Aplikasi *Web crawler* Berdasarkan *Breadth First Search* dan *Back-Link*

Sulastri dan Eri Zuliarso
Fakultas Teknologi Informasi, Universitas Stikubank Semarang
email : striq@unisbank.ac.id, eri@unisbank.ac.id

Abstrak

Web crawler, juga sering dikenal sebagai *Web Spider* atau *Web Robot* adalah salah satu komponen penting dalam sebuah mesin pencari modern. Fungsi utama *Web crawler* adalah untuk melakukan penjelajahan dan pengambilan halaman-halaman Web yang ada di Internet. Pada tulisan ini akan disajikan ujicoba perbandingan algoritma penelusuran program *crawler* menggunakan *Breadth First Search* dan banyaknya *backlink* (*Backlink Count*). Pengujian berdasarkan situs *www.dmoz.org* dan dir.yahoo.com.

Kata kunci : *Crawler*, *Web*

PENDAHULUAN

Pertumbuhan *World Wide Web* yang eksplosif membuat sukar menemukan informasi yang sesuai dengan keinginan pemakai. Terlalu banyak server dan halaman yang harus dilihat dan dilakukan secara on line tetap merupakan tugas yang mengkonsumsi waktu. Hal inilah yang disebut masalah penemuan sumberdaya internet (*internet resource discovery problem*).

Web crawler, juga sering dikenal sebagai Web Spider atau Web Robot adalah salah satu komponen penting dalam sebuah mesin pencari modern. Fungsi utama *Web crawler* adalah untuk melakukan penjelajahan dan pengambilan halaman-halaman Web yang ada di Internet. Hasil pengumpulan situs Web selanjutnya akan diindeks oleh mesin pencari sehingga mempermudah pencarian informasi di Internet.

Mendesain sebuah *crawler* yang baik saat ini menemui banyak tantangan. Secara eksternal, *crawler* harus mengatasi besarnya situs Web dan link jaringan. Secara internal , *crawler* harus mengatasi besarnya volume data. Sehubungan dengan terbatasnya sumber daya komputasi dan keterbatasan waktu, maka harus hati-hati memutuskan URL apa yang harus di scan dan bagaimana urutannya. *Crawler* tidak dapat mengunduh semua halaman web. Penting bagi *crawler* untuk memilih halaman dan mengunjungi halaman yang penting dulu dengan memprioritaskan URL yang penting tersebut dalam antrian. *Crawler* juga harus memutuskan berapa frekuensi untuk merevisi halaman yang pernah dilihat, untuk memberikan informasi ke client perubahan yang terjadi di Web.

Dalam penelitian ini telah diteliti tantangan bagaimana memprioritaskan antrian URL untuk menelusuri halaman yang lebih relevan berdasarkan *Breadth First Search* dan banyaknya *Backlink*.

Dasar dari *Web crawler*

Walaupun banyak aplikasi untuk *Web crawler*, pada intinya semuanya secara fundamental sama. Berikut ini proses yang dilakukan *Web crawler* pada saat bekerja :

1. Mengunduh halaman Web.
2. Memparsing halaman yang didownload dan mengambil semua link.
3. Untuk setiap link yang diambil, ulangi proses.

Dalam langkah pertama, sebuah *web crawler* mengambil URL dan mengunduh halaman dari Internet berdasarkan URL yang diberikan. Seringkali halaman yang diunduh disimpan ke sebuah file atau ditempatkan di basisdata. Dengan menyimpan halaman web, maka *crawler* atau program yang lain dapat memanipulasi halaman itu untuk diindeks (dalam kasus mesin pencari) atau untuk

pengarsipan untuk digunakan oleh pengarsip otomatis.

Tahap kedua, *Web crawler* memparsing keseluruhan halaman yang diunduh dan mengambil link-link ke halaman lain. Tiap link dalam halaman didefinisikan dengan sebuah penanda HTML yang serupa dengan yang ditunjukkan disini :

```
<A
HREF="http://www.host.com/directory/file.html
">Link</A>
```

Setelah *crawler* mengambil link dari halaman, tiap link ditambahkan ke sebuah daftar untuk dicrawler.

