering)*, dan dilanjutkan dengan representasi kalimat menggunakan *keywords* yang ada di ontologi bantuan *(words.owl)* sehingga terbentuk SPARQL yang dapat dijalankan untuk melakukan *query* data yang terletak di ontologi restaurants.owl. Hasil pengujian menunjukkan bahwa aplikasi pencarian yang dibuat mampu menangani berbagai variasi pola pertanyaan dengan nilai rasio Recall dan Precision adalah 1:1, ini berarti aplikasi ini memiliki efektifitas dan efisiensi yang tinggi dalam hasil pencariannya.

Kata Kunci : ontologi, owl, restoran, pencarian semantik, SPARQL, SPIN

PENDAHULUAN

Perkembangan Teknologi Informasi, terutama internet, dewasa ini sangat cepat. Semakin banyaknya penyedia jasa hosting, baik yang berbayar maupun gratis menjadikan semakin banyak munculnya website baru dengan beragam informasi. Banyaknya informasi yang beredar di internet sudah diprediksi jauh sebelumnya seperti yang disampaikan oleh Berners-Lee dan Fischetti (1999) yang mengutip artikel dari Paul Krill dengan judul "*Overcoming Information Overload*," dari majalah

InfoWorld terbitan 7 January 2000. Semakin banyak informasi, semakin banyak hal 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. 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 (Yu, 2011).

Salah satu kasus yang mengalami hal tersebut adalah kasus pencarian data restoran atau rumah makan. Sebagai contoh, jika pengguna ingin mencari penjual sushi di jalan kaliurang dengan mengetikkan : "restoran penjual sushi di jalan kaliurang", maka mesin pencari akan menghasilkan daftar web yang berhubungan dengan sushi (meski tidak terletak di jalan kaliurang) dan berbagai hal tentang jalan kaliurang (meski tidak berhubungan dengan restoran dan sushi), 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 dalam buku "Developer Guide to Semantic Web" (2011) memberikan pernyataan yang cukup keras dengan menyatakan bahwa *"Searching on the Web can be very frustrating"*.

Tim Berners-Lee dalam buku "Weaving The Web" (1999) 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. Lei (2006) 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. Lebih lanjut Lei (2006) menegaskan bahwa pencarian semantik (*semantic search*) memainkan peran penting dalam mewujudkan tujuan ini, karena menjanjikan untuk menghasilkan jawaban yang tepat untuk pertanyaan pengguna dengan mengambil manfaat dari ketersediaan informasi semantik secara eksplisit.

Penggunaan *semantic search* ini akan mengakibatkan hasil pencarian bisa lebih relevan sesuai yang diinginkan oleh pengguna. 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 (Zou, 2005), sebagai contoh: apabila dimiliki suatu RDF dengan relasi "Restoran XYZ menyediakan Sushi", kemudian dilakukan pencarian dengan kata kunci : "restoran yang menjual makanan jepang", karena "jepang" bukan nama makanan, melainkan jenis makanan, sedangkan antara "sushi" dengan "jepang" tidak ada ekuivalensinya maka data tidak akan ditemukan. Untuk menyelesaikan hal tersebut perlu dibuat *triple* misalnya : "sushi" *rdfs:SubClassOf* "makanan jepang" maka dengan menggunakan teknik *inference*, data restoran penjual sushi yang notabene makanan jepang 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. (Knublauch, dkk, 2011). 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 :

Badra, dkk (2011), melakukan penelitian di perusahaan otomotif Renault mengenai *Product Range Spesification* (PRS) yang dimodelkan sebagai *Constraint Satisfaction Problem* (CSR).

Penelitian bertujuan untuk mempelajari mempresentasikan PRS menggunakan Semantik Web. Penelitian ini juga membahas tentang penggunaan constraint di OWL dan penerapan SPIN untuk menyelesaikan masalah PRS tersebut di atas.

Fürber dan Hepp (2010), 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 (2006), 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.

Perbandingan penggunaan *rule* dalam *Islamic Finance* berbasis semantik dilakukan oleh Mamadolimova, dkk (2011). Paper tersebut membandingkan penggunaan SWRL, SPIN dan JenaRules untuk pemodelan dalam bidang *Islamic Finance*. Sementara itu Dameron (2013), menggunakan SPIN untuk pengembangan model pada pemilihan pasien dengan kriteria untuk tes klinik, SPIN dipilih karena mampu menggunakan *rule* dengan bahasa yang sama dengan *schema* dan data yang tersimpan di RDF. Sedangkan Su, C (2013) memanfaatkan SPIN untuk mengembangkan model ontologi dalam bidang pelayanan rencana diet

METODE

Menurut Grimes (2010) *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 ontologi.

