

Analisis Penerapan Pemeringkatan ReconRank Query Singleterm untuk Query Multiterm Web Semantik

Urip T. Setijohatmo^a, Jonner Hutahaean^a, Setiadi Rachmat^a, Alaika Mustikaati^b

^a Dosen Jurusan Teknik Komputer Polban

^b Alumni Jurusan Teknik Komputer Polban 2015

Abstrak

Pemeringkatan merupakan bagian penting dari suatu mesin pencari karena mewakili keakuratan hasil pencarian. Seringkali hasil pencarian tidak sesuai dengan apa yang user kehendaki selain karena keterbatasan alamiah yang melekat pada mesin pencari konvensional, ia juga dapat dimanipulasi (dan telah terjadi) untuk kepentingan tertentu. Salah satu cara mengatasinya adalah dengan membentuk teknologi/model web semantik. Dengan demikian pemeringkatan juga merupakan bagian dari web semantik. Penelitian ini melakukan perluasan algoritme pemeringkatan ReconRank single query term untuk diterapkan ke multi term web semantik. Pertama kali terms query diproses menggunakan metode *unweighted shorted path* untuk menghasilkan *sentences* dengan rangkaian *edge path* terpendek. Adalah mungkin untuk menghasilkan jawaban dengan panjang path sama, sehingga bila ini terjadi pemeringkatan lebih jauh perlu dilakukan. Di sinilah peran pemeringkatan ReconRank diperlukan. Studi kasus untuk penelitian ini menggunakan Ontology Pariwisata. Hasil eksperimen menunjukkan bahwa ReconRank dapat diterapkan untuk query multiterm dengan keakuratan diatas 50%.

Kata Kunci: *ReconRank, Ontology, Search engine, web semantic*

1. Pendahuluan

Informasi yang tersedia di internet sekarang ini sangat banyak sehingga untuk mendapatkan suatu informasi dibutuhkan alat untuk mencari informasi tersebut dengan cepat. *Search engine* atau mesin pencari menyediakan fasilitas untuk pencarian informasi yang dibutuhkan dari World Wide Web (WWW) yang berisi berbagai halaman HTML dan link lainnya. *Search engine* ini dapat menampilkan informasi yang relevan sesuai dengan *query* berupa *keyword* yang diinputkan oleh *user*.

1.1 Latar Belakang

Mesinpencari seperti Google, Yahoo, dan Bing merupakan mesin pencari web konvensional yang sekarang ini telah menjadi jalan utama untuk mencari suatu informasi yang dibutuhkan. Walaupun mesin-mesin pencari ini sanggup memberikan berbagai informasi yang dibutuhkan, seringkali ketepatan dalam mencari informasi tersebut dipertanyakan [4]. Salah satu cara untuk menangani permasalahan tersebut, Tim Berners-Lee, penemu World Wide Web, menghadirkan web semantik.

Web Semantik adalah pengembangan dari World Wide Web dimana makna semantik dari suatu informasi pada web didefinisikan. Web semantik meletakkan data pada web dalam suatu bentuk sehingga mesin secara alami dapat memahami atau mengubahnya menjadi format tertentu. Melalui web semantik inilah berbagai perangkat lunak akan mampu mencari, membagi, dan mengintegrasikan informasi dengan cara yang lebih mudah.

Suatu mesin pencari membutuhkan fungsi perankingan untuk memprioritaskan hasil pencarian yang relevan dengan *query*. Salah satu algoritma untuk mesin pencari web semantik adalah algoritma ReConRank. ReConRank dihidirkan oleh Hogan A dan rekan-rekannya [3] yang merupakan perkembangan dari algoritma PageRank (algoritma dasar mesin pencari web konvensional) untuk menentukan urutan *resources* pada file RDF dan *context*-nya dengan tujuan meningkatkan kualitas perankingan. Akan tetapi, algoritma ini hanya menangani *query single term* sehingga *user* harus membatasi melakukan *query* dengan satu *term* saja sedangkan pada kenyataannya *user* membutuhkan *query* yang *multiterm*. Penelitian ini dilakukan untuk melanjutkan penelitian sebelumnya [1] dengan tujuan merealisasikan lebih detail cara memperluas algoritma ReConRank agar dapat menangani *query multi term*.

Karena *query* yang akan digunakan pada penelitian ini adalah *multiterm* dan topical subgraph yang merupakan input parameter dari algoritma ReConRank berupa *graph*, maka untuk menentukan tingkat kerelevanan *query* dengan hasil pencarian, penulis akan menggunakan *shortest path* (lintasan terpendek) pada topical subgraph. Zhou dan rekannya juga menerapkan penghitungan *shortest path* pada penelitiannya [5]. Mereka menggunakan algoritma Minimum Spanning Tree pada suatu ontologi.

Pada penghitungan jarak terpendek dikarenakan *edge* pada *graph* algoritma ReConRank itu berupa *predicate* dan bukan merupakan bobot angka, maka *shortest path* yang dibutuhkan untuk mengurutkan tingkat kerelevanan *query* adalah *unweighted shortest path* (lintasan terpendek tidak berbobot). Algoritma Breadth-First Search merupakan salah satu algoritma untuk menentukan *shortest path* pada suatu *graph* yang tidak berbobot. *Unweighted shortest path* didapatkan dengan cara membandingkan jumlah node antar *term query* pada *graph*. *Shortest path* inilah yang akan diimplementasikan pada algoritma ReConRank.

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah dikemukakan, maka masalah yang muncul adalah bagaimana memperluas algoritma ReConRank agar dapat digunakan untuk *query multi term* sehingga *user* mendapatkan jawaban yang lebih relevan.

1.3 Batasan

Pada permasalahan ini, pembuatan aplikasi maupun penelitiannya dibatasi oleh parameter-parameter berikut ini:

1. *Keyword* merupakan objek pada suatu *quads* yang termasuk *object literal*.
2. Yang dimaksud dengan *multi term* adalah bukan yang merupakan *idiom* (gabungan kata yang membentuk arti baru), *slang* (kosakata tidak formal), ataupun *proverb* (peribahasa).

Contoh:

- *Idiom* : *as easy as pie* = sangat mudah
 - *Slang*: *the kids* = *children*; *kick the bucket* = *die*
 - *Proverb* : *A fruitless life is useless life* = Hidup tidak berarti tanpa berbuat sesuatu yang bermanfaat
3. Penelitian yang dilakukan hanya berfokus pada salah satu bagian dari arsitektur *Semantic Web Search Engine* yaitu *Semantic Search and Query Engine*.

1.3.1 Research Question

Bagaimana algoritma ReConRank dapat diperluas dari sistem *query single term* menjadi *query multiterm* dengan kepresisian yang lebih baik

1.3.2 Hipotesa

Algoritma ReConRank dapat diperluas dari *query single term* menjadi *query multiterm* menggunakan *shortestpath* yang menghubungkan setiap *term*-nya dengan nilai kepresisian diatas 50%.

1.4 Tujuan

Penelitian ini bertujuan untuk memperluas algoritma ReConRank yang hanya dapat menangani single term agar *user* dapat melakukan peng-*query*-an dengan *multiterm* untuk diterapkan pada aplikasi perankingan hasil pencarian web semantik.

2. Studi Pustaka

2.1. Referensi Konsep

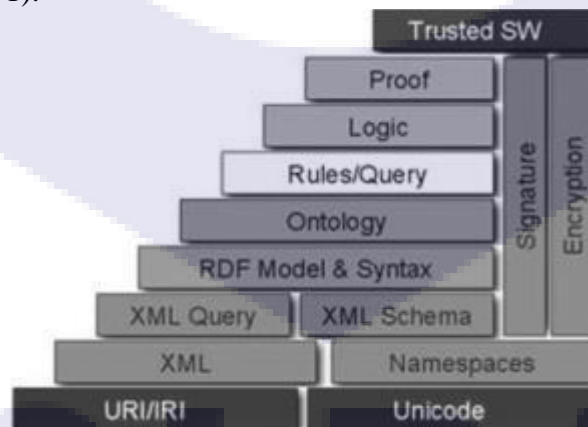
2.1.1. Semantic Web

Semantic web (web 3.0) merupakan perluasan dari web yang sebelumnya (web 2.0), dimana informasi yang diberikan memiliki arti yang jelas, lebih memungkinkan adanya kerjasama antara komputer dengan manusia dan komputer dengan komputer. Dalam waktu yang dekat, perkembangan ini akan mengantarkan fungsi baru yang signifikan sebagai mesin menjadi jauh lebih baik dalam memproses dan "mengerti" data yang mereka tampilkan [6].

Berikut ini adalah komponen semantik menurut Antoniou, G., et al [4]:

a. Komponen Semantic Web

Pembuatan *Semantic Web* dimungkinkan dengan adanya sekumpulan standar yang dikoordinasi oleh *world wide web consortium* (W3C). Standar yang paling penting dalam membangun *Semantic Web* adalah XML, XML Schema, RDF, OWL, dan SPARQL (Gambar 1).



Gambar 1 Layer Semantic Web [4]

Untuk lebih terperinci komponen Semantic web dapat dibaca pada [4]. Komponen yang dieksplorasi pada penelitian ini adalah Ontology dan Rules/Query dalam OWL dan SPARQL

Web Ontology Language (OWL)

Web ontology language (OWL) adalah suatu bahasa yang dapat digunakan oleh aplikasi-aplikasi yang bukan sekedar menampilkan informasi tersebut pada manusia, melainkan juga yang perlu memproses isi informasi isi. Ontologi sendiri dapat

didefinisikan sebagai suatu cara untuk mendeskripsikan arti dan relasi dari istilah-istilah. Deskripsi tersebut berisi *classes*, *properties*, dan *instances*.

Deskripsi ini dapat membantu sistem komputer dalam menggunakan istilah-istilah tersebut dengan cara yang lebih mudah. Dengan menggunakan OWL, kita dapat menambah *vocabulary* tambahan disamping semantiks formal yang telah dibuat sebelumnya menggunakan XML, RDF, dan RDF Schema. Hal ini sangat membantu penginterpretasian mesin yang lebih baik terhadap isi Web. Untuk mendeskripsikan *properties* dan *classes*, OWL menambahkan *vocabulary* seperti:

- “among others”
- Relasi antar *classes* (misalnya: “disjointness”)
- Kardinalitas (misalnya: “exactly one”)
- Kesamaan (*equality*)
- Karakteristik *property* (misalnya: “symmetry”)
- *Enumerated classes*

SPARQL

SPARQL protocol and RDF query language (SPARQL) adalah sebuah *protocol* dan bahasa *query* untuk *Semantic Web's resources*. Sebuah *query* yang menggunakan SPARQL dapat terdiri atas *triple patterns*, konjungsi (*or*), dan disjungsi (*and*). Untuk menjalankan SPARQL dapat menggunakan beberapa *tools* dan APIs seperti: ARQ, Rasqal, RDF::Query, twingql, Pellet, dan KAON2. Tools tersebut memiliki API yang memungkinkan pemrogram untuk memanipulasi hasil query dengan berbagai aplikasi yang ada.

Berikut adalah struktur dari SPARQL *query* :

- a. Deklarasi *prefix* (PREFIX) untuk menyingkat URI
- b. Definisi *dataset* (FROM ...), *graph* yang akan dikenai *query*
- c. *Result clause* (SELECT ...), identifikasi informasi yang harus dikembalikan
- d. *Query Pattern* (WHERE ...)
- e. *Query Modifiers* (ORDER BY...)

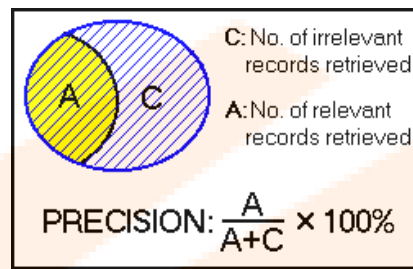
Berikut adalah contoh SPARQL sederhana untuk mengambil seluruh nilai p dan o:

```
SELECT ?p ?o
WHERE { ?s ?p ?o }
```

Pada penelitian ini, penulis akan menggunakan RDF sebagai format metadata (yang terdiri dari *context*, *subject*, *predicate*, dan *object*), OWL sebagai bahasa ontologi yang akan digunakan sebagai datanya, dan SPARQL akan digunakan untuk membuat suatu query pada database Virtuoso (Subbab 2.3.1).

2.1.2. Precision

Precision adalah rasio angka dari link relevan yang terambil untuk jumlah angka yang tidak relevan maupun relevan terhadap link yang terambil. Biasanya ditampilkan dalam persentase (Jizba, 2000).