Langkah ketiga dari Web crawling adalah mengulangi proses.Semua *crawler* bekerja dengan rekursif atau bentuk perulangan, tetapi ada dua cara berbeda untuk menanganinya. Link dapat dicrawl dalam cara *depth-first* atau *breadth-first*.

Breadth-first crawling menguji tiap link pada sebuah halaman sebelum memproses ke halaman berikutnya. Jadi, algoritma ini menelusuri tiap link pada halaman pertama dan kemudian menelusuri tiap link pada halaman pertama pada link pertama dan begitu seterusnya sampai tiap level pada link telah dikunjungi.

Penelusuran menggunakan *backlink*, memprioritaskan penelusuran berdasarkan banyaknya *backlink* di suatu halaman atau ***Backlink Count***. ***Backlink Count*** adalah banyaknya gadengan (link) ke *p* yang muncul di seluruh Web. Secara intuitif, sebuah halaman *p* yang dilink oleh banyak halaman lebih penting daripada halaman yang jarang menjadi referensi. Tipe "penghitungan kutipan" telah banyak digunakan secara luas untuk mengevaluasi pengaruh dari makalah yang dipublikasikan. Di Web, *Backlink Count* bermanfaat untuk merangking hasil *query*, dan memberikan pemakai halaman yang secara umum menjadi perhatian. Sebagai catatan untuk mengevaluasi *Backlink Count* membutuhkan penghitungan *backlink* di keseluruhan Web. Sebuah *crawler* dapat memperkirakan harga *Backlink Count,* yaitu banyaknya link ke *p* berdasarkan kunjungan yang telah dilakukan ke halaman web.

Evaluasi *Crawler*

Tujuan untuk mendesain *crawler* adalah jika mungkin mengunjungi halaman dengan nilai penting tinggi sebelum halaman dengan nilai penting yang lebih rendah. Adapun model pengujian suatu *crawler* adalah sebagai berikut :

- Dalam melakukan penelusuran, *crawler* menggunakan model ***Crawl* dan *Stop***. Dalam model ini, *crawler* C mulai pada halaman awal p_o dan berhenti setelah mengunjungi *K* halaman. Pada titik ini sebuah *crawler* yang sempurna harus sudah mengunjungi halaman $r_1,...,r_K$ dimana r_1 adalah halaman dengan harga terpenting, r_2 yang penting berikutnya, begitu berikutnya. Halaman $r_1,...,r_K$ disebut halaman "*hot*". *K* halaman yang akan dikunjungi oleh *crawler* pada penelitian ini hanya akan memuat *M* halaman dengan rangking lebih tinggi atau sama dengan $I(r_K)$. Kinerja *crawler C* didefinisikan sebagai $P_{CS}(C)= M/K$.. Kinerja dari *crawler* yang ideal dikenal metrik ***harvest-rate :***

$$hr = \frac{\#r}{\#p}, hr \in [0,1] \quad \text{.....(1)}$$

Harvest-rate merepresentasikan pecahan halaman Web yang ditelusuri yang memenuhi target **#r** dalam keseluruhan halaman **#p** yang diperoleh.Rasio harvest-rate ini harus tinggi, jika tidak *crawler* akan banyak meluangkan waktu untuk mengeliminasi halaman yang tidak relevan.

- **Crawl and Stop with Threshold**.

  Diasumsikan bahwa *crawler* mengunjungi *K* halaman. Namun demikian, *crawler* diberi target penting *G*, dan sembarang halaman dengan $I(p) \geq G$ termasuk sebagai halaman "hot". Diasumsikan bahwa total banyaknya halaman yang hot adalah *H*. Kinerja dari *crawler*, $P_{ST}(C)$, perbandingan halaman hot *H* dengan *K* halaman yang dikunjungi ketika crawl telah berhenti. Jika $K < H$, maka *crawler* yang ideal memiliki kinerja K / H. Jika $K \geq H$, maka *crawler* yang ideal mempunyai kinerja yang sempurna 1.