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* (Strzalkowski, dkk, 1999). 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 (1999) 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 (2001) 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.

SPARQL *Protocol and RDF Query Language* (SPARQL) adalah sebuah protokol dan bahasa query untuk *Semantic Web's resources*. Sebuah *query* yang menggunakan SPARQL dapat terdiri atas *triple patterns*, *conjunction* (*or*), dan *disjunction* (*and*) (DuCharme, 2013).

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*. *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, Klausa Prefix merupakan sebuah metode untuk menyingkat alamat dataset yang akan digunakan pada *query* tersebut; b) Select, klausa Select digunakan untuk menentukan *result* apa saja yang akan ditampilkan; c) Where, klausa 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.

Sebagai contoh diberikan *query* sebagai berikut :

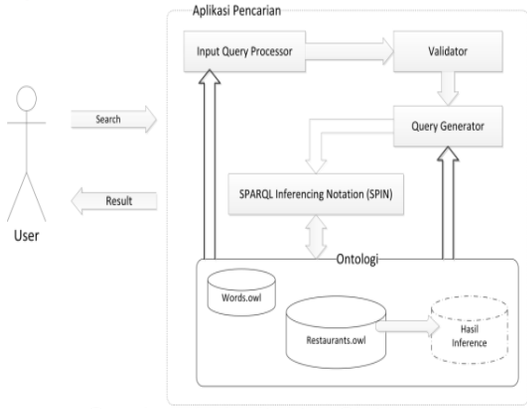
```
PREFIX foaf:
<http://xmlns.com/foaf/0.1/>
SELECT ?name ?email
WHERE {
  ?person a foaf:Person.
  ?person foaf:name ?name.
  ?person foaf:mbox ?email.
}
```

Query di atas akan menghasilkan daftar nama dan email dari dataset.

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.* (Knublauch, 2009)

Knublauch,dkk (2011), 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

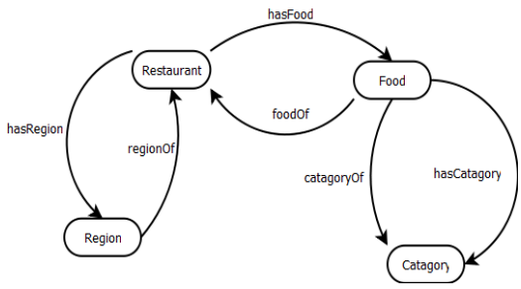
semantic query di seluruh Data RDF. SPARQL juga telah digunakan secara luas di antara RDF *query* engine dan *graph store*, selain itu SPARQL juga menyediakan ekspresivitas yang cukup baik untuk *query* dan *general computation of data*. Untuk memfasilitasi penyimpanan dan pemeliharaan, *query* SPARQL disimpan dalam RDF *triples*, dengan menggunakan SPIN SPARQL *Syntax*.



Gambar 1. Arsitektur Sistem

Arsitektur Sistem yang sesuai untuk pencarian berbasis semantic dengan memanfaatkan SPIN ditunjukkan oleh Gambar 1. Komponen-komponen dalam arsitektur tersebut, yaitu : 1) *User,* pengguna yang memberi masukan ke aplikasi pencarian berupa kalimat pencarian, isian dari pengguna tersebut kemudian diterima oleh *Input Query Processor.2) Input Query Processor* yang 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* yang 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,* yang 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 Restaurants.owl* dan *Words.owl, Restaurants.owl* adalah bagian yang menyimpan basis pengetahuan mengenai data restoran dalam bentuk *Triple*, sedangkan *Words.owl* adalah ontologi bantuan yang digunakan untuk menyimpan data *keyword, stopword* dan *directionword.* 6) Hasil *Inference,* Bagian ini berisikan hasil *inference* yang mengacu pada data *triple* yang terdapat di dalam *ontology Restaurants.owl.* 7) SPARQL Inferencing Notation (SPIN) yang 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*