Gambar 2 Precision

Presisi adalah probabilitas jumlah link yang relevan dari semua *link* yang diambil dibandingkan dengan jumlah keseluruhan link yang diambil (Desrianti, 2011). Secara matematis, presisi dapat dituliskan sebagai:

$$precision = \frac{|\{relevant\ documents\} \cap \{retrieved\ documents\}|}{|\{retrieved\ documents\}|}$$

Penghitungan presisi pernah dilakukan terhadap mesin pencari Google dan Yahoo oleh B.T. Sampath Kumar dan J.N. Prakash [8] untuk membandingkan mana mesin pencari yang lebih besar kepresisiannya. Penelitian tersebut tidak hanya menghitung kepresisian, namun juga recall. Penghitungan dilakukan dengan menentukan mana *site* atau *link* yang relevan dengan *query*. Cara menghitung kepresisiannya adalah dengan cara sebagai berikut.

$$precision = \frac{\text{Sum of the scores of sites retrieved by a search engine}}{\text{Total number of sites selected for evaluation}}$$

Hasil pencarian yang relevan pada Google dan Yahoo dikategorikan menjadi ‘lebih relevan’, ‘kurang relevan’, ‘tidak relevan’, ‘link’ dan ‘situs tidak dapat diakses’ atas dasar dari kriteria berikut:

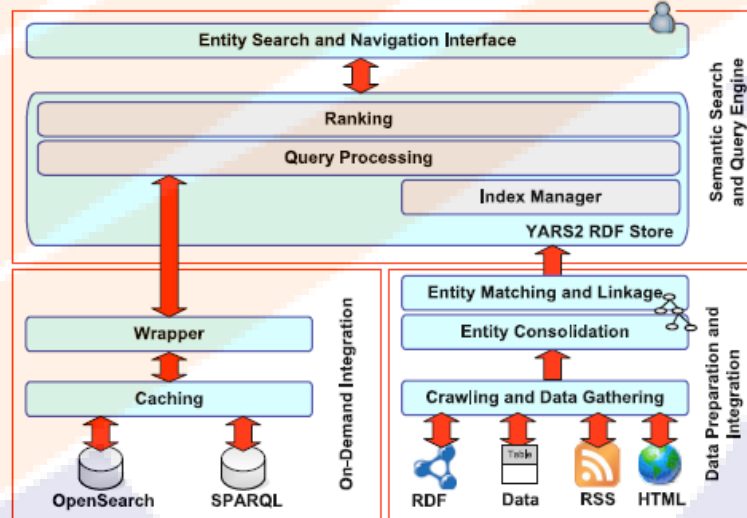
- Jika halaman web erat kaitannya dengan *query* pencarian, maka dikategorikan sebagai ‘lebih relevan’ dan diberi skor 2
- Jika halaman web tidak terkait dengan *query* tetapi terdiri dari beberapa konsep yang relevan dengan *query*, maka dikategorikan ‘kurang relevan’ dan diberi skor 1
- Jika halaman web tidak berhubungan dengan *query*, maka dikategorikan sebagai ‘tidak relevan’ dan diberi skor 0
- Jika sebuah halaman web terdiri dari serangkaian *link*, bukan informasi yang dibutuhkan, maka dikategorikan sebagai ‘link’ dan diberikan skor 0,5 jika salah satu atau dua link terbukti berguna.
- Jika muncul pesan ‘situs tidak dapat diakses’ untuk URL tertentu, halaman tersebut diperiksa lagi nantinya. Jika pesan terjadi berulang kali pada halaman itu, maka dikategorikan sebagai ‘situs tidak dapat diakses’ dan diberi skor 0.

Precision akan digunakan pada penelitian ini sebagai alat ukur dalam menentukan nilai keakuratan dari hasil pencarian menggunakan algoritma perankingan ReConRank yang diperluas dengan sistem *query* multi *term*. Namun, untuk penelitian ini akan menggunakan 2 kategori saja yaitu relevan dan tidak relevan. Kategori yang relevan akan diberi skor 1 dan yang tidak relevan diberi skor 0.

2.2. Artikel Ilmiah Penelitian Sejenis

2.2.1. Semantic Web Search Engine

Selain itu, Harth A , et al pada [7] menjelaskan bahwa SWSE (*Semantic Web Search Engine*) adalah suatu sistem yang mengumpulkan data dalam format terstruktur dari web dan bertindak sebagai repositori data web yang memungkinkan untuk peng-*query*-an ad-hoc dan analisis data yang kompleks. Arsitektur yang dapat diimplementasikan pada web semantik adalah arsitektur *semantic web search engine*. Arsitektur dari *semantic web search engine*(SWSE) merupakan hasil adaptasi dari arsitektur *search engine* dan database / *data warehousing*. Gambar 3 menunjukkan arsitektur SWSE dan data *flow* yang ada di dalam sistemnya.



Gambar 3 Arsitektur *Semantic Web Search Engine*

Inti dari SWSE adalah *Semantic Search and Query Engine* (seperti yang digambarkan pada Gambar 3). YARS2 merupakan arsitektur terdistribusi terukur untuk *indexing* dan *querying* dataset RDF yang besar dan beroperasi pada model data *named graph*, dimana RDF *triple* (subject, predicate, object) ditambahkan dengan context sehingga membentuk *quadruple* (subject, predicate, object, context).

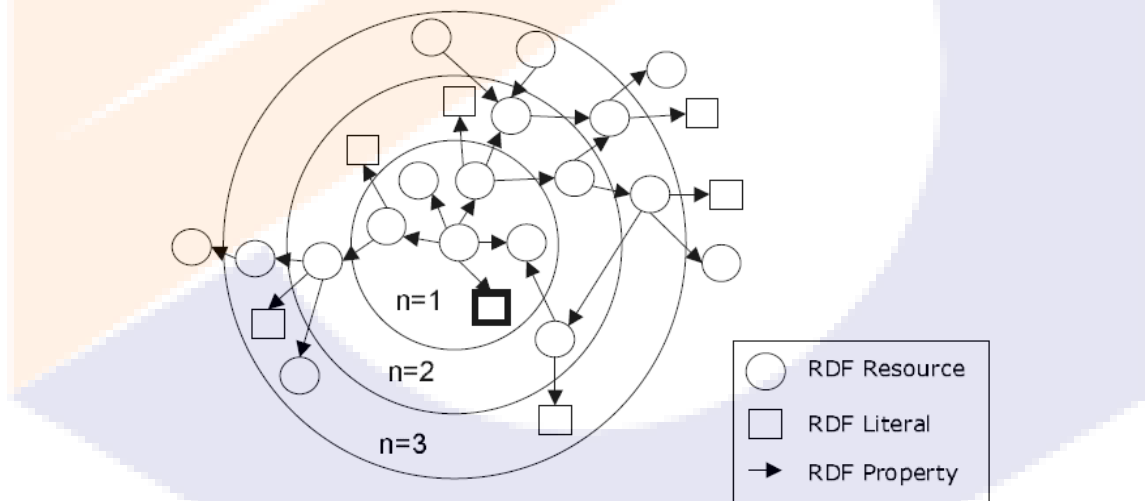
Indexing Manager menghasilkan dan melayani pencarian *local keyword* dan *quad* (*named graph*) *indices*. Query Processor berkoordinasi dengan beberapa Index Manager melalui suatu jaringan dan menawarkan SPARQL *end-point*. ReConRank digunakan untuk menentukan peringkat entitas dalam result-set yang menyediakan *metric* untuk kepentingan entitas tertentu dan juga kepercayaan terhadap sumber data; *metric* ini digunakan untuk memesan hasil presentasi di UI.

Dari ketiga arsitektur Semantic Web Search Engine tersebut, penelitian yang dilakukan lebih berfokus pada bagian Semantic Search and Query Engine, khususnya bagian Ranking dan Query Processing.

2.2.2. Metode Perankingan dengan Algoritma ReConRank

Algoritma ini merupakan algoritma yang telah dihadirkan oleh Hogan A. dan rekan-rekannya [3]. ReConRank adalah gabungan dari ResourceRank dan ContextRank yang merupakan pengembangan dari algoritma PageRank untuk menentukan urutan

resources dalam file RDF dengan *context*-nya untuk meningkatkan kualitas pengurutan [3]. Studi yang dilakukan Hogan A. dan rekan-rekannya menggunakan pendekatan yang berfokus pada topical subgraph. Mereka mengatakan bahwa topical subgraph merupakan *subgraph* yang merepresentasikan hasil *query term* yang sesuai dengan *keyword* yang dimasukkan user. Topical subgraph didapatkan dengan cara memilih *inlinks* dan *outlinks* yang dimiliki oleh *subject* dari *literal* yang cocok dengan *keyword* pencarian. Aspek penting yang memengaruhi hasil pencariannya adalah ukuran *subgraph* yang dipilih. Mereka menggunakan parameter n untuk menentukan berapa banyak *hop* di sekitar *literal* yang cocok yang harus dimasukkan ke dalam Topical Subgraph, yaitu dengan memasukkan node ke dalam *graph* yang dicapai dalam langkah n dari node *subject* literal yang cocok. Gambar 4 menunjukkan bagaimana cara memilih Topical Subgraph dengan sebuah pencocokan *literal* suatu *keyword* (bertanda tebal) dengan $n = 1, 2, 3$. Nilai n menentukan seberapa luas atau sempitnya hasil pencarian. Semakin besar nilai n , maka nilai *recall* naik dan presisi turun. Sedangkan jika nilai n semakin kecil, maka nilai *recall*-nya turun dan presisi-nya naik.

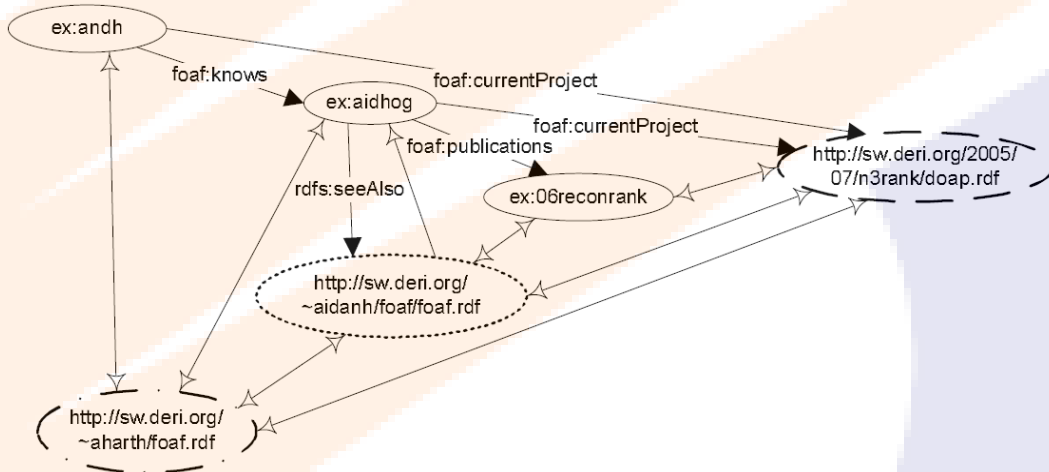


Gambar 4 Topical Subgraph mencocokkan *keyword* dengan $n = 1, 2, 3$.

Setelah menentukan Topical Subgraph, dibuat ResourcesRank. Mereka menyebut ResourcesRank sebagai *resources* relevan yang muncul sebagai *subject* setidaknya satu kali dalam suatu kumpulan data. Selanjutnya algoritma perankingan pada *context graph* yang disebut ContextRank diterapkan. Pada [3] menggunakan contoh Topical Subgraph untuk pencarian keyword “ReConRank” dengan $n = 1$ dengan contoh RDF *graph* dari *resources*, *edge* dan *context*-nya yang ditunjukkan pada Gambar 5.

POLBAN

3. Menurunkan *graph* dengan mengimplikasikan *link* dalam cara tertentu antara *context* dan *resources*(Gambar 8). Ada tiga jenis hubungan antara *contexts* dan *resources*, yaitu:
- Link* antara *context* dan *resource(s)* yang dikandungnya.
 - Link* antara *resources* dan *context* yang mengandungnya.
 - Link* dari *context* ke *context*. Contohnya *resource* di *context* A berhubungan dengan *context* B, maka secara tidak langsung *context* A berhubungan dengan *context* B.



Gambar 8 Struktur *link* dari *graph* untuk analisis ReConRank

4. Menghitung nilai ranking tiap node menggunakan algoritma penghitungan nilai ranking vektor. Proses penghitungan ranking vektor adalah sebagai berikut (notasi algoritma ditunjukkan pada Tabel 1):

- Inisialisasi.** Buat sebuah estimasi awal *eigenvector* R_0 dengan tiap node i diinisialisasi oleh i_i / m .
- Sebelum iterasi pertama dan selama iterasi.** Jumlah dari nilai ranking dari setiap *dead-link nodes*, $\sum_{j \in \text{dead}G} R_k(j)$, dan jumlah dari *outlinking nodes*, $\sum_{j \in \text{live}G} R_k(j)$, dihitung sebelum iterasi pertama dan selama iterasi tiap k . Nilai ini dikombinasikan dalam bentuk seperti berikut sebelum iterasi k untuk membuat nilai rank dasar untuk tiap node, \min_k .

$$\min_k = \frac{\sum_{j \in \text{dead}G} R_{k-1}(j)}{n} + \sum_{j \in \text{live}G} R_{k-1}(j) * \frac{1-d}{n}$$

Eigenvector yang lama disimpan untuk penghitungan iterasi selanjutnya.

- Selama iterasi.** Untuk tiap node i , *rank*-nya dihitung sebagai berikut:

$$R_k(i) = \sum_{j \in \text{in}_i} \left(\frac{d}{o_j} * R_k(j) \right) + \min_k$$

Jika *edge*-nya berbobot, maka *rank*-nya dihitung menggunakan rumus sebagai berikut:

$$R_k(i) = \sum_{j \in \text{in}_i} \left(\frac{d}{\sum_{m \in \text{out}_j} W(j, m)} * W(j, i) * R_{k-1}(j) \right) + \min_k$$

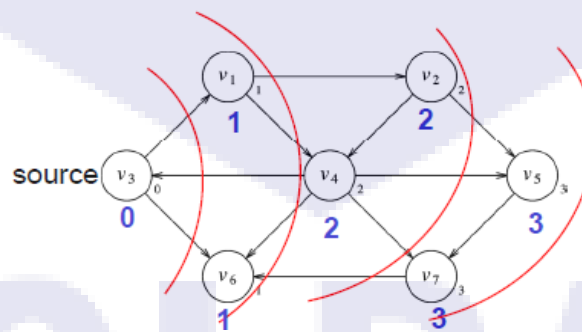
- d. **Antara setiap iterasi kelima.** Algoritmanya menggunakan *quadraticextrapolation* diantara setiap iterasi kelima untuk mempercepat konvergensi.
- e. **Akhir iterasi.** Setelah setiap iterasi, norm L1 dari sisa dihitung, yang merupakan jumlah dari perubahan absolut dari setiap nilai selama iterasi. Saat sisa L1 berada di bawah threshold tertentu, misal 0.001, penghitungan berhenti dan dianggap bahwa estimasi R dari *eigenvector* dominan telah ditemukan.