Adapun algoritma crawling adalah sebagai berikut :

```
Input: starting_url: seed URL
Procedure:
  [1] enqueue(url_queue, starting_url)
  [2] while (not empty(url_queue))
  [3]   url = dequeue(url queue)
  [4]   page = crawl_page(url)
  [5]   enqueue(crawled_pages, (url, page))
  [6]   url list = extract urls(page)
  [7]   foreach u in url_list
  [8]     enqueue(links, (url, u))
  [9]     if (u∉url_queue and (u,-)∉crawled_pages)
  [10]      enqueue(url_queue, u)
  [11]  reorder_queue(url_queue)

Function description:
  enqueue(queue, element): append element at the end of queue
  dequeue(queue)         : remove the element at the beginning
                           of queue and return it
  reorder_queue(queue)   : reorder queue using information in
                           links (refer to Figure 2.2)
```

Deskripsi fungsi :

enqueue(queue, element) :

menambahkan elemen di ujung dari queue

dequeue(queue) :

menghilangkan elemen di awal queue dan memberikan ke program yang memanggil

reorder_queue :

mengurutkan queue menggunakan informasi yang ada di tautan.

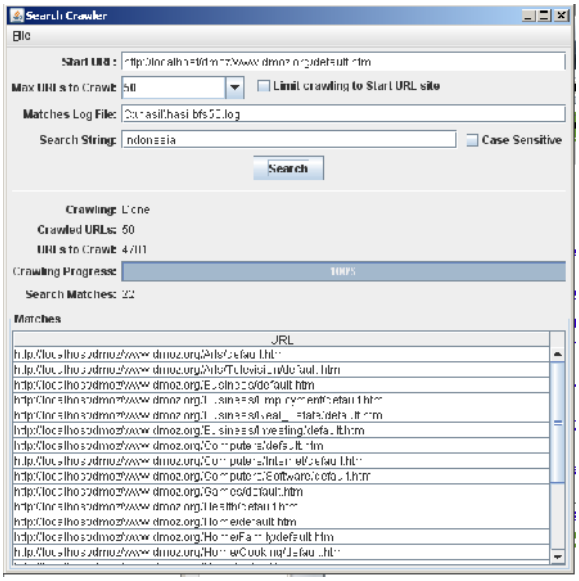
Secara umum, *crawler* akan diberi URL awal untuk melakukan penelusuran,kata kunci yang dicari dan banyaknya halaman yang mengandung kata kunci . Apabila dalam halaman web ditemukan kata kunci, maka halaman web itu akan disimpan. Crawling akan berakhir apabila banyaknya halaman web yang mempunyai kata kunci sudah sama dengan batas halaman banyaknya halaman yang mengandung kata kunci. Atau crawling akan berhenti apabila sudah tidak ada lagi halaman yang akan dicrawling.

Pada saat crawling menggunakan pengurutan *breadth first*, maka pada saat mengunjungi halaman web, setiap *hyperlink* yang ada diuji apakah sudah ada dalam list atau belum. Untuk link yang belum dikunjungi ditambahkan ke list.

Saat melakukan crawling menggunakan pengurutan berdasarkan *backlink*, maka pada saat mengunjungi halaman web, setiap *hyperlink* yang ada dilihat merujuk ke halaman web yang mana. Kalau halaman web yang dirujuk belum ada di list, maka halaman tersebut dimasukkan ke dalam list dan *backlink*nya diberi nilai 1. Jika dalam list sudah ada dan belum dikunjungi maka *backlink*-nya ditambah satu. Sesudah seluruh *hyperlink* dalam halaman itu diperiksa, kemudian list diurutkan berdasarkan banyaknya *backlink*.

Tampilan program

Search *Crawler* adalah *crawler* Web dasar untuk melakukan pencarian Web. Search *Crawler* mempunyai tiga bagian utama yaitu *Search*, *Stats* dan *Matches*. Bagian *Search* pada bagian jendela atas mempunyai kendali untuk memasukkan kriteria pencarian, termasuk didalamnya URL untuk pencarian, jumlah maksimum URL yang akan dicrawl, dan kata yang akan dicari.



Gambar 1. *Search Crawler*

Kriteria pencarian dapat ditambah dengan memilih batas pencarian ke situs pada awal URL dan dengan memilih *check box case sensitive* untuk pencarian string.