Perancangan ontologi restoran dilakukan dengan mengikuti langkah-langkah seperti disarankan oleh Noy, dkk. (2001), 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 Yogyakarta. Sehubungan dengan kebutuhan pencarian berbasis semantik yang dilengkapi dengan *inference,* maka digunakan ontologi yang sesuai untuk kebutuhan tersebut. Ontologi restoran memiliki empat kelas yaitu : *Restaurant* (informasi utama mengenai restoran seperti nama maupun alamat restoran)*, Food* (informasi utama mengenai nama-nama makanan yang umumnya dijual di restoran)*, Region* (informasi mengenai area di daerah sebagai penunjuk lokasi restoran) dan *Category* (informasi mengenai kategori atau jenis makanan, seperti misalnya makanan jepang, makanan cina, makanan tradisional, *sea food*, dan lain-lain)*.* Diagram Relasi antar kelas (*class diagram*) digunakan untuk menunjukkan interaksi antar kelas dalam sistem (Berners Lee, 2001). Relasi akan mengijinkan suatu kelas mengetahui atribut, relasi dan hubungan yang ada pada kelas lainnya. Relasi antar kelas pada ontologi *Restaurant* ditunjukkan oleh Gambar 2.



Gambar 2. Diagram Relasi Antar Kelas

Pada Gambar 2 terlihat bahwa kelas *Restaurant* memiliki relasi *hasFood* dengan kelas *Food* yang berarti sebuah restoran memiliki berbagai makanan. Sebaliknya kelas *Food* memiliki relasi *foodOf* dengan kelas *Restaurant* yang berarti sebuah makanan di kelas *Food* dimiliki satu atau lebih restoran. Kelas *Restaurant* juga memiliki relasi *hasRegion* dengan kelas *Region* yang berarti setiap restoran memiliki satu atau lebih *region*. Sebaliknya kelas *Region* memiliki relasi *regionOf* dengan kelas *Restaurant* yang berarti suatu *region* di kelas *Region* dimiliki satu atau lebih restoran. Kelas *Food* memiliki relasi *hasCatagory* dengan kelas *Category* yang berarti setiap makanan memiliki satu atau lebih jenis atau kategori makanan. Sebaliknya kelas *Catagory* memiliki relasi *catagoryOf* dengan kelas *Food* yang berarti suatu kategori makanan di kelas *Catagory* dimiliki satu atau lebih makanan.

Pengembangan sistem pencarian ini menggunakan NLP untuk memahami maksud pertanyaan yang ditulis oleh pengguna, untuk memudahkan proses tersebut dibutuhkan ontologi bantuan yang terdiri dari tiga buah kelas, yaitu : 1) *Keyword,* digunakan sebagai standar untuk memberntuk sebuah kalimat. Keyword yang disimpan merupakan representasi dari *Data Type Propertis* (DTP), *Object Data Properties* (OBP) dan *Class* (CLP). Masing-masing keyword memiliki sinonim atau kata lain yang mempunyai makna yang sama dengan keyword yang telah didefinisikan. Pada konsep pencarian data restoran, sangat dimungkinkan munculnya kata kunci dari pertanyaan pengguna yang bersifat ambigu. Sebagai contoh adalah kata “jual” yang merupakan kata dasar hasil *stemming* dari kata “menjual” dan “dijual”. Sebagai contoh akan diberikan dua buah kalimat, yang pertama : “Tampilkan restoran yang menjual bakso”, yang kedua : “Tampilkan makanan yang dijual SariJogja”. Kata “jual” pada kalimat pertama bermakna “menyediakan makanan dengan nama…” yang memiliki sinonim “*food_name*”, sedangkan kata “jual” pada kalimat kedua bermakna “disediakan oleh restoran bernama…” yang memiliki sinonim “*restaurant_name*”. Untuk mengantisipasi hal tersebut maka pada *keyword* diberikan juga properti tambahan “*forclass*” yang menunjukkan sinonim yang mana yang digunakan dengan mengacu pada kelas yang ditemukan sesuai pertanyaan pengguna. 2) *Stopword,* ini digunakan untuk menghapus kata yang tidak bermakna dalam kalimat. Pembuangan *stopword* adalah proses pembuangan *term* yang tidak memiliki arti atau tidak relevan. *Term* diperolah dari daftar *stopword*, apabila kata masuk dalam daftar *stopword* maka kata tersebut tidak akan diproses lebih lanjut. Contoh stopword : yang, asal, punya, buat,dengan. 3) *Directionword* digunakan untuk menyimpan daftar kata perintah yang diijinkan di dalam kalimat pencarian dari pengguna. Kata perintah yang dipilih digunakan untuk membatasi pola kalimat pencarian yang diisikan pengguna sehingga akan lebih memudahkan dalam proses pemaknaan kalimat tersebut.