Tabel 1 Notasi yang digunakan untuk menghitung ranking vektor

Konstanta	Keterangan
d	Faktor terkecil dari penghitungan PageRank: $d = 0.85$
Variabel	Description
G	<i>Graph</i> yang akan dianalisis, direpresentasikan dengan <i>connectivity matrix</i>
λ_1	<i>Eigenvector</i> pertama dari G
n	Jumlah node pada G
m	Jumlah <i>link</i> pada G
R_k	Nilai ranking vektor untuk iterasi k , pendekatan dari λ_1
i_j	Jumlah <i>inlink</i> ke node j
o_j	Jumlah <i>outlink</i> dari node j
in_j	Himpunan node yang menghubungkan node j
out_j	Himpunan node yang node j hubungkan
$dead_G$	Himpunan <i>dead node</i> pada G , node tanpa <i>outlink</i>
$live_G$	Himpunan <i>live node</i> pada G , node dengan <i>outlink</i>
$W(i, j)$	Pembobotan link yang masuk dari node i ke node j
min_k	Ranking dasar atau minimum suatu node pada iterasi k karena link yang universal

Algoritma ReConRank akan dikaji lebih lanjut untuk mengetahui perilaku detail pada aplikasi perangkian hasil pencarian mesin pencari web semantik sehingga dapatdimodifikasi menggunakan konsep *query multi term*.

Unweighted Shortest Paths adalah menemukan path yang panjangnya terpendek memiliki ciri-ciri sebagai berikut: tidak ada nilai bobot pada *edge* (bobot semua *edge* dianggap sama).



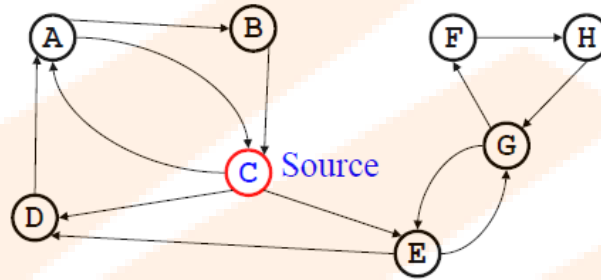
Gambar 9 2Unweighted shortest paths

Untuk setiap *vertex* (titik), harus diketahui bahwa:

1. Apakah sudah dilewati (*known*)
2. Jaraknya dari titik awal (d_v)
3. *Predecessor vertex* sepanjang *path* terpendek dari *vertex* awal (p_v).

Salah satu algoritma yang menjadi solusi untuk mencari *path* terpendek dari suatu *graph* adalah Breadth-First Search. Ide dasar dari algoritma ini adalah memulai dari

vertexsource, untuk menemukan simpul yang dapat dicapai dengan menggunakan *edge* 0, 1, 2, 3, ... , N-1.



Gambar 10 Mencari *path* terpendek difokuskan pada panjang *path*

Source code untuk mencari *path* terpendek tanpa pembobotan:

```

Distance[s] := 0
Enqueue(Q, s); Mark(s) //After a vertex is marked once
                        // it won't be enqueued again
while queue is not empty do
    X := Dequeue(Q);
    for each vertex Y adjacent to X do
        if Y is unmarked then
            Distance[Y] := Distance[X] + 1;
            Previous[Y] := X; //if we want to record paths
            Enqueue(Q, Y); Mark(Y);
    
```

Gambar 113 Algoritma Breadth-First Search

Zhou, Qi, et al [5] melakukan penelitian menggunakan SPARK dengan mengeksplorasi pendekatan baru dari pengadaptasian *keyword* dengan query web semantik: Pendekatan secara otomatis menerjemahkan *querykeyword* ke pertanyaan logika formal sehingga *end-user* dapat menggunakan *keyword* yang sudah lazim untuk melakukan pencarian semantic. Zhou mengatakan bahwa terjemahan SPARK terdiri dari 3 langkah utama yaitu pemetaan *term*, pembentukan *graph query*, dan perankingan *query*. Zhou membentuk *graph query* menggunakan algoritma Minimum Spanning Tree(algoritma pencarian jarak terpendek) yang diterapkan untuk membangun kemungkinan *query graph* untuk setiap set *query*. Dengan algoritma tersebut, Zhou dapat menyimpulkan ada tidaknya suatu hubungan eksplisit antar *term*. Zhou menghasilkan *query graph* dengan menyesuaikan aturan berikut:

1. Kelas *Resources* yang dipetakan oleh *term* atau ditemukan oleh eksplorasi *graph* dianggap sebagai node variabel
2. *Resources instances* dan *literal* dianggap sebagai node akhir
3. *Resources* properti dianggap sebagai *edge querygraph*.

Karena SPARQL adalah bahasa *query* berbasis pola *graph*, Zhou menganggap pengkonversian *query graph* menjadi sesuai SPARQL string *query* itu sangat mudah.

Unweighted Shortest Paths nantinya akan diterapkan pada perluasan algoritma ReConRank untuk mendapatkan *path* terpendek dari masing-masing *term* (sebagai *vertex*) yang telah diinputkan oleh *user*.

3. METODOLOGI

a. Jenis Penelitian

Jenis penelitian ini adalah penelitian kuantitatif, dimana penelitian ini merupakan penelitian ilmiah yang sistematis terhadap bagian-bagian dan fenomena serta hubungan-hubungannya yang bertujuan untuk mengembangkan dan menggunakan model-model matematis, teori-teori dan/atau hipotesis yang berkaitan dengan suatu fenomena.

b. Objek Penelitian

Objek penelitian merupakan sesuatu yang akan menjadi pusat penelitian. Penelitian ini difokuskan pada skema perluasan sistem *query multi term* dan *shortest path* pada algoritma perankingan ReConRank.

c. Data Penelitian

Data yang digunakan merupakan data sekunder, dimana data penelitian diperoleh dari sumber lain. Data didapatkan dari berbagai sumber yang tersedia media internet dengan jenis data RDF/OWL. Untuk penelitian ini, data yang akan digunakan harus memiliki *context* sebanyak 2 atau lebih *context*, memiliki *resources* yang merupakan URI maupun *object literal*, serta *predicates*. Karena pada penelitian sebelumnya memenuhi persyaratan untuk penelitian ini, maka penulis akan menggunakan data yang sama.

d. Tahapan Penelitian

Berikut merupakan tahapan-tahapan yang dilakukan dalam penelitian ini:

d.1. Pendefinisian Masalah

Tahapan ini merupakan tahapan awal dalam penelitian ini. Pendefinisian masalah yang dilakukan mencakup penentuan topik penelitian, *research question*, hipotesa, serta batasan masalah. Topik penelitian yang dipilih adalah mengenai penerapan konsep *query multi term* pada algoritma ReConRank yang merupakan kelanjutan dari penelitian sebelumnya yang dilakukan oleh [1], mengenai algoritma ReConRank dengan *query single term*. Dari topik yang dipilih, maka *research question* dan hipotesa ditentukan untuk memaparkan apa yang ingin diketahui dari penelitian yang akan dilakukan serta batasan masalah untuk menjelaskan batasan-batasan yang perlu diteliti dan tidak.

d.2. Studi Literatur

Pada tahapan ini penulis melakukan studi literatur untuk memahami teori yang terkait dengan penelitian yang akan dilakukan seperti teori mengenai Web Semantik, algoritma ReConRank, cara menghitung keakuratan dengan *precision* dan teknologi yang akan digunakan dalam membangun aplikasi. Studi dilakukan dengan cara mencari referensi yang bersumber dari berbagai jurnal dan halaman web yang didapat dari media internet.

d.3. Pengumpulan Data

Pengumpulan data bertujuan untuk mendapatkan data yang akan dibutuhkan dalam pengembangan aplikasi. Tahap ini dilakukan dengan cara mencari sumber yang relevan dengan penelitian ini. Data yang digunakan adalah data dengan bahasa ontologi OWL yang menggunakan format *metadata* RDF atau biasa disebut RDF/OWL. Nama *graph* dari OWL yang didapatkan disertai dengan jumlah datanya dapat dilihat pada Tabel 2.

Tabel1 Data penelitian

No	Nama <i>Graph</i>	Jumlah Data
1	http://www.owl-ontologies.com/travel.owl	212
2	http://www.atl.lmco.com/projects/ontology/ontologies/hotel/hotelA.owl	82
3	http://www.atl.lmco.com/projects/ontology/ontologies/museum.owl	117
	Total	411

d.4. Analisis

Analisis yang dilakukan pada penelitian ini adalah sebagai berikut:

1. Analisis Algoritma ReConRank

Pada tahapan ini dilakukan proses analisis terhadap algoritma ReConRank tentang bagaimana perilaku algoritma ReConRank dalam merankingkan hasil pencarian sesuai dengan *query* dari *user* yang hanya dapat menangani *query single term*. Dari penganalisisan ini, akan ditentukan letak penerapan skema perluasan algoritma ReConRank.

2. Analisis ReConRank dengan Pendekatan *Shortest Path*

Pada tahapan ini, dilakukan penganalisisan algoritma ReConRank tentang bagaimana menerapkan konsep *query multi term* pada algoritma ReConRank. Setiap tahapan pada algoritma ReConRank dianalisis sehingga mendapatkan algoritma ReConRank yang telah dimodifikasi oleh konsep *multi term*.

Tahapan pertama pada penganalisisan algoritma ReConRank adalah analisis pembentukan topical subgraph. Pada *query single term*, topical subgraph ini terdiri dari *subject* dari *keyword* yang cocok dengan *query* yang diinputkan oleh *user* dan *outlink* yang dimilikinya. Karena pada penelitian ini menggunakan konsep *multi term*, maka peneliti menggunakan penerapan *unweighted shortest path* untuk menentukan lintasan terpendek dari *graph*. Lintasan yang dimaksud adalah lintasan antarterm *query*. Semakin pendek lintasan yang didapat, maka semakin besar keterkaitan antarterm. Algoritma *unweighted shortest path* yang akan dipakai dalam penelitian ini adalah algoritma Breadth-First Search. Dari algoritma ini, akan didapatkan hasil pencarian berdasarkan *shortest path*. Jika dalam penentuan *shortest path* terdapat panjang lintasan yang sama, maka akan dirankingkan menggunakan algoritma ReConRank. *Path* tersebut merupakan input parameter untuk topical subgraph. Setelah tahap penganalisisan topical subgraph adalah analisis pada pembentukan *graph* selanjutnya yaitu Resources Graph, Context Graph, dan ReCon Graph. Resources Graph merupakan *graph* yang terdiri dari path dan hubungan antarpathnya, Context Graph terdiri dari *context* dan hubungan antarcontextnya, serta ReCon Graph dibentuk dari penggabungan antara Resources Graph dan Context Graph.

3. Analisis Alat Ukur

Pada penelitian ini dibutuhkan keakuratan hasil pencarian dengan *query multi term* pada algoritma ReConRank. Oleh karena itu, penelitian ini menggunakan konsep *precision* untuk mengukur keakuratannya. Setelah algoritma ReConRank dimodifikasi agar dapat menangani *query multi term*, maka dilakukan pengukuran terhadap hasil pencarian dengan *query multi term*.

d5. Eksperimen dan Pembahasan

Eksperimen dilakukan dengan cara membuat *query* yang diinputkan pada mesin pencari,. *Query* yang akan diinputkan dibedakan menjadi beberapa jenis dilihat dari jumlah *term* yang diinputkan. Penulis menentukan *query* dengan jumlah *term* yang berbeda sebagai alat eksperimen karena penelitian ini menggunakan *multi term* yang berarti bahwa *term* yang dimasukkan terdiri dari 2 *term* atau lebih sehingga harus dipastikan bahwa aplikasi yang dibangun mampu menangani *query multi term*. Berikut beberapa tingkatan *query* yang akan dijadikan alat untuk eksperimen:

- a. *Query* tingkat mudah: jumlah *query* sebanyak 2 *term*.
- b. *Query* tingkat sedang: jumlah *query* sebanyak 3 *term*.
- c. *Query* tingkat sulit: jumlah *query* sebanyak 4 *term*.

Setiap *query*, selanjutnya, akan diidentifikasi apakah hasil setiap *query* tersebut relevan dengan *query* yang diinputkan dengan mengukur keakuratan dari hasil pencarian tersebut.

Penghitungan keakuratan dilakukan secara manual, yaitu dengan menghitung *link* mana saja yang sesuai dengan *keyword user* maupun yang tidak menggunakan rumus *precision*.

$$Precision = \frac{\text{Jumlah link relevan yang terambil}}{\text{Jumlah link yang terambil dalam pencarian}} \times 100\%$$

Link yang dimaksud diatas adalah URI yang merupakan node-node pada *path*. Node yang dipilih sebagai *link* relevan adalah *node* yang terkait dengan *keyword* yang dimasukkan oleh *user*. Sedangkan *link* yang terambil merupakan semua *node* yang terdapat pada masing-masing *path*.

Pada tahap ini juga dilakukan pembahasan dan evaluasi untuk mengetahui bagaimana pengaruh perilaku algoritma ReConRank setelah diperluas dengan *query multi term* terhadap hasil keakuratan berdasarkan pengujian yang telah dilakukan sebelumnya.

4. HASIL DAN PEMBAHASAN

Pada bab ini dilakukan analisis mengenai problem domain. Selanjutnya dilakukan analisis arsitektur mesin pencari web semantik dengan menentukan pada bagian mana yang akan digunakan pada penelitian ini. Setelah itu, dilakukan penganalisisan mengenai algoritma ReConRank dengan menentukan bagian mana dari tahap algoritma tersebut yang akan diperluas menggunakan konsep *query multiterm*. Untuk mendapatkan keterkaitan antarterm yang telah diinputkan oleh *user*, maka dari *graph* hasil perluasan dengan *query multi term* dilakukan penganalisisan menggunakan *shortest path*. Dari hasil penentuan *shortest path*, dilakukan tahap pembuatan Resources graph, Context graph, dan ReCon graph apabila hasil pengurutan dengan *shortest path* bernilai sama. Dari ReCon graph, dilakukan perankingan. Terakhir, penghitungan keakuratan dilakukan dengan menggunakan penghitungan *precision*. Berikut ini penjelasan yang lebih detail mengenai hal tersebut.