Bagian *Stats*, yang berlokasi di tengah jendela, mempunyai control yang menunjukkan status crawling yang sedang berjalan ketika pencarian sedang berjalan. Bagian ini juga mempunyai progress bar untuk mengindikasikan progress menuju selesainya pencarian. Bagian *Matches* yang ada dibawah jendela mempunyai table yang memperlihatkan semua kecocokan yang ditemukan dalam pencarian. Daftar ini berisi URL dari halaman Web yang memuat string yang dicari.

Untuk menghitung relevansi suatu halaman web dengan kata kunci dihitung

menggunakan rumus similaritas tekstual. Karena tidak memungkinkan untuk mengumpulkan seluruh halaman web, maka dibuat asumsi jika dalam suatu halaman web terdapat kata yang sesuai dengan kata kunci, maka halaman tersebut dianggap relevan dengan query pemakai.

Sebagai perbandingan dicoba juga *crawler* dengan similaritas tekstual murni, jadi akan diuji berdasarkan pencocokan suatu kata yang dimasukkan dengan suatu kata di halaman web.

Hasil eksekusi program

Untuk menguji program crawl, maka dilakukan crawling pada dua situs directory di internet yaitu *www.dmoz.org* dan dir.yahoo.com. Web Directory atau link directory adalah sebuah directory di World Wide Web. Directory menspesialisasikan dalam linking (me-link) ke situs web yang lain dan mengkategori link-link tersebut.

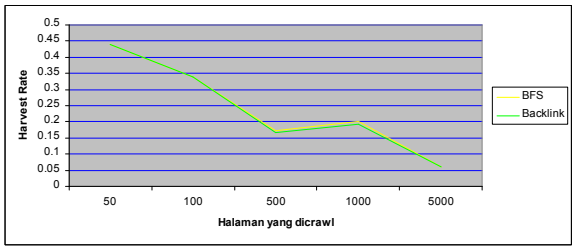
Web directory bukanlah search engine (mesin pencari) dan tidak menampilkan daftar halaman web berdasarkan pada kata kunci; tetapi berdasarkan kategori dan subkategori. Pengkategorian biasanya didasarkan pada keseluruhan halaman web bukan berdasarkan satu halaman atau pada himpunan kata kunci, dan situs dibatasi khusus pada sedikit kategori.

Open Directory Project (ODP), juga dikenal sebagai **Dmoz** (dari *directory.mozilla.org*, yang merupakan domain name awal), adalah *open content directory* multibahasa dari link-link *World Wide Web* yang dimiliki oleh Netscape yang dikonstruksi dan dipelihara oleh komunitas editor suka rela. ODP menggunakan skema hirarki ontology untuk mengorganisir daftar situs. Daftar pada topic yang serupa dikelompokkan menjadi kategori yang mencakup kategori yang lebih kecil.

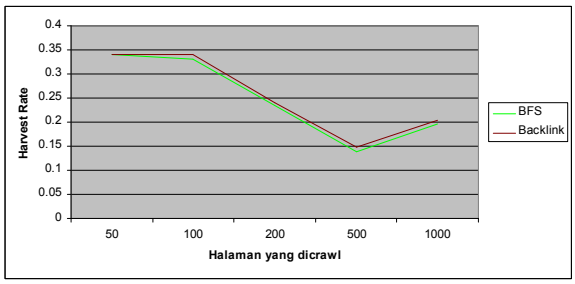
Yahoo!Directory adalah *web directory* yang menjadi pesaing *Open Directory Project* dalam besar link yang tercatat. *Directory Yahoo!* Lahir terlebih dahulu daripada ODP. Ketika Yahoo! Berubah ke *crawler-based listing* pada Oktober 2002, maka directory yang diedit manusia berkurang banyak.

Untuk melakukan pengujian program, pada saat melakukan crawling pada *www.dmoz.org* digunakan kata kunci Indonesia, sedangkan pada dir.yahoo.com digunakan kata kunci Computer.

Adapun perbandingan kinerja crawl keseluruhan ditampilkan dalam grafik di bawah ini. Harvest-rate adalah perbandingan banyaknya halaman yang mengandung kata kunci dengan banyaknya halaman yang dicrawl.