Menurut Strzkaowski (1999), pengolahan NLP sebagai bagian dari proses *Information Retrieval* terdiri dari beberapa tahap, yaitu : validasi kalimat, *tokenization*, *stemming* dan *filtering*.

Proses validasi kalimat adalah proses untuk menentukan apakah kalimat pencarian yang dimasukkan sesuai dengan aturan produksi (Mandala, 1999). Valid tidaknya suatu kalimat akan dilihat dari struktur pembentuk kalimat perintah yang diinputkan pada kotak pencarian sistem.

Pola keteraturan aturan produksi diidentifikasi mengandung lima unsur yaitu a) Kp, merupakan kata perintah, yang mengawali kalimat masukan. Kata-

kata ini berperan untuk membentuk suatu kalimat sesuai kaidah bahasa indonesia yang baku. Kata Perintah yang digunakan misal cari, carikan, tampilkan. b) Ctg, merupakan *object catagory*, mewakili obyek yang dicari (restoran, makanan,lokasi, jenis). c) Dtp , yang merupakan *datatype property*,mewakili atribut dari kelas (nama restoran, alamat, dan lain-lain). d) Obp, merupakan *object property*, mewakili relasi antar kelas (menjual, berlokasi). e) Value, merupakan nilai dari datatype property atau object property

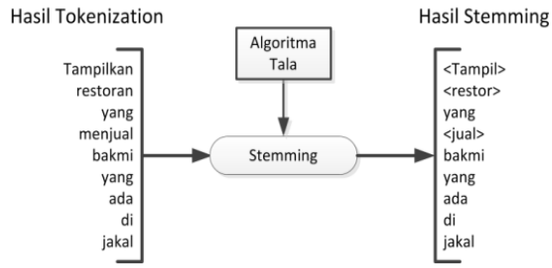
Sesuai kelima unsur yang ada dibentuk kalimat pencarian, namun kalimat yang diterima sebagai kunci pencarian haruslah kalimat yang valid, kalimat yang dianggap invalid akan ditolak oleh sistem dan harus diperbaiki sesuai dengan aturan produksi yang telah ditetapkan.

Berdasarkan unsur-unsur pembentuk di atas, ditetapkan pola pembentukan kalimat dengan identifikasi: kalimat minimal harus terdiri dari kata perintah (cari, carikan, sebutkan dan tampilkan), kategori / *class* pencarian (restoran, makanan, lokasi dan jenis) dan nilai / *value* yang akan dicari. identifikasi berikutnya adalah kalimat harus diawali dengan kata perintah kemudian diikuti dengan sinonim dari *class* yang dicari.

Gambar 3. Ilustrasi Proses Tokenisasi

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. Gambar 3 menunjukkan ilustrasi tokenisasi pada kalimat "Tampilkan restoran yang menjual bakmi yang ada di jakal"

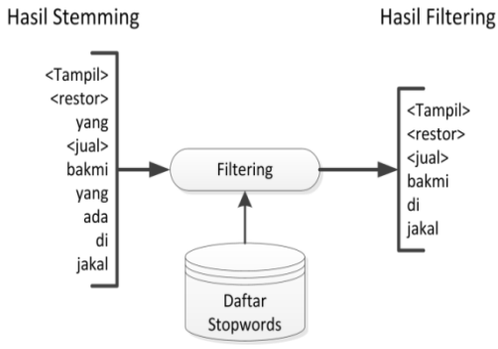
Proses *stemming* adalah proses pembentukan kata dasar. Setiap Kata yang diperoleh dari tahap *tokenization* dikenai proses *stemming*. Algoritma *stemming* yang digunakan dalam penelitian ini menggunakan algoritma berbasis aturan dari Tala (2003). *Stemming* bermanfaat mereduksi kata untuk menghindari ketidakcocokan dikarenakan adanya kata-kata yang berbeda namun memiliki makna dasar yang sama, kata-kata tersebut direduksi menjadi satu bentuk. Gambar 4 menunjukkan contoh ilustrasi *stemming* pada kata-kata hasil tokenisasi.



Gambar 4. Ilustrasi Proses *Stemming*

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. Sebagai contoh untuk proses penghilangan stopword dapat dilihat pada Gambar 5. Pada Gambar 5 tersebut terlihat bahwa kata "yang" dan "ada" termasuk dalam daftar *stopwords* sehingga perlu dihilangkan.