4.1. Analisis Problem Domain

Analisis problem domain merupakan tahap untuk menjelaskan mengenai apa yang akan diteliti dan keterkaitan antarstudi yang telah dilakukan, mulai dari mesin pencari

web semantik sampai dengan mendapatkan hasil keakuratan dari hasil pencariannya. Berikut ini hasil analisis domain yang telah dilakukan:

4.1.1. Mesin Pencari Web Semantik

Web semantik merupakan perluasan dari web sebelumnya (web 2.0 atau web konvensional) yang mampu menyediakan penerjemah bahasa manusia menjadi bahasa mesin. Dengan menggunakan mesin pencari web semantik, *user* dapat mendapatkan informasi berdasarkan *query* yang diinputkan. Selanjutnya, mesin pencari akan memproses *query* tersebut dan melakukan proses pencarian terhadap data-data yang berada pada database Virtuoso berdasarkan *query* dari *user*. Terakhir, mesin pencari akan menampilkan hasil pencariannya kepada *user* yang berupa *listURL* beserta konten dari *query* itu sendiri.

Dalam memproses suatu *query*, mesin pencari memerlukan perankingan yang berfungsi untuk memprioritaskan hasil pencariannya sebelum ditampilkan kepada *user*. Perankingan merupakan salah satu bagian dari arsitektur *Semantic Web Search Engine* (SWSE). Bagian-bagian dari arsitektur SWSE terdapat pada Gambar 3 di subbab 2.2.1. Dari ketiga bagian SWSE, perankingan berada pada bagian *Semantic Search and Query Engine* yang merupakan bagian inti dari arsitektur SWSE, tepatnya pada YARS2 RDF Store. YARS2 merupakan arsitektur terdistribusi terukur untuk *indexing* dan *querying* dataset RDF yang besar dan beroperasi pada model data *named graph*, dimana RDF *triple* (subject, predicate, object) ditambahkan dengan *context* sehingga membentuk quadruple (*subject, predicate, object, context*) [7].

RDF merupakan singkatan dari *Resources Description Framework* yang memungkinkan komunikasi dan interaksi pada level mesin. *Query* dari *user* akan diubah menjadi bahasa mesin yang berupa format RDF. RDF terdiri dari tiga komposisi atau yang disebut RDF *triple*, antara lain *subject*, *predicate*, dan *object*. *Predicate* merupakan komposisi yang menerangkan sudut pandang dari *subject* yang dijelaskan *object*, sementara *subject* dan *object* merupakan entitas. *Object* di dalam RDF dapat menjadi *subject* yang diterangkan oleh *object* yang lainnya. Dengan inilah *object* dapat berupa masukan yang dapat diterangkan secara jelas dan detail, sesuai dengan keinginan user yang memberikan *keyword*. *Subject* dan *object* dalam RDF memiliki jenis yang berbeda yaitu *literal* dan *resource*. *Literal* berbentuk persegi panjang yang merupakan *keyword*. Sedangkan *resource* berbentuk oval yang merupakan URI.

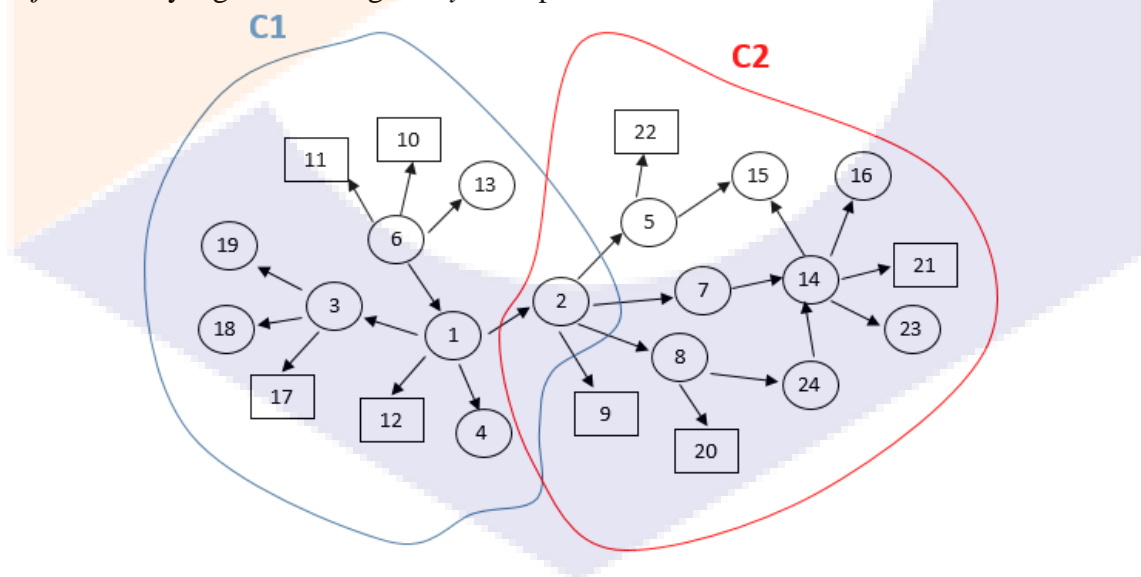
4.1.1.1. Algoritma ReConRank dengan Query Multi Term

Algoritma perankingan yang akan digunakan untuk mesin pencari web semantik adalah algoritma ReConRank. Parameter input pada algoritma ini adalah topical subgraph, sedangkan *outputnya* adalah nilai ranking untuk setiap *vertex*. Berdasarkan studi yang telah dilakukan, berikut tahapan-tahapan pada algoritma ReConRank:

1. Membuat *resourcesgraph* dari parameter input topical subgraph dengan menentukan *resources* yang setidaknya pernah berposisi menjadi *subject* sebagai *vertexnya* dan hubungan antara *resources* sebagai *edgenya*.

2. Membuat *contexts graph* dari parameter input topical subgraph dengan menentukan *context* sebagai *vertex*-nya dan hubungan antara *context*-nya sebagai *edge*.
3. Menggabungkan *resources graph* dan *contexts graph* dengan melakukan tahap sebagai berikut.
 - a. Menentukan *link* antara *context* dan *resource(s)* yang dikandungnya.
 - b. Menentukan *link* antara *resources* dan *context* yang mengandungnya.
 - c. Menentukan *link* dari *context* ke *context*. Contohnya *resource* di *context* A berhubungan dengan *context* B, maka secara tidak langsung *context* A berhubungan dengan *context* B.
4. Menghitung nilai ranking untuk setiap *vertex* pada *ReCon graph* menggunakan algoritma penghitungan ranking vektor (subbab 2.2.1).

Dalam suatu dataset, RDF triple (*subject*, *predicate*, *object*) harus ditentukan terlebih dahulu dengan mengilustrasikan menggunakan RDF graph. Selanjutnya menentukan *named graph* dari RDF graph tersebut dengan menambahkan *context* sehingga membentuk *quadruple* (*subject*, *predicate*, *object*, *context*) agar dapat membuat suatu topical subgraph yang akan dijadikan parameter *input* untuk algoritma ReConRank. Topical subgraph didapat dengan cara memilih *outlink* yang dimiliki oleh *subject* dari *object literal* yang cocok dengan *keyword* pencarian.



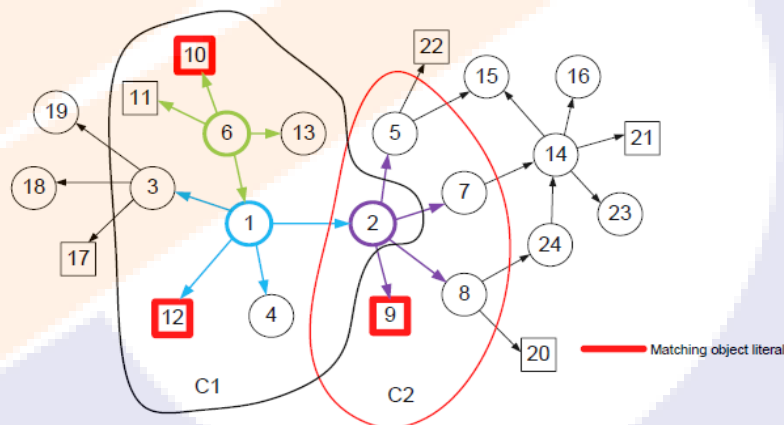
Gambar 124 Dataset dalam bentuk *named graph*

Gambar 12 menunjukkan dataset yang direpresentasikan dalam bentuk *graph*. Berdasarkan gambar diatas, dapat disimpulkan bahwa pada *named graph* tersebut terdapat:

1. *Subject* sebanyak 9 buah diantaranya 1, 2, 3, 5, 6, 7, 8, 14, dan 24.
2. *Object* terdapat pada semua *node/vertex*, kecuali *node* 6. *Object* yang merupakan *literal* antara lain 10, 11, 12, 17, 20, 21, 22 dan sisanya adalah *object* yang merupakan *resources*.
3. *Predicate* terdapat pada semua *edge* yang ditandai dengan tanda panah.
4. *Context* terdiri dari C1 dan C2.

5. Konsep merupakan *outlink* dan *inlink* dari *node* itu sendiri. Pada satu *context* terdapat beberapa konsep. Salah satu konsep yang terdapat pada Gambar 15 adalah pada *node* 6. *Node* 6 memiliki konsep yang terdiri dari *node* 11, 10, 13, dan 1. Dengan kata lain, *node-node* tersebut berada dalam konsep yang sama yaitu *node* 6. Setiap *path* yang terbentuk memiliki beberapa konsep. Misalnya *path* antara *node* 10 dengan *node* 12. Jalur pada *path* tersebut adalah 10, 6, 1, dan 12. *Path* tersebut memiliki 4 konsep yang terdiri dari konsep *node* 10 (*node* 6), konsep *node* 6 (*node* 10, 6, 1, dan 12), konsep *node* 1 (*node* 3, 6, 2, 4, dan 12), dan konsep *node* 12 (*node* 1).

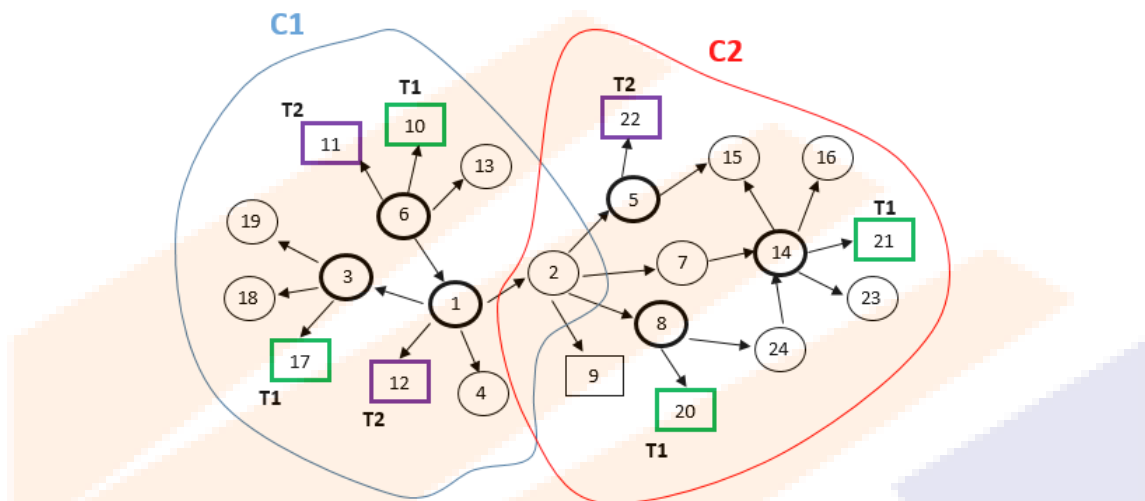
Untuk menerapkan sistem *multiterm* pada topical subgraph, diperlukan tahap yang berbeda dari *single term* yang mana pada sistem *single term* akan dilakukan penentuan *subject* dari *object literal* yang cocok dengan *keyword* (Gambar 13). *Subject* tersebut dinamakan *resources*, yang diantaranya no 1, 2, dan 6. *Object literal* yang cocok dengan *keyword* ditandai dengan persegi berwarna merah.



Gambar 13 Topical subgraph untuk *query single term*

Dari *named graph* pada Gambar 12, dapat dibentuk topical subgraph untuk *query* multi *term* dengan menentukan *subject* pada masing-masing *term*. Setiap *term* merupakan *object literal* dan tanda panah yang menunjuk *object literal* merupakan *predicate*, maka *node/vertex* yang menunjuk *object literal*-nya adalah *subject*. Dimisalkan *query* yang diinputkan *user* adalah “orang pertama”. *Term* “orang” didefinisikan sebagai T1 dan *term* “pertama” sebagai T2. Gambar 21 menunjukkan bahwa T1 ditandai dengan warna hijau sedangkan T2 ditandai dengan warna ungu. *Subject* dari masing-masing *term* ditandai dengan warna hitam tebal. Setiap T1 akan dihubungkan dengan T2 yang berada di C1 maupun C2.

Apabila salah satu *term* tidak tersedia di dalam ontologi, maka sistem akan menganggap *keyword* tersebut sebagai *single term*. Sebagai contoh pada penginputan *keyword* “orang pertama”, *term* “pertama” tidak tersedia di dalam ontologi. Sistem akan melakukan pencarian menggunakan algoritma ReConRank untuk *single term* dengan inputan *keyword*nya adalah “orang”. Aplikasi seperti itu telah dibangun pada penelitian sebelumnya. Namun jika semua *term* yang dimasukkan tidak tersedia di dalam ontologi, maka sistem tidak akan menampilkan hasil pencarian apapun.



Gambar 14 5Subject dari *object literal* T1 dan T1

Dari Gambar 14, *multiterm* T1 dan T2 yang ada pada graph digabungkan sehingga membentuk suatu *path*. *Path* tersebut dihitung jumlah *nodenya* untuk dibandingkan dengan *path* lain (Tabel3). *Path* dengan jumlah *node* paling sedikit merupakan yang menempati ranking paling tinggi dan yang jumlah *nodenya* paling banyak merupakan *path* dengan ranking yang paling rendah. Apabila ada jumlah *node* yang sama, maka *path* tersebut akan diranking menggunakan algoritma ReConRank. Berikut hasil penentuan *path* yang dibentuk antara T1 dan T2 beserta jumlah *nodenya* yang ditampilkan pada Tabel 3.