Gambar 2. Hasil pengujian dengan kata kunci Indonesia di *www.dmoz.org*



Gambar 3. Hasil pengujian dengan kata kunci Computer di dir.yahoo.com

Dari hasil percobaan, didapatkan bahwa menggunakan BFS dan *backlink*, banyaknya halaman yang sesuai berbanding terbalik dengan banyaknya halaman yang dicrawl. Semakin banyak halaman yang dicrawl, kinerja akan semakin menurun. Dari perbandingan crawl di *www.dmoz.org*, maka BFS sedikit lebih baik dibandingkan dengan menggunakan *backlink*. Sedang di dir.yahoo.com, pengurutan menggunakan *backlink* lebih baik dari pada menggunakan BFS.

KESIMPULAN

Dalam penelitian ini dapat disimpulkan beberapa hal sebagai berikut :

1. Telah dibuat program crawl baik dengan metode *breadth first search* dan metode pengurutan *backlink*.
2. Program crawl telah diuji coba dan berjalan dengan baik pada *web directory*

www.dmoz.org dan *dir.yahoo.com*

3. Dari perbandingan crawl di *www.dmoz.org*, maka BFS sedikit lebih baik dibandingkan dengan menggunakan *backlink*. Sedang di *dir.yahoo.com*, pengurutan menggunakan *backlink* lebih baik dari pada menggunakan BFS. Hal ini kemungkinan didasarkan pada cara membuat struktur *directory*. Di ODP, struktur dibuat secara manual oleh para relawan. Sedangkan di *dir.yahoo.com* menggunakan program crawl otomatis. Sehingga dimungkinkan bahwa dalam pembuatan *directory dir.yahoo.com* memperhitungkan banyaknya *backlink*. Sedangkan di ODP lebih memperhitungkan struktur pengetahuan yang umum menggunakan ontology.

DAFTAR PUSTAKA

Antoniou, G.,F.v. Harmelen, 2008, *A Semantic Web Primer 2nd Ed.*, MIT Press

Budi Yuwono, Savio L. Y. Lam, Jerry H. Ying, Dik L. Lee, 2006, "A World Wide Web Resource Discovery System", *In Proceedings of ICDE*

G.A. Miller, 1995, "WORDNET : A Lexical Database for English", *Communication ACM, Vol 2, No. 11, Nov. 1995, pp 39-41.*

G. Pant, P. Srinivasan, F. Menczer, "*Crawling the Web",*

Gomez-Perez, A., O. Corcho, 2002, Ontology "Languages for the Semantic Web"*, IEEE Intelligent,* January / February

Gruber, T., *What is an Ontology?,* http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

Junghoo Cho, Hector Garcia-Molina, and Lawrence Page, 1998, **"**Efficient crawling through URL ordering**",** *In Proceedings of the Seventh International World Wide Web Conference*, pages 161--172, April

Marc Ehrig and Alexander Maedche, 2003, "Ontology-Focused Crawling of Web Documents"*, In Proceedings of Symposium on Applied Computing,Florida, USA*

Nicola Guarino, Claudio Masolo, Guido Vetere, 1999, "OntoSeek: Content-Based Access to the Web", *IEEE Intelligents System*

Open Directory Project, http://www.dmoz.org

P Srinivasan, F. Menczer, G. Pant, 2004, "A General Evaluation Framework for Topical Crawlers"*, Kluwer Academic Publishers*

S. Chakrabarti, M. van den Berg, and B. Dom, 1999, "Focused crawling: a new approach to topic-specific Web resource discovery", *Computer Networks (Amsterdam, Netherlands),* vol 31, pp. 1623-1640, Available : citeseer.ist.psu.edu/chakrabarti99focused.html

Sergey Brin and Lawrence Page, 1998, **"**The anatomy of a large-scale hypertextual Web search engine**",** *In Proceedings of the Seventh International World Wide Web Conference*, pages 107--117, April 1998

S. Ganesh, M. Jayaraj, V. Kalyan Srinivasal Murthy, G. Aghila, 2004, "Ontology-based Web crawler"*, In Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04), IEEE*