Proses representasi kalimat adalah proses untuk mencari makna atau maksud dari kalimat pencarian yang diisikan oleh pengguna.



Gambar 5. Ilustrasi Proses *Filtering*

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 tiga langkah, langkah pertama adalah melakukan pengecekan kata perintah. Kata pertama dalam kalimat pencarian harus berupa kata perintah yang terdapat pada daftar *Directionword* dari ontologi bantuan. Jika kata tersebut tidak ditemukan di daftar *Directionword* maka kalimat dianggap tidak valid.

Apabila 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 tiga kemungkinan yaitu : *restaurant, food* dan *catagory*. *Class region* yang ditemukan diartikan bahwa pengguna ingin menanyakan lokasi suatu restoran, informasi mengenai lokasi ini juga diberikan apabila pengguna membuat pertanyaan dengan inti kalimat pencarian berisi *class restaurant*, sehingga untuk *class region* akan dianggap sama dengan *class restaurant*. Berdasarkan hal tersebut maka kemudian dapat dicari padanan dari tiap kata mulai kata yang ketiga (kata pertama adalah kata perintah, 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 : KP – CLASS – PROPERTY – VALUE – PROPERTY – 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*.

Tampil	→	kata perintah
Restor	→	class (restaurant)
Jual	→	property (foodName)
Bakmi	→	value
~~Di~~	→	~~property (regionName)~~ → diabaikan karena tidak dikuti value
Sekitar	→	property (regionName)
Jakal	→	value

Gambar 6. Hasil Representasi Kalimat

Sebagai contoh kalimat pencarian : “Tampilkan restoran yang menjual bakmi yang ada di sekitar jakal”, setelah proses *filtering* akan menghasilkan bentuk : “tampil restor jual bakmi di sekitar jakal” dengan hasil representasi ditunjukkan Gambar 6.

SPARQL Filter dibuat berdasar pola yang ditemukan pada representasi kalimat sebelumnya dan dibentuk dengan format *property-value*. Sebagai contoh hasil representasi yang ditunjukkan Gambar 6 akan menghasilkan format seperti terlihat di Gambar 7.

Hasil Representasi	Jual	Bakmi	Sekitar	Jakal
Pattern	Prop	Value	Prop	Value
Sentence	foodName	Bakmi	regionName	Jakal

Gambar 7. Format Hasil Representasi

Berdasar *Pattern* dan *Sentence* yang pada Gambar 7 akan dapat dibentuk SPARQL Filter yang dapat dilihat di Gambar 8.

```
FILTER((regex(?foodName,'Bakmi','i') &&
            regex(?regionName,'jakal','i'))
```

Gambar 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 9a.

```
//sparql construct untuk food_catagory
CONSTRUCT {
    ?s :foodCatagory ?fc .
}
WHERE {
    ?s :hasFood ?f . ?f :hasCatagory ?c .?c :catagoryName ?fc .
}
//sparql construct untuk price_catagory
CONSTRUCT {
    ?sbj :priceCatagory ?pc .
}
WHERE {
    ?sbj :lowestPrice ?h .
    BIND (IF((?h < 20000), "MURAH", "MAHAL") AS ?pc) .
}
```

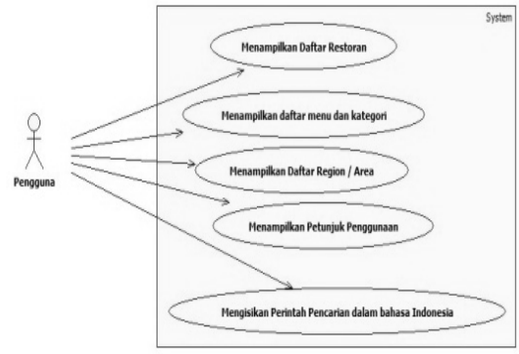
(a)

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

(b)

Gambar 9. 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 scriptnya dapat dilihat pada Gambar 9b. Hal yang sama juga dilakukan untuk properti *hasCatagory* → *catagoryOf,* dan properti *hasRegion* → *regionOf*.