Tabel2 Penentuan *path* terhadap T1 dan T2

T1 – T2	<i>Path</i>	Jumlah <i>Node</i>
10 - 11	10 – 6 – 11	3
10 - 12	10 – 6 – 1 – 12	4
10 - 22	10 – 6 – 1 – 2 – 5 – 22	6
17 – 11	17 – 3 – 1 – 6 – 11	5
17 - 12	17 – 3 – 1 – 12	4
17 - 22	17 – 3 – 1 – 2 – 5 – 22	6
20 - 11	20 – 8 – 2 – 1 – 6 – 11	6
20 - 12	20 – 8 – 2 – 1 – 12	5
20 - 22	20 – 8 – 2 – 5 – 22	5
21 – 11	21 – 14 – 7 – 2 – 1 – 6 – 11	7
21 - 12	21 – 14 – 7 – 2 – 1 – 12	6
21 - 22	21 – 14 – 15 – 5 – 22	5

Setelah *path* ditentukan, maka selanjutnya mengurutkan *path* secara *ascending* berdasarkan jumlah *nodenya* (Tabel 4). Dari Tabel 3 dapat disimpulkan bahwa:

1. *Path* antara 10 dan 11 merupakan *path* yang paling sedikit *nodenya* sehingga *path* tersebut ditempatkan sebagai ranking pertama
2. *Path* 10 - 12 dan 17 – 12 memiliki jumlah *node* yang sama sehingga akan diranking menggunakan algoritma ReConRank untuk menentukan posisi ranking ke-2 dan ke-3
3. *Path* 17 – 11, 20 – 12, 20 – 22, dan 21 – 22 memiliki jumlah *node* yang sama. Pada *path* ini akan dilakukan perankingan menggunakan algoritma ReConRank untuk menentukan posisi ranking ke-4, 5, 6, dan 7.

4. *Path* 10 – 22, 17 – 22, 20 – 11, dan 21 – 12 juga akan diranking menggunakan algoritma ReConRank untuk menentukan ranking ke-8, 9, 10, 11 karena *path* tersebut memiliki jumlah *node* yang sama.
5. *Path* 21 – 11 mendapatkan ranking ke-12 dan merupakan ranking yang terakhir karena memiliki *path* yang paling banyak jumlah *node*nya.

Tabel3 Pengurutan *shortest path* berdasarkan jumlah *node*

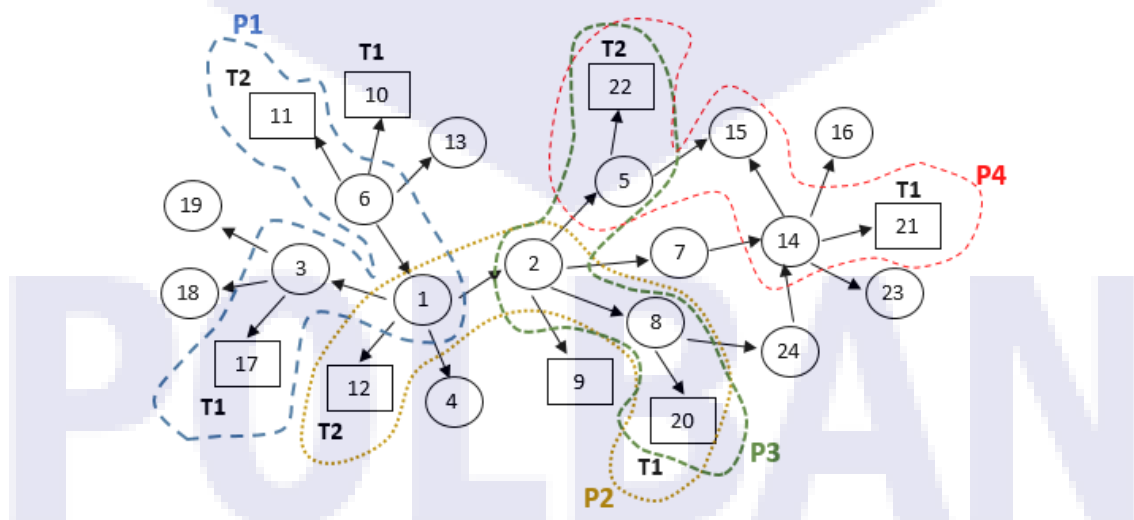
T1 – T2	<i>Path</i>	Jumlah <i>Node</i>	Ranking ke-
10 – 11	10 – 6 – 11	3	1
10 – 12	10 – 6 – 1 – 12	4	2 atau 3
17 – 12	17 – 3 – 1 – 12	4	2 atau 3
17 – 11	17 – 3 – 1 – 6 – 11	5	4 atau 5 atau 6 atau 7
20 – 12	20 – 8 – 2 – 1 – 12	5	4 atau 5 atau 6 atau 7
20 – 22	20 – 8 – 2 – 5 – 22	5	4 atau 5 atau 6 atau 7
21 – 22	21 – 14 – 15 – 5 – 22	5	4 atau 5 atau 6 atau 7
10 – 22	10 – 6 – 1 – 2 – 5 – 22	6	8 atau 9 atau 10 atau 11
17 – 22	17 – 3 – 1 – 2 – 5 – 22	6	8 atau 9 atau 10 atau 11
20 – 11	20 – 8 – 2 – 1 – 6 – 11	6	8 atau 9 atau 10 atau 11
21 – 12	21 – 14 – 7 – 2 – 1 – 12	6	8 atau 9 atau 10 atau 11
21 – 11	21 – 14 – 7 – 2 – 1 – 6 – 11	7	12

Untuk *path* yang jumlah *node* yang sama, masing-masing *path* dihitung rankingnya menggunakan algoritma ReConRank dengan tahapan yang sudah dijelaskan pada subbab 2.2. Berikut menentukan ranking ke-4, 5, 6, dan 7 untuk *path* 17 – 11, 20 – 12, 20 – 22 dan 21 – 22.

1. Pembentukan Topical Subgraph

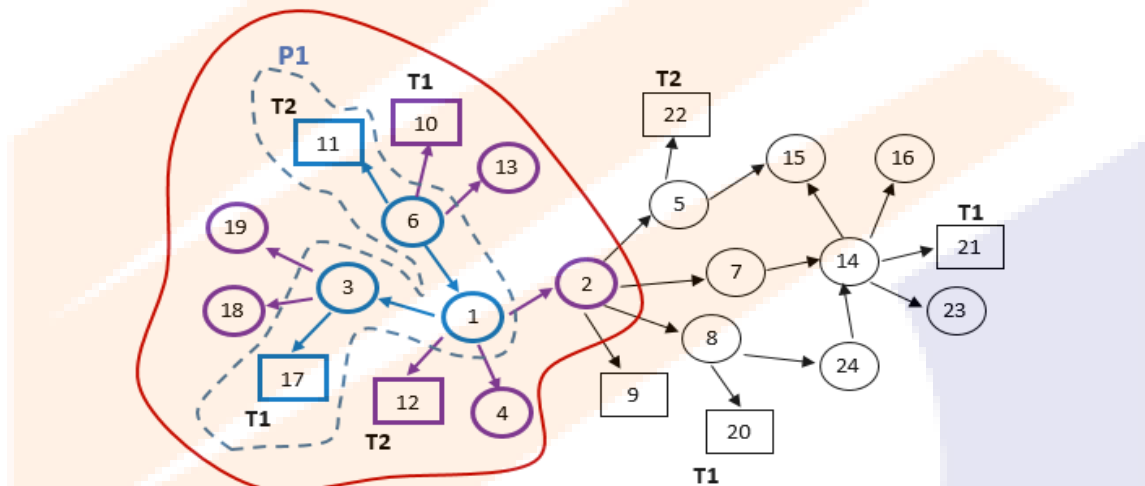
Topical subgraph yang dibentuk merupakan subgraph terdiri dari *path*, serta outlink pada masing-masing node pada suatu *path*.

Gambar 15 berikut ini merupakan penggambaran dari *path* 17 – 11, 20 – 12, 20 – 22 dan 21 – 22. *Path* 17 – 11 ditampilkan dengan garis berwarna biru dengan nama P1, *path* 20 – 12 berwarna kuning dengan nama P2, *path* 20 – 22 berwarna hijau dengan nama P3, dan *path* 21 – 22 berwarna merah dengan nama P4.

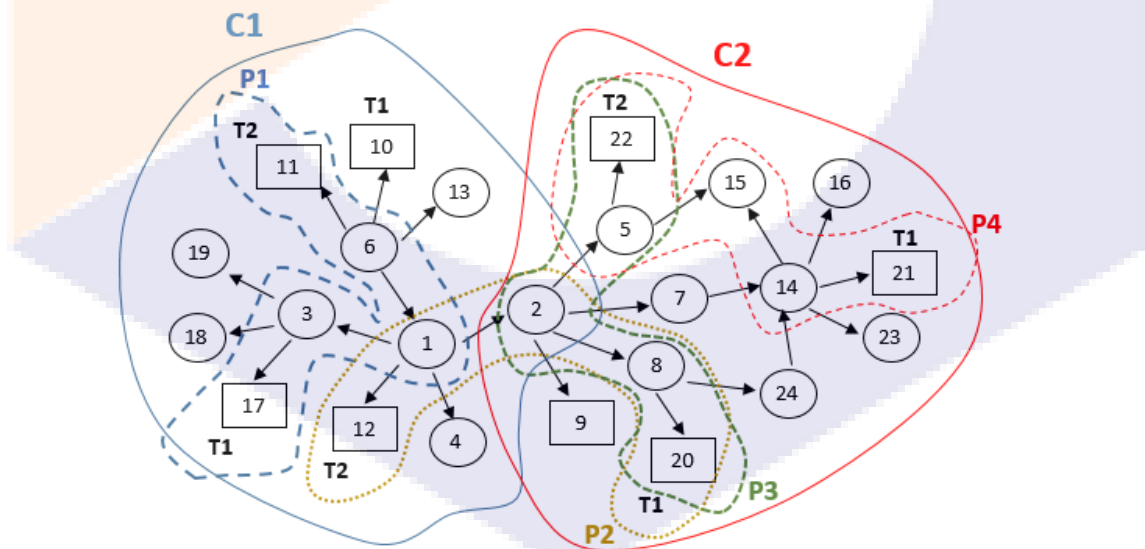


Gambar 15 Dua *path* dengan jumlah *node* 4 buah

Pada Gambar 16 bawah ini menunjukkan bahwa topical subgraph (berwarna merah) untuk salah satu *path* yaitu P1 (berwarna biru) merupakan *path* itu sendiri serta *outlink* pada node yang terdapat pada *path* P1 yang diantaranya *outlink* dari node 6, 1, dan 3. Sama halnya untuk P2, P3, dan P4 sehingga topical subgraph untuk semua *path* terbentuk seperti pada Gambar 17.



Gambar 16 Topical Subgraph untuk *path* P1



Gambar 17 Topical Subgraph untuk *path* berjumlah node 5

2. Pembentukan Resources Graph

Resources Graph merupakan *graph* yang terdiri dari beberapa *resource* yang berhubungan antara 2 *object literal* sebagai *vertex* dan hubungan antarp ath sebagai *edge*.

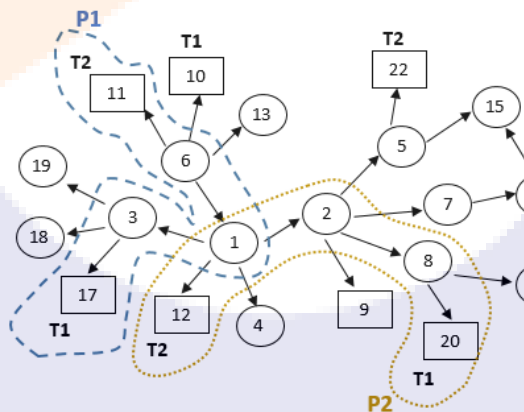
Pada setiap *object literal* pada *path* tersebut, ditentukan *resources* yang merupakan *subject* dari *object literal* itu. Dari Gambar 17 dapat ditentukan hubungan antar *path* pada topical subgraph tersebut. Pada node di masing-masing *path* harus dipastikan terlebih dahulu apakah node pada *path* satu berhubungan dengan node pada *path* lain. Misalnya untuk penentuan hubungan antara *path* P1 dengan P2

(Gambar 18). *Path* P1 terdiri dari node 17, 3, 1, 6, dan 11 memiliki hubungan dengan P2 (20 – 8 – 2 – 1 – 12) karena memenuhi persyaratan berikut.

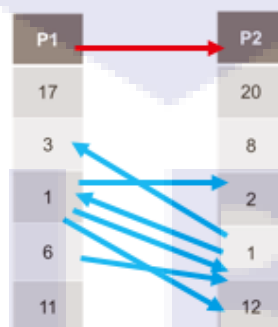
- Salah satu atau beberapa node pada P1 memiliki *outlink* yang berarah ke salah satu atau beberapa node pada P2, yaitu node 1 yang memiliki *outlink* dengan node 2 dan node 12 serta node 6 memiliki *outlink* dengan node 1.
- Salah satu atau beberapa node pada P2 memiliki *outlink* yang berarah ke salah satu atau beberapa node pada P1. Node tersebut diantaranya node 1 memiliki *outlink* dengan node 3.
- Terdapat node yang terdapat pada *path* P1 dan P2. Node yang dimaksud adalah node 1. Karena itu, node 1 pada *path* P1 saling berhubungan dengan node P2.

Berdasarkan uraian diatas dapat diketahui bahwa jumlah *outlink* yang dimiliki oleh node-node pada P1 adalah sebanyak 4 buah dan jumlah *outlink* pada P2 adalah sebanyak 2 buah. Karena jumlah *outlink* pada P1 lebih banyak daripada P2, maka *edge* yang terbentuk berarah dari P1 ke P2.

Pada gambar 19 mengilustrasikan penentuan hubungan antara P1 dengan P2. Tanda panah berwarna biru merupakan *link* antara node P1 dengan P2 dan tanda panah berwarna merah adalah hasil pembentukan *edge* antara path P1 dan P2.



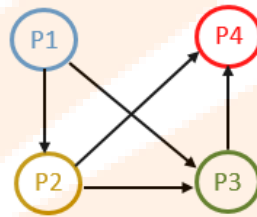
Gambar 18 *Path* P1 dan P2



Gambar 19 Penentuan *edge* antara P1 dan P2

Sama halnya dengan P1 dan P2 diatas, cara penentuan *edge* juga dilakukan antara P1 dengan P3, P1 dengan P4, P2 dengan P3, P2 dengan P4, dan P3 dengan P4 sehingga membentuk Resources Graph. Pada Resources Graph, P1, P2, P3 dan P4

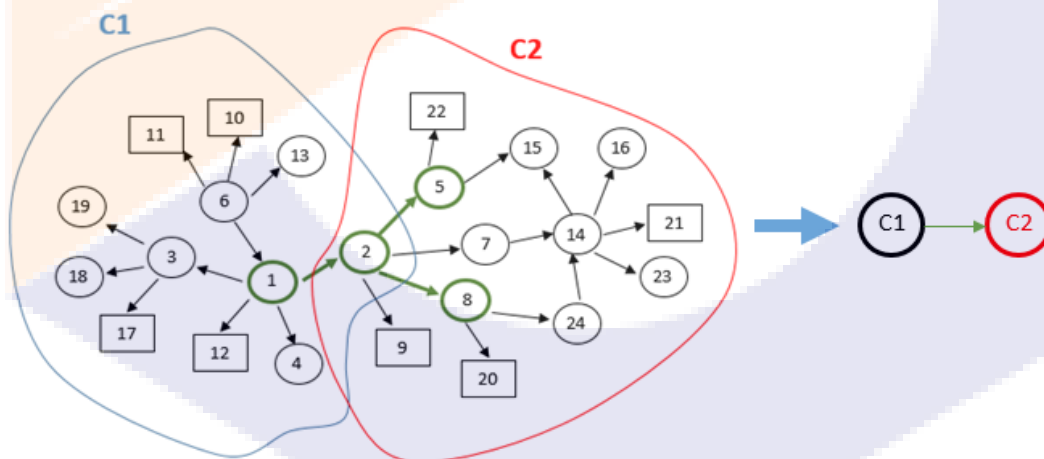
dijadikan *vertex* dan hubungan antara keempatnya merupakan *edge*. Hasil pembentukan Resources Graph dapat dilihat dari Gambar 20.



Gambar 20 Hasil pembentukan Resources Graph

3. Pembentukan Context Graph

Context Graph didapatkan dari hubungan antarcontext yang terdapat pada topical subgraph. Pada *graph* ini, *context*-nya dijadikan sebagai *vertex* dan hubungan antarcontextnya sebagai *edge*. Dari gambar 17, dapat diketahui bahwa pada *graph* tersebut jumlah *context*-nya 2, yaitu C1 dan C2. Hubungan antar *context* dapat dibentuk melalui *resources* yang ada di dalamnya. Jika *resources* pada *context* pertama berhubungan dengan *resources* pada *context* kedua, maka *context* pertama dan kedua memiliki hubungan. Hasil Context Graph dapat dilihat dari Gambar 21.



Gambar 21 Hasil pembentukan Context Graph

4. Pembentukan ReCon Graph

ReCon Graph dibentuk dari penggabungan antara Resources Graph dan Context Graph. *Vertex* pada ReCon Graph adalah *vertex* yang terdapat pada Resources Graph dan Context Graph, sedangkan *edgenya* adalah *edge* yang terdapat pada kedua *graph* tersebut juga namun ditambah dengan *implied links*. Berikut ini adalah cara menentukan *implied link* berdasarkan studi pada subbab 2.2:

- a. Menentukan *link* antara *context* dan *resource(s)* yang dikandungnya.

Dapat diketahui bahwa *context* C1 mengandung *resources* yang terdapat pada *path* P1, P2, dan P3. Sedangkan *context* C2 mengandung *resources* yang terdapat pada *path* P2, P3, dan P4. Maka *implied link* akan dibuat antara *context* dengan *resources* tersebut.

- b. Menentukan *link* antara *resources* dan *context* yang mengandungnya.

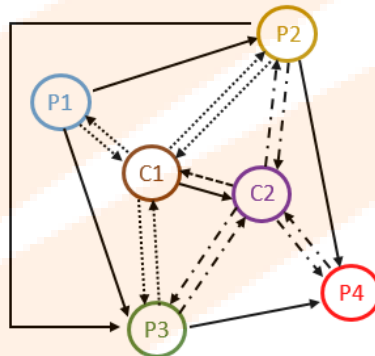
Untuk tahapan ini, *implied link* akan dibuat antara *resources* dan *context* dengan menghubungkan *resources* pada *path* P1 dengan *context* C1, *path* P2

dengan *context* C1 dan C2, *path* P3 dengan *context* C1 dan C2, serta P4 dengan *context* C2.

- c. Menentukan *link* dari *context* ke *context*.

Karena *resource* pada *path* P2 di *context* C1 berhubungan dengan *context* C2 dan *path* P3 di *context* C2 berhubungan dengan *context* C1, maka secara tidak langsung *context* C1 saling berhubungan dengan *context* C2.

Hasil *implied* ditunjukkan pada Gambar 22 dengan hasil *implied link* ditandai dengan garis hitam putus-putus.



Gambar 22 6ReCon Graph yang dibentuk dari hasil *implied link*

5. Penghitungan Ranking dari ReCon Graph

ReCon Graph yang telah dibentuk akan dihitung nilai rankingnya. Berikut langkah-langkah yang harus dilakukan untuk menghitung nilai ranking:

- a. Menentukan *inlink* dan *outlink* dari masing-masing *node* pada *path*.

Inlink merupakan arah *edge* yang menunjuk ke *resources* lain, sedangkan *outlink* merupakan arah *edge* yang ditunjuk oleh *resources* lain. *Inlink* pada suatu *path* dapat diketahui dari arah tanda panah pada *path* antara term T1 ke T2. Jika arah panah searah dengan arah *term* dari T1 ke T2, maka disebut *inlink*. Namun apabila sebaliknya, maka disebut *outlink*. Hal ini dapat diketahui dengan menggunakan contoh *keyword* yang diinputkan adalah “orang pertama” dengan *term* T1 adalah “orang” dan *term* T2 adalah “pertama”, maka maknanya berbeda jika arah *edge*-nya antara T2 ke T1 sehingga membentuk kata “pertama orang”. Dengan begitu, maka dapat disimpulkan bahwa arah *edge* antara suatu *resources* berpengaruh terhadap nilai *inlink* dan *outlink*.

Tabel 4 *Inlink* dan *outlink* pada P1, P2, P3, dan P4

<i>Path</i>	T1 – T2	Jalur	<i>Inlink</i>	<i>Outlink</i>
P1	17 – 11	17 – 3 – 1 – 6 – 11	2	5
P2	20 – 12	20 – 8 – 2 – 1 – 12	9	10
P3	20 – 22	20 – 8 – 2 – 5 – 22	11	10
P4	21 – 22	21 – 14 – 15 – 5 – 22	6	3

- b. Menentukan total nilai *inlink* (*i*) dan *outlink* (*o*) antara *path* dengan ReCon Graph. *Inlink* dan *outlink* yang dijadikan alat hitung perankingan merupakan hasil penjumlahan antara *inlink* dan *outlink* pada *path* dan ReCon Graph.

Tabel 5 Jumlah total *inlink* dan *outlink* pada *path* dan ReCon Graph

<i>Vertex</i>	<i>i</i> (<i>path</i>)	<i>o</i> (<i>path</i>)	<i>i</i> (ReCon Graph)	<i>o</i> (ReCon Graph)	Total <i>i</i>	Total <i>o</i>
P1	2	5	1	3	3	8

P2	9	10	3	4	12	14
P3	11	10	4	3	15	13
P4	6	3	1	3	7	6
C1	-	-	4	4	4	4
C2	-	-	4	4	4	4

- c. Menentukan estimasi *eigenvector* awal (R_o) dengan nilai jumlah *inlink* suatu *vertex* dibagi jumlah *link* pada ReCon Graph. Berikut adalah nilai *eigenvector* dari hasil penentuan *inlink* dan *outlink* yang ditunjukkan pada Tabel 6.

$$R_o = \frac{i_i}{m} = \left[\frac{3}{94} \quad \frac{12}{94} \quad \frac{15}{94} \quad \frac{7}{94} \quad \frac{4}{94} \quad \frac{4}{94} \right]$$

dimana $i = 1, 2, 3, 4, 5$ dengan nilai $i_1 = P1, i_2 = P2, i_3 = P3, i_4 = C1$, dan $i_5 = C2$

- d. Hitung nilai minimal untuk tiap *vertex* menggunakan persamaan (3) pada subbab 2.5. ReCon graph pada gambar tidak memiliki *dead node* dan memiliki 5 buah *live node*. Maka didapatkan nilai *min* sebagai berikut :

$$\begin{aligned} \min_k &= \frac{\sum_{j \in \text{dead}_G} R_{k-1}(j)}{n} + \sum_{j \in \text{live}_G} R_{k-1}(j) * \frac{1-d}{n} \\ \min_1 &= \left(\frac{3}{94} + \frac{12}{94} + \frac{15}{94} + \frac{7}{94} + \frac{4}{94} + \frac{4}{94} \right) \times \frac{1-0.85}{6} \\ &= 0.011968085 \end{aligned}$$

- e. Hitung nilai *eigenvector* dengan menggunakan rumus dibawah ini (subbab 2.2), karena *edge* pada *graph* tidak diberi bobot.

$$R_k(i) = \sum_{j \in \text{in}_i} \left(\frac{d}{o_j} * R_k(j) \right) + \min_k$$

$$R_1(P1) = \frac{0.85}{4} * \left(\frac{12}{94} + \frac{15}{94} + \frac{4}{94} \right) + 0.011968085$$

⋮

$$R_1(C1) = \frac{0.85}{4} * \left(\frac{12}{94} + \frac{15}{94} + \frac{7}{94} + \frac{4}{94} \right) + 0.011968085$$

- f. Hitung L1 norm nya dengan cara mengurangi nilai norma *eigenvector* pada iterasi sekarang dengan nilai norma *eigenvector* pada iterasi sebelumnya. Lakukan pengulangan terhadap langkah d-e-f sampai pada kondisi jika nilai L1 norm tersebut sudah mencapai threshold tertentu (misal 0.001), maka penghitungan dihentikan dan *eigenvector* yang merupakan nilai ranking telah ditemukan.

Dari langkah tersebut, akan didapatkan nilai ranking untuk menempati posisi ke-4, 5, 6, dan 7 bagi *path* 17 – 11, 20 – 12, 20 – 22 dan 21 – 22. Begitu pula untuk mendapatkan nilai ranking ke-2 atau 3 dan antara ke-8 atau 9 atau 10 atau 11.

Suatu *path* dikatakan merupakan *path* pendek apabila *node* yang dimilikinya berjumlah 3 yaitu 2 *object literal* dan 1 *node* antara kedua *object literal* tersebut. Sedangkan *path* panjang adalah *path* yang memiliki *node* 5 atau lebih.

Penghitungan Keakuratan

Berdasarkan studi yang telah dilakukan pada subbab 2.1.2, dapat diketahui bahwa *precision* adalah jumlah kelompok link relevan dari total jumlah *link* yang ditemukan oleh sistem. Berikut cara pengukuran *precision* yang akan dilakukan pada penelitian ini:

$$Precision = \frac{\text{Jumlah link relevan yang terambil}}{\text{Jumlah link yang terambil dalam pencarian}}$$

Tabel 7 menunjukkan bahwa penghitungan *precision* dapat diukur dengan mengacu pada rasio yang telah dikemukakan sebelumnya. Untuk menghitung rasio *precision*, jumlah *link* relevan yang terambil didefinisikan a sedangkan jumlah *link* yang terambil dalam penelusuran sebagai b.

Tabel 6 Penghitungan *precision*

	Relevant	Not Relevant	Total
Retrieved	a	b	a + b

Dengan demikian, maka *precision* dapat dinyatakan sebagai berikut:

$$P = \frac{a}{a + b}$$

Untuk mendapatkan nilai persentasi dari *precision*, maka rumus *precision* yang digunakan adalah :

$$P = \frac{a}{a + b} \times 100\%$$

Link yang relevan (a) dalam aplikasi yang dibangun adalah berupa URI yang relevan dari *keyword* yang telah diinputkan. Sedangkan *link* yang terambil oleh sistem terdiri dari *link* yang relevan maupun *link* yang tidak relevan. Penentuan *link* yang relevan dilihat dari URI yang muncul sebagai hasil pencarian.

Penghitungan keakuratan ini dilakukan secara manual, hampir sama dengan penelitian oleh B.T. Sampath Kumar dan J.N. Prakash [8]. Akan tetapi, perbedaan terletak pada banyaknya kategori kerelevanan. Penelitian ini cukup menggunakan 2 kategori kerelevanan, yaitu 'relevan' dan 'tidak relevan' atas dasar kriteria berikut ini.