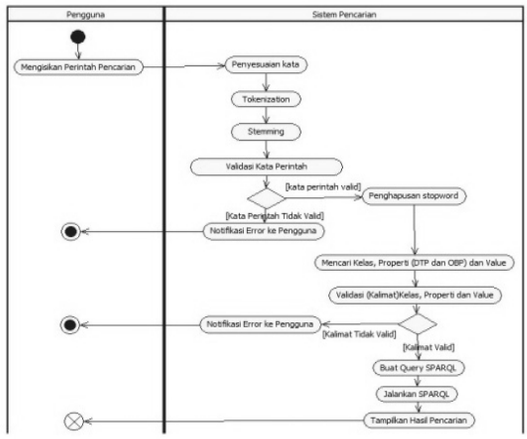


Gambar 10. *Use Case Diagram*

Use Case digunakan untuk mendiskripsikan proses yang terjadi dalam sistem dan elemen-elemennya serta hubungannya dengan *actor*. *Use Case* dari aplikasi pencarian data restoran ditunjukkan oleh Gambar 10. Berdasarkan tersebut dapat dijelaskan bahwa pengguna dapat memasukkan kata atau kalimat pencarian dengan menggunakan bahasa Indonesia. Sebagai tambahan fasilitas aplikasi yang dibangun menyediakan menu yang langsung dapat diakses yaitu Menampilkan Daftar Restoran, Menampilkan Daftar Menu dan Kategori, Menampilkan Daftar Region/Area dan Menampilkan Petunjuk Penggunaan.

Activity Diagram disini digunakan untuk memodelkan aliran kerja (*workflow*) dari proses pencarian data restoran berbasis semantik. Dalam diagram tersebut digambarkan urutan aktifitas dan proses bisnis untuk *use case* pada fungsi "mengisikan perintah pencarian" yang dilakukan oleh pengguna. Model pencarian

direpresentasikan dalam *activity diagram* Gambar 11.

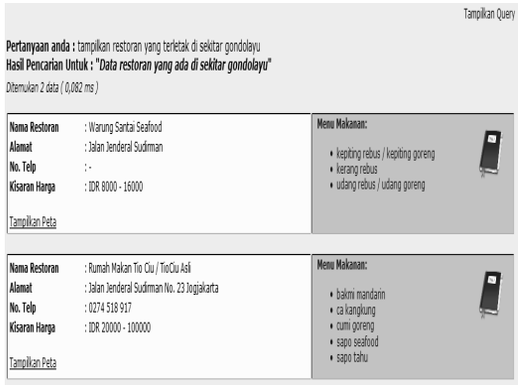


Gambar 11. *Activity Diagram*

PEMBAHASAN

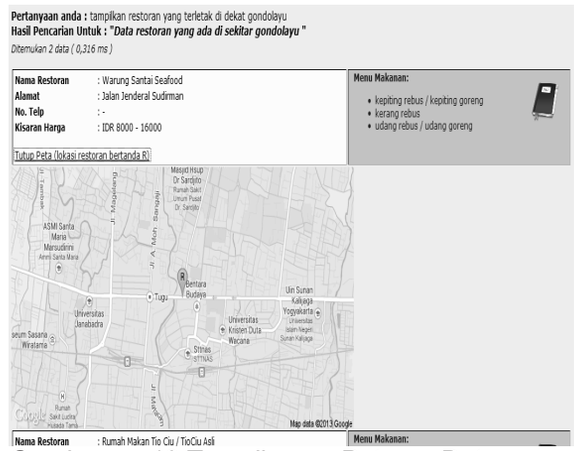
Kalimat pencarian yang diisikan ke dalam kotak pengisian berupa kalimat perintah yang menggunakan pola yang sudah diatur sehingga kalimat pencarian tersebut lebih mudah diolah oleh sistem untuk menemukan data yang dinginkan.

Contoh hasil pencarian dengan kalimat pencarian : "Tampilkan restoran yang terletak di dekat gondolayu", ditunjukkan oleh Gambar 12.



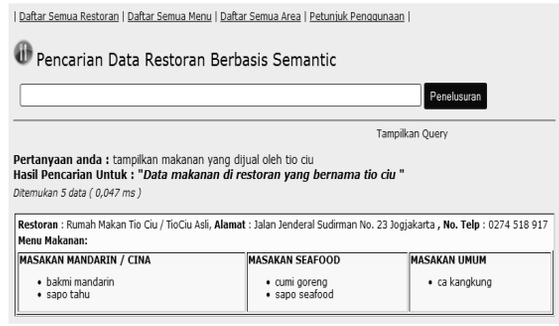
Gambar 12.Tampilan Pencarian Data Restoran

Jika pengguna pelakukan klik pada *link* "Tampilkan Peta" seperti terlihat pada Gambar 12, maka akan dimunculkan peta yang mengacu pada *Google Map* di bagian bawah data dengan lokasi restoran diberi *mark* dengan tulisan **R**, dan otomatis link "Tampilkan Peta" berubah menjadi Tutup Peta (lokasi restoran bertanda R), seperti yang ditunjukkan oleh Gambar 13.