- Jika *link* yang muncul erat kaitannya dengan *query* pencarian, maka dikategorikan sebagai 'relevan' dan diberi skor 1
- Jika *link* yang muncul tidak berhubungan dengan *query*, maka dikategorikan sebagai 'tidak relevan' dan diberi skor 0

Dari skor yang didapatkan, maka hasil *precision* didapatkan dengan menggunakan cara :

$$precision = \frac{\text{Total dari jumlah skor setiap link yang relevan}}{\text{Jumlah link yang muncul pada mesin pencari}}$$

Proses Menghitung Ranking untuk *Path* yang Jumlah Nodenya Sama

Pada tahap ini dilakukan penghitungan nilai ranking untuk setiap *vertex* pada *graph*. Berdasarkan studi yang dilakukan pada subbab 2.4, langkah-langkah yang dapat dilakukan untuk menghitung nilai tersebut adalah :

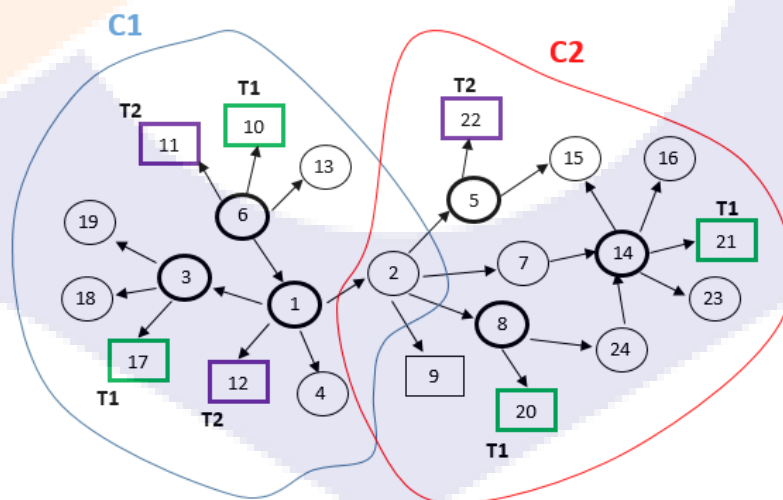
- a. Menentukan *inlink* dan *outlink* dari masing-masing *path*.
- b. Menentukan total nilai *inlink* (i) dan *outlink* (o) antara *path* dengan ReCon Graph

- c. Menentukan estimasi *eigenvector* awal (R_o)
- d. Hitung nilai minimal untuk tiap *vertex*
- e. Hitung nilai *eigenvector* selanjutnya
- f. Hitung L1 norm nya dengan cara mengurangkan nilai norma *eigenvector* pada iterasi sekarang dengan nilai norma *eigenvector* pada iterasi sebelumnya. Lakukan pengulangan terhadap langkah d-e-f sampai pada kondisi jika nilai L1 norm tersebut sudah mencapai threshold tertentu (misal 0.001), maka penghitungan dihentikan dan *eigenvector* yang merupakan nilai ranking telah ditemukan.

Eksperimen dan Menghitung Keakuratan

Eksperimen dilakukan dengan cara memasukkan beberapa *keyword*, lalu berdasarkan ReCon Graph yang dihasilkan beserta nilai ranking dari setiap *path* yang ada pada *graph* tersebut dapat dilihat perilaku dari algoritma ReConRank. *Keyword* yang dipilih merupakan :

1. *Keyword* yang akan menghasilkan konsep yang lebih spesifik.
Konsep yang dimaksud adalah *inlink* dan *outlink* dari *node* yang merupakan *subject* dari *object literal* yang cocok dengan *keyword* dari *user* serta *node* itu sendiri. *Keyword* yang paling relevan adalah *keyword* yang *term-term*nya berada dalam konsep yang sama.



Gambar 23 Ilustrasi data set *graph*

Misalnya *user* menginputkan *keyword* “baju hangat”. *Term* “baju” direpresentasikan sebagai T1 dan *term* “hangat” sebagai T2. Dari gambar 23, dapat dilihat bahwa *path* terpendek dari *keyword* “baju hangat” adalah 10-6-11. *Node* 10 dan 11 berada dalam konsep yang sama karena keduanya berada dalam konsep *node* 6 dan merupakan *outlink* dari *node* 6 itu. Konsep yang dimaksud adalah *inlink* dan *outlink* dari *node* 6, yaitu 1, 13, 10, 11 serta *node* 6 itu sendiri.

2. *Keyword* yang memiliki konsep lebih dari 1 sehingga *path* terpendeknya adalah *path* yang menghasilkan banyak *node*. *Path* yang terbentuk adalah *path* yang *node-nodenya* memiliki konsep berbeda diantara *node* yang terbentuknya. Misalnya : “baju bantal” dengan *term* “baju” cocok dengan *object literal* pada *node* 20 dan

term “bantal” pada *node* 22. Keyword “baju bantal” memiliki *path* terpendek 20-8-2-5-22. Antara kedua *term* tersebut memiliki konsep yang berbeda karena konsep *term* “baju” berada pada konsep *node* 8 dan *term* “bantal” berada pada konsep *node* 5. *Term* baju dengan bantal merupakan konsep yang berbeda sehingga akan menghasilkan *path* yang panjang.

3. *Keyword* yang akan dijadikan alat eksperimen merupakan *keyword* yang menghasilkan konsep berupa sinonim dari *keyword* yang dicari. Misalnya: *keyword* “baju baru” akan menghasilkan konsep “pakaian baru” juga karena *term* baju merupakan sinonim dari pakaian.

Berikut hasil eksperimen yang telah dilakukan :

1. Eksperimen dengan konsep yang lebih spesifik

Keyword terdiri dari 3 kali penginputan *keyword* yang terdiri dari :

- a. *Keyword* “bandung diponegoro”
- b. *Keyword* “bandung diponegoro museum”
- c. *Keyword* “bandung diponegoro museum geologi”

Dari eksperimen keempat *keyword* tersebut, akan menghasilkan ranking untuk masing-masing *vertex* dari ReCon Graph yang terbentuk. *Vertex* yang dimaksud adalah *path-path* dan *context*. Keterangan *node* yang dikandung oleh masing-masing *path* akan ditampilkan lampiran.

- a. *Keyword* “bandung diponegoro”

Berikut hasil perankingan yang telah dilakukan berdasarkan jumlah *node* yang sama. Jumlah *path* yang dihasilkan adalah sebanyak 6 *path*. Berikut hasil perankingan dari *keyword* “bandung diponegoro”.

Tabel 8 Hasil eksperimen *keyword* “bandung diponegoro”

No	<i>Path</i>	Σ <i>Node</i>	Σ <i>Inlink</i>	Σ <i>Outlink</i>	Nilai Ranking	Urutan ke-
1	P4	3	-	-	-	1
2	P2	4	3	2	0.2059451525671121	2
3	P5	4	2	3	0.1604668483012126	3
4	P0	5	4	4	0.17848533012843273	4
5	P1	5	4	4	0.17848533012843273	5
6	P3	5	2	2	0.10132773492641256	6

Data detail *path* untuk hasil eksperimen ini ditunjukkan pada Lampiran B.

Hasil *precision* dari *keyword* “bandung diponegoro” adalah sebagai berikut.

$$\text{precision} = \frac{\text{Total dari jumlah skor setiap link yang relevan}}{\text{Jumlah link yang muncul pada mesin pencari}}$$

$$= \frac{16}{22} \times 100\% = 72.73\%$$

Berikut data detail mengenai *link* yang terambil oleh sistem dan *link* yang relevan dengan *user* ditunjukkan pada Tabel 9 dan table 10.

Tabel 9 7 *Link* yang terambil dari *keyword* “bandung diponegoro”

No.	Nama <i>Link</i>	Skor	Jumlah <i>Link</i>
1	Hotel:santika	0	2
2	Hotel:vio	0	2
3	Museum:bandung	1	1
4	Museum:geologi	1	9

5	Museum:sribaduga	0	2
6	Travel:bandung	1	6
TOTAL			22

Tabel 10 8Link yang relevan dengan keyword “bandung diponegoro”

No.	Nama Link	Jumlah Skor
1	Museum:bandung	1
2	Museum:geologi	9
3	Travel:bandung	6
TOTAL		16

b. Keyword “bandung diponegoro museum”

Berikut hasil perankingan yang telah dilakukan berdasarkan jumlah *node* yang sama. Keyword ini menghasilkan jumlah *path* sebanyak 72 buah. Ranking ke-7 teratas ditunjukkan pada Tabel 11.

Tabel 11 Hasil eksperimen keyword “bandung diponegoro museum”

No	Path	$\sum Node$	$\sum Inlink$	$\sum Outlink$	Nilai Ranking	Urutan ke-
1	P48	3	-	-	-	1
2	P55	4	-	-	-	2
3	P24	5	5	3	0.1664658313165932	3
4	P31	5	5	3	0.1664658313165932	4
5	P60	5	3	5	0.10197998084131395	5
6	P67	5	3	5	0.10197998084131395	6
7	P36	6	14	14	0.0571036488791345	7
8

Data detail *path* untuk hasil eksperimen ini ditunjukkan pada Lampiran B. Berikut adalah hasil *precisionkeyword* “bandung diponegoro museum” :

$$\begin{aligned}
 precision &= \frac{\text{Total dari jumlah skor setiap link yang relevan}}{\text{Jumlah link yang muncul pada mesin pencari}} \\
 &= \frac{192}{373} \times 100\% \\
 &= 51.48\%
 \end{aligned}$$

Berikut data detail mengenai *link* yang terambil oleh sistem dan *link* yang relevan dengan *user* ditunjukkan pada Tabel 12 dan Tabel 13.

Tabel 12 9Link yang terambil dari keyword “bandung diponegoro museum”

No.	Nama Link	Skor	Jumlah Link
1	Hotel:santika	0	24
2	Hotel:vio	0	24
3	Museum:affandi	0	24
4	Museum:bandung	1	12
5	Museum:brawijaya	0	24
6	Museum:geologi	1	144
7	Museum:satriamandala	0	24
8	Museum:serangga	0	24
9	Museum:sribaduga	0	25
10	Museum:wayang	0	12
11	Travel:bandung	1	36
TOTAL			373

Tabel 13 10Link yang relevan dengan keyword “bandung diponegoro museum”

No.	Nama Link	Jumlah Skor
1	Museum:bandung	12

2	Museum:geologi	144
3	Travel:bandung	36
TOTAL		192

c. *Keyword* “bandung diponegoro museum geologi”

Keyword yang diinputkan menghasilkan 216 *path* dengan masing-masing *path* berjumlah 3, 4, dan 5 *node*. Berikut eksperimen yang dihasilkan berdasarkan *keyword* “bandung diponegoro museum geologi” dengan jumlah *node* pada *path* sebanyak 3 buah ditunjukkan pada Tabel 14.

Tabel 14 Hasil eksperimen *keyword* “bandung diponegoro museum geologi”

No	Path	\sum Node	\sum Inlink	\sum Outlink	Nilai Ranking	Urutan ke-
1	P145	3	-	-	-	1
2	P146	4	3	3	0.22263572969373216	2
3	P166	4	3	3	0.22263572969373216	3
4	P167	4	3	3	0.22263572969373216	4
5	P73	5	6	4	0.1618477996478465	5
6	P95	5	6	4	0.1618477996478465	6
7	P144	5	5	3	0.13414828444273008	7
8	P181	5	3	6	0.07835278212604194	8
9

Data detail *path* untuk hasil eksperimen ini ditunjukkan pada Lampiran B.

Berikut merupakan hasil *precision* dari *keyword* “bandung diponegoro museum” :

$$precision = \frac{\text{Total dari jumlah skor setiap link yang relevan}}{\text{Jumlah link yang muncul pada mesin pencari}}$$

$$= \frac{648}{1191} \times 100\% = 54.41\%$$

Berikut data detail mengenai *link* yang terambil oleh sistem dan *link* yang relevan dengan *user* ditunjukkan pada Tabel 15 dan Tabel 16.

Tabel 15 Data terambil dari *keyword* “bandung diponegoro museum geologi”

No.	Nama Link	Skor	Jumlah Link
1	Hotel:santika	0	72
2	Hotel:vio	0	72
3	Museum:affandi	0	72
4	Museum:bandung	1	36
5	Museum:brawijaya	0	72
6	Museum:category_geologi	1	72
7	Museum:geologi	1	432
8	Museum:satriamandala	0	72
9	Museum:serangga	0	72
10	Museum:sribaduga	0	75
11	Museum:wayang	0	36
12	Travel:bandung	1	108
TOTAL			1191

Tabel 16 Data yang relevan dengan “bandung diponegoro museum geologi”

No.	Nama Link	Jumlah Skor
1	Museum:bandung	36
2	Museum:category_geologi	72
3	Museum:geologi	432
4	Travel:bandung	108

TOTAL	648
-------	-----

2. *Keyword* yang memiliki konsep lebih dari 1 terdiri dari:

a. *Keyword* “kuta geologi”

Keyword ini menghasilkan 3 path yang semuanya berjumlah 7 dan 8 *node*. Berikut hasil eksperimen untuk *keyword* “kuta geologi”.

Tabel 17 Hasil eksperimen *keyword* “kuta geologi”

No	Path	\sum Node	\sum Inlink	\sum Outlink	Nilai Ranking	Urutan ke-
1	P1	7	16	30	0.20855746347861773	1
2	P2	7	16	30	0.20855746347861773	2
3	P0	8	-	-	-	3

Tabel 1811 Data yang terambil dari *keyword* “kuta geologi”

No.	Nama Link	Jumlah Link
1	Museum:category_geologi	1
2	Travel:KutaBeach	6
3	Museum:geologi	3
4	Travel:bali	6
5	Travel:bandung	6
TOTAL		22

b. *Keyword* “kuta geologi santika”

Keyword yang diinputkan adalah *keyword* yang berjumlah 3 term. Hasil eksperimen pada *keyword* ini menghasilkan *path-path* yang telah dipaparkan pada Tabel 19.

Tabel 19 Hasil eksperimen *keyword* “kuta geologi santika”

No	Path	\sum Node	\sum Inlink	\sum Outlink	Nilai Ranking	Urutan ke-
1	P0	9	19	38	0.2198506063368055	1
2	P1	9	19	38	0.2198506063368055	2
3	P2	10	-	-	-	3

Tabel 2012 Data yang terambil dari *keyword* “kuta geologi santika”

No.	Nama Link	Jumlah Link
1	Museum:category_geologi	1
2	Travel:KutaBeach	6
3	Hotel:santika	6
4	Museum:geologi	3
5	Travel:bali	6
6	Travel:bandung	6
TOTAL		28

c. *Keyword* “kuta geologi santikabrawijaya”

Keyword yang diinputkan adalah *keyword* yang berjumlah 4 term. Hasil eksperimen pada *keyword* ini menghasilkan *path-path* yang telah dipaparkan pada Tabel 21.

Tabel 2113 Hasil eksperimen *keyword* “kuta geologi santika brawijaya”

No	Path	\sum Node	\sum Inlink	\sum Outlink	Nilai Ranking	Urutan ke-
1	P2	12	38	50	0.14914613095238097	1
2	P3	12	38	50	0.14914613095238097	2
3	P4	12	38	50	0.14914613095238097	3

4	P5	12	38	50	0.14914613095238097	4
5	P0	13	37	51	0.2198506063368055	5
6	P1	13	37	51	0.2198506063368055	6

Tabel 22 14 Data yang terambil dari *keyword* “geologi kuta santika brawijaya”

No.	Nama <i>Link</i>	Jumlah <i>Link</i>
1	Museum:category_geologi	2
2	Travel:KutaBeach	12
3	Hotel:santika	12
4	Museum:geologi	6
5	Travel:bali	12
6	Travel:bandung	12
7	Museum:brawijaya	12
TOTAL		68

3. *Keyword* yang akan dijadikan alat eksperimen merupakan *keyword* yang menghasilkan konsep berupa sinonim dari *keyword* yang dicari terdiri dari :
- a. *Keyword* “jogja parangtritis”. Tabel 23 menunjukkan hasil eksperimen dari *keyword* “jogja parangtritis”.

Tabel 23 Hasil eksperimen *keyword* “jogja parangtritis”

No	<i>Path</i>	\sum <i>Node</i>	\sum <i>Inlink</i>	\sum <i>Outlink</i>	Nilai Ranking	Urutan ke-
1	P0	5	10	20	0.2	1
2	P1	5	10	20	0.2	2
3	P2	5	10	20	0.2	3

POLBAN

Tabel 24 Data yang terambil dari *keyword* “jogja parangtritis”

No.	Nama <i>Link</i>	Jumlah <i>Link</i>
1	Travel:yogyakarta	3
2	Travel:ParangtritisBeach	6
3	Museum:jogja	6
TOTAL		15

b. *Keyword* “yogyakarta affandi”

Tabel 78 menunjukkan hasil eksperimen dari *keyword* “yogyakarta affandi”.

Tabel 25 Hasil eksperimen *keyword* “yogyakarta affandi”

No	<i>Path</i>	\sum Node	\sum Inlink	\sum Outlink	Nilai Ranking	Urutan ke-
1	P0	3	5	10	0.24589430908066282	1
2	P1	3	5	10	0.24589430908066282	2
3	P2	5	9	20	0.25	3
4	P3	5	9	20	0.25	4
5	P4	6	10	27	0.25	5
6	P5	6	10	27	0.25	6

Tabel 26 15 Data yang terambil dari *keyword* “yogyakarta affandi”

No.	Nama <i>Link</i>	Jumlah <i>Link</i>
1	Travel:yogyakarta	4
2	Travel:ParangtritisBeach	4
3	Museum:jogja	8
4	Museum:affandi	11
TOTAL		27

4.1.2. Evaluasi Eksperimen

Berdasarkan Tabel 80, dari eksperimen yang telah dilakukan, berikut hasil keseluruhan dari eksperimen penelitian ini :

1. *Keyword* yang akan menghasilkan konsep yang lebih spesifik.
Keyword dengan konsep yang lebih spesifik berarti konsep yang *path* terpendeknya memiliki jumlah *node* sebanyak 3 *term*. *Term* tersebut terdiri dari 2 *node* sebagai *object literal* dan 1 *node* sebagai *subject* sehingga *path* tersebut memiliki 1 konsep.
2. *Keyword* yang memiliki konsep lebih dari 1.
Keyword dengan konsep lebih dari 1 adalah konsep yang *path* terpendeknya memiliki jumlah *node* lebih dari 3, yaitu 2 *object literal* dan 2 atau lebih *node* sebagai *subject* sehingga memiliki 2 atau lebih konsep.
3. *Keyword* yang akan dijadikan alat eksperimen merupakan *keyword* yang menghasilkan konsep berupa sinonim dari *keyword* yang dicari.

Tabel 27 Hasil eksperimen

	Keyword	Path	Jumlah Node	Inlink	Outlink	Ranking
Eksperimen 1	"bandung diponegoro"	P4	3	-	-	-
		P2	4	3	2	0.2059451525671121
		P5	4	2	3	0.1604668483012126
	bandung diponegoro museum	P0	5	4	4	0.17848533012843273
		P1	5	4	4	0.17848533012843273
		P3	5	2	2	0.10132773492641256
		P48	3	-	-	-
		P55	4	-	-	-
		P24	5	5	3	0.1664658313165932
		P31	5	5	3	0.1664658313165932
		P60	5	3	5	0.10197998084131395
		P67	5	3	5	0.10197998084131395
		P36	6	14	14	0.0571036488791345
	
	bandung diponegoro museum geologi	P145	3	-	-	-
		P146	4	3	3	0.22263572969373216
		P166	4	3	3	0.22263572969373216
		P167	4	3	3	0.22263572969373216
		P73	5	6	4	0.1618477996478465
		P95	5	6	4	0.1618477996478465
		P144	5	5	3	0.13414828444273008
		P181	5	3	6	0.07835278212604194
	
Eksperimen 2	kuta geologi	P1	7	16	30	0.20855746347861773
		P2	7	16	30	0.20855746347861773
		P0	8	-	-	-
	kuta geologi santika	P0	9	19	38	0.2198506063368055
		P1	9	19	38	0.2198506063368055
		P2	10	-	-	-
	kuta geologi santika brawijaya	P2	12	38	50	0.14914613095238097
		P3	12	38	50	0.14914613095238097
		P4	12	38	50	0.14914613095238097
		P5	12	38	50	0.14914613095238097
		P0	13	37	51	0.2198506063368055
		P1	13	37	51	0.2198506063368055
	
Eksperimen 3	jogja parangtritis	P0	5	10	20	0.2
		P1	5	10	20	0.2
		P2	5	10	20	0.2
	yogyakarta affandi	P0	3	5	10	0.24589430908066282
		P1	3	5	10	0.24589430908066282
		P2	5	9	20	0.25
		P3	5	9	20	0.25
		P4	6	10	27	0.25
		P5	6	10	27	0.25
	

Berdasarkan hasil eksperimen yang didapatkan, maka dapat diketahui bahwa :

1. Semakin banyak *term* yang diinputkan, maka semakin banyak *path* yang terbentuk.
2. Nilai ranking suatu *path* dipengaruhi oleh jumlah *inlink* pada *path* tersebut. Semakin banyak jumlah *inlink* maka nilai ranking suatu *path* semakin besar dan semakin sedikit jumlah *inlink*nya maka nilai ranking *path* semakin kecil.

3. Apabila nilai *inlink* antarpath sama namun nilai *outlinknya* berbeda, maka nilai ranking suatu *path* lebih besar jika jumlah *outlinkpath* tersebut lebih sedikit. Begitu juga sebaliknya, jika jumlah *outlink* suatu *path* lebih banyak, maka nilai rankingnya lebih kecil.

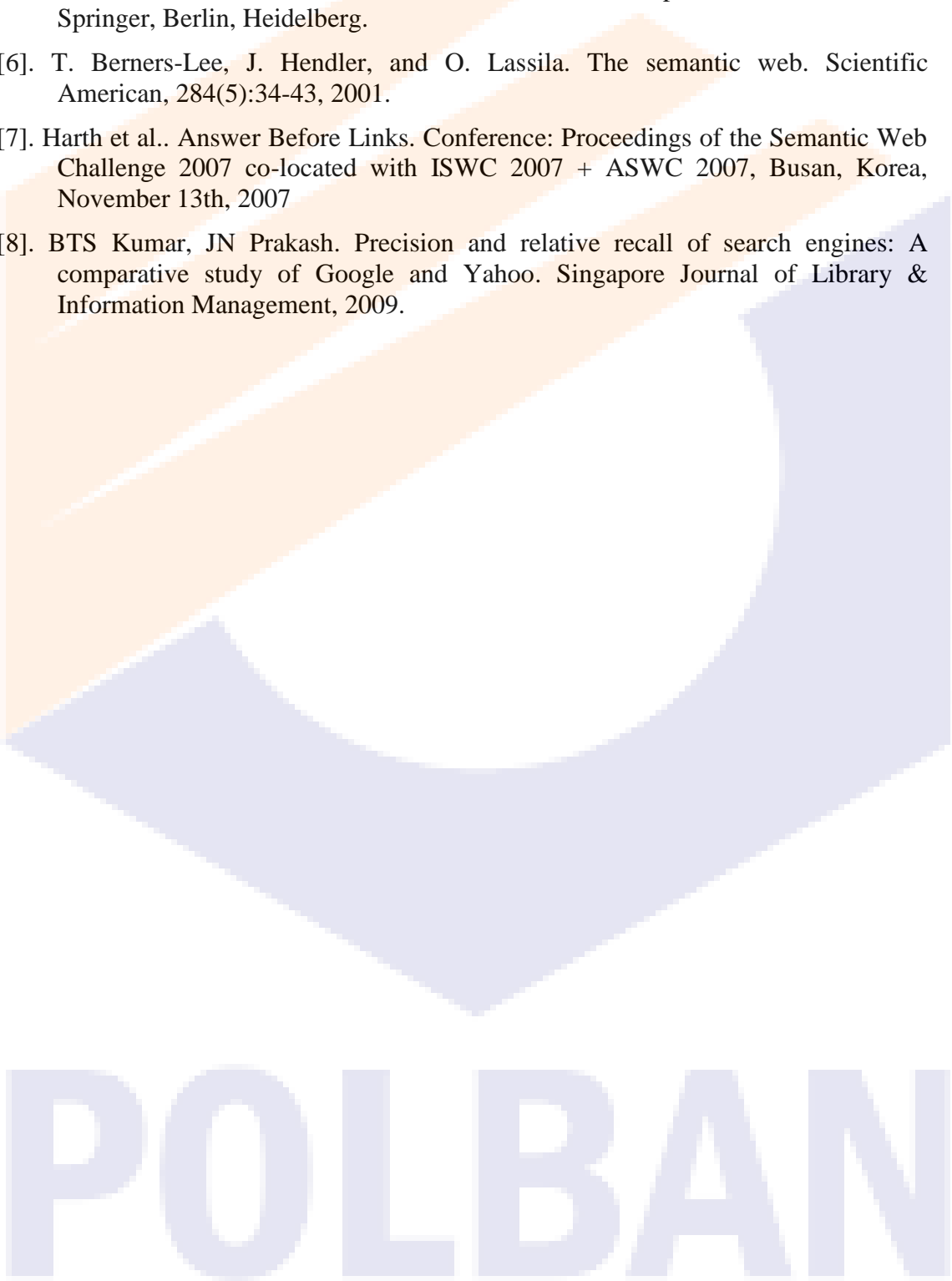
Dari hasil eksperimen diatas, dapat disimpulkan bahwa algoritma ReConRank dapat diperluas dari *query single term* menjadi *query multi term* menggunakan *shortest path* sehingga hipotesa dapat diterima.

Hasil kepresisian yang didapatkan menentukan tingkat keakuratan dari hasil pencarian terhadap *query* yang diinputkan. Berikut hasil penentuan tingkat keakuratan yang telah dilakukan pada tahapan eksperimen :

1. Setelah dilakukan penghitungan keakuratan pada *keyword* yang memiliki konsep yang lebih spesifik, maka secara keseluruhan perolehan hasil keakuratan pada algoritma ReConRank *multiterm* ini diatas 50%.
2. Untuk eksperimen pada *keyword* yang memiliki konsep lebih dari 1, path terpendek hasil pencarian memiliki *node* banyak (5 atau lebih) sehingga korelevannya diragukan, tidak seperti pada *keyword* dengan konsep lebih spesifik yang *path* terpendeknya memiliki jumlah *node* sedikit yaitu 3 (terdiri dari 1 *node* URI dan 2 *nodeobject literal*). Pada eksperimen ini tidak dilakukan penghitungan keakuratan karena *link* yang relevan tidak dapat ditentukan. Pada dasarnya, *keyword* yang *path* terpendeknya memiliki konsep lebih dari 1 ini merupakan *keyword* yang kurang relevan sehingga tidak bermakna. *Keyword* yang bermakna menentukan apakah *link* yang dihasilkan relevan atau tidak.
3. Eksperimen dengan konsep sinonim dari *keyword* ”jogja parangtritis” menghasilkan *path* dengan *node* yang terdiri dari sinonim kata ”jogja” yaitu ”yogyakarta”. Pada *dataset* yang ada, kata ”parangtritis” berhubungan dengan ”yogyakarta” sehingga secara tidak langsung ”jogja” juga akan berhubungan dengan ”yogyakarta”. *Keyword* sinonim digunakan sebagai alat eksperimen untuk memastikan bahwa aplikasi dapat menampilkan kata lain yang semakna dari *keyword* yang user inputkan. Sinonim yang terdapat pada data penelitian hanya berjumlah 2 kata sinonim, namun dapat merepresentasikan fungsi sinonim terhadap *query* yang diinputkan.

Daftar Pustaka

- [1]. Urip T. Setijohatmo and Setiadi Rachmat, "Pemrosesan Query Multi Terms Pada Mesin Pencari Web Semantik," IRWNS, Bandung, 2014
- [2]. <https://www.w3.org/2003/Talks/0624-BrusselsSW-IH/>
- [3]. Aidan Hogan, Andreas Harth, Stefan Decker "ReConRank: A Scalable Ranking Method for Semantic Web Data with Context", Proceedings of Second International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS 2006)
- [4]. Antoniou, G. (Grigoris), ... [et al.]. A Semantic Web primer – 3rd ed.p. cm. The MIT PressCambridge, MassachusettsLondon, England-ISBN 978-0-262-01828-9

- 
- [5]. Zhou Q., Wang C., Xiong M., Wang H., Yu Y. (2007) SPARK: Adapting Keyword Query to Semantic Search. In: Aberer K. et al. (eds) The Semantic Web. ISWC 2007, ASWC 2007. Lecture Notes in Computer Science, vol 4825. Springer, Berlin, Heidelberg.
 - [6]. T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. Scientific American, 284(5):34-43, 2001.
 - [7]. Harth et al.. Answer Before Links. Conference: Proceedings of the Semantic Web Challenge 2007 co-located with ISWC 2007 + ASWC 2007, Busan, Korea, November 13th, 2007
 - [8]. BTS Kumar, JN Prakash. Precision and relative recall of search engines: A comparative study of Google and Yahoo. Singapore Journal of Library & Information Management, 2009.