Gambar 13.Tampilan Peta Data Restoran

Untuk contoh hasil pencarian dengan kalimat pencarian : "Tampilkan makanan yang dijual oleh Tio Ciu", ditunjukkan oleh Gambar 14.



Gambar 14. Tampilan Pencarian Data Makanan

Pada Gambar 14 terlihat bahwa tampilan data makanan dilengkapi dengan data restoran yang dibutuhkan, serta penulisan makanan dibagi per jenis makanan sehingga memudahkan pembacaan informasi.

Pengujian Sistem yang pertama menggunakan pengujian dengan analisis linguistik. Pengujian dilakukan dengan memasukkan kalimat perintah pencarian menggunakan berbagai pola kalimat. Ini untuk melihat sejauh mana kemampuan aplikasi dalam memahami perintah pencarian, mengenali kata kunci, melakukan validasi dan membuat *query statement* yang tepat sehingga akhirnya

menghasilkan informasi yang tepat sesuai perintah pencarian yang diinginkan baik kalimat sederhana maupun kalimat kompleks.

Tabel 1. Hasil Pengujian Analisis Linguistik *Single Value*

Kalimat Pencarian	Pemaknaan Oleh Sistem	Hasil
Tampilkan Restoran yang ada di seputar amplaz	Data restoran yang ada di sekitar amplaz	OK
Tampilkan Restoran yang menjual Bakmi	Data restoran yang makanannya bernama bakmi	OK
Tampilkan Restoran yang menjual makanan jepang	Data restoran dengan jenis makanan bernama jepang	OK
Tampilkan Restoran Jepang	Data restoran dengan kategori makanannya bernama jepang	OK
Tampilkan Restoran Makanan Jepang	Data restoran dengan jenis makanan bernama jepang	OK
Tampilkan Restoran bernama tio ciu	Data restoran dengan nama restoran adalah tio ciu	OK
Tampilkan Lokasi Restoran yang menjual makanan Jepang	Data restoran dengan jenis makanan bernama jepang	OK
Tampilkan Makanan yang ada di Jambon Resto	Data makanan di restoran yang bernama jambon resto	OK
Tampilkan Restoran yang bertarif mahasiswa	Data restoran yang memiliki tarif murah	OK

Pengujian kalimat sederhana seperti ditunjukkan oleh Tabel 1, dilakukan dengan mengisikan kalimat pencarian *single value*. Contoh: "Tampilkan rumah makan yang menjual udang". Aplikasi akan mencari restoran yang nama makanannya bernama udang, yang berarti terdapat satu properti "*foodName"* dan satu *value* "udang".

Tabel 2. Hasil Pengujian Analisis Linguistik *Single Value*

Kalimat Pencarian	Pemaknaan Oleh Sistem	Hasil
Tampilkan Restoran jepang yang ada di Jakal	Data restoran dengan kategori makanannya bernama jepang dan yang ada di sekitar jakal	OK
Tampilkan restoran bernama phuket yang ada di seputar taman siswa	Data restoran yang ada di sekitar taman siswa dan dengan nama restoran adalah phuket	OK
tampilkan makanan cina yang dijual oleh tio ciu	Data makanan dengan kategori makanannya bernama cina dan di restoran yang bernama tio ciu	OK
Tampilkan menu yang terbuat dari seafood di pelem golek	Data makanan yang kategori makanannya bernama seafood dan di restoran yang bernama pelem golek	OK
tampilkan warung berharga pelajar di daerah palagan	Data restoran yang memiliki tarif murah dan yang ada di sekitar palagan	OK
tampilkan restoran cina dan jepang yang terletak di perempatan sgm	Data restoran dengan kategori makanannya bernama cina atau jepang dan yang ada di sekitar sgm	OK

Pencarian kompleks juga diujikan untuk melihat kemampuan aplikasi dalam menangani kalimat pencarian yang memiliki kata kunci lebih dari satu baik properti maupun *value*. Contoh : "Tampilkan restoran yang menjual steak yang ada di sekitar amplaz", akan membutuhkan properti "*foodName*" dengan *value* "steak" dan properti "*regionName*" dengan *value* "amplaz". Hasil pengujiannya ditunjukkan oleh Tabel 2.

Kalimat Pencarian	A	B	C	R	P
tampilkan restoran yang terletak di sekitar tamsis	5	0	0	1	1
tampilkan makanan yang dijual pelem golek	10	0	0	1	1
tampilkan restoran jepang	3	0	0	1	1
tampilkan menu seafood yang ada di tio ciu	7	0	0	1	1
tampilkan restoran yang terletak di gondolayu dan menjual udang	2	0	0	1	1
tampilkan restoran yang menjual udang	5	0	0	1	1
tampilkan masakan seafood dan pasta	17	0	0	1	1

Keterangan :

A : *Retrieved Relevant*
B : *Not Retrieved Relevant*
C : *Retrieved Not Relevant*
R : *Recall* (A / (A + B))
P : *Precision* (A / (A + C))

Gambar 15. Hasil Pengujian *Recall* and *Precision*

Pengujian yang kedua adalah pengujian efektifitas dan efisiensi *information retrieval*. Sesuai yang disarankan oleh Grossman dan Frieder (1998), maka pengujian efektifitas dan efisiensi dilakukan dengan melihat nilai dari uji *Recall* dan *Precision*. Mengacu pada rumus perhitungan yang dibuat oleh Hasugian (2006), hasil pengujiannya ditunjukkan pada Gambar 15.

Berdasarkan hasil yang ditunjukkan di Gambar 15, terlihat bahwa nilai untuk *recall* dan *precision* dari aplikasi yang dikembangkan memiliki nilai 1:1 untuk berbagai variasi kalimat pencarian, maka sesuai yang disampaikan oleh Lee dalam Hasugian (2006) dapat dikatakan bahwa aplikasi pencarian data restoran berbasis semantik ini memiliki efektifitas dan efisiensi hasil pencarian yang tinggi.

KESIMPULAN

Pemanfaatan SPARQL Inferencing Notation (SPIN) dalam prototype aplikasi pencarian data restoran berbasis semantik menghasilkan rasio 1:1 untuk uji *recall and precision* yang berarti aplikasi ini memiliki efektifitas dan efisiensi yang tinggi, selain itu aplikasi juga mampu memahami dan memberikan hasil yang relevan untuk berbagai variasi kalimat pencarian termasuk adanya *multi value,* properti yang kembar serta penggunaan kata hubung (dan, atau).

DAFTAR PUSTAKA

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

Berners-Lee, T. dan Fischetti, M. , 1999, *Weaving The Web*, Harper San Francisco, USA

Berners-Lee, T., Hendler, J., dan Lassila, O., 2001, *The Semantis Web.* American Scientific, USA

Dameron, 0., Besana,P., Zekri, O., Bourdé, A., Burgun, A., Cuggia, M., 2013, OWL model of clinical trial eligibility criteria compatible with partially-known information, Journal of Biomedical Semantics 2013, *http://www.jbiomedsem.com/content/4/1/17*

DuCharme, B, *Learning SPARQL*, S. St.Laurent, Ed. O'Reilly Media, 2011

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.

Grossman, D.A., dan Frieder, O., 1998, *Information Retrieval : Algorithms and Heuristic*, Springer, New York

Hasugian, J., 2006, Penggunaan Kosa Kata Terkendali dalam Sistem Temu Balik Informasi Berbasis Teks, *PUSTAHA:* Jurnal Studi Perpustakaan dan Informasi, Vol. 2, No. 2

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.

Knublauch,H., 2009, *SPIN SPARQL Inferencing Notation*, www.spinrdf.org, diakses tanggal 10 November 2011

Knublauch,H., Hendler, J.,A, Idehen, K., 2011, SPIN - Overview and Motivation, *http://www.w3.org/Submission/spin-overview/*, diakses tanggal 29 November 2012

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

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

Mamadolimova,A., Ambiah, N., Lukose, D., 2011, *Modeling Islamic Finance Knowledge for Contract Compliance in Islamic Banking*, Knowledge-Based and Intelligent Information and Engineering Systems Lecture Notes in Computer Science Volume 6883, 2011, pp 346-355 , Springer, NewYork

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

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

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,* diakses 5 Maret 2012

Su, C., Chen Y., Chih,C. , 2013, Personalized Ubiquitous Diet Plan Service Based on Ontology and Web Services*, International Journal of Information and Education Technology*, Vol. 3, No. 5, October 2013

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

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

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.