ANALISA CAPAIAN WEB CRAWLER DENGAN MENGGUNAKAN ALGORITMA GENETIK

ALI SELAMAT[1] & LIM YEE WAY[2]

Abstrak. Dengan perkembangan teknologi maklumat yang kian mendadak, proses carian maklumat dalam internet menjadi satu kerja yang susah dan memakan masa. Enjin carian seperti *Google, LookSmart, Altavista* dan *Yahoo!* Telah diperkenalkan dan digunakan untuk memudahkan proses carian maklumat dalam internet. *Web crawler* adalah komponen yang sangat penting kepada enjin carian dan digunakan sebagai agen untuk mencari sumber maklumat dalam internet. Namun, *web crawler* menghadapi beberapa masalah seperti keputusan carian yang dijanakan olehnya adalah kurang tepat, tidak terkini, masalah limpahan makumat (*overload)* dalam penyimpanan maklumat yang diperolehi oleh agen ke dalam pangkalan data dan tidak mempunyai perhubungan antara pengguna dengan *web crawler*. Selain itu, *web crawler* tidak dapat mencapai laman web tertentu. Oleh sebab itu, satu *Web Crawler* pintar yang bernama *UtmCrawler* diperkenalkan untuk mengatasi dua masalah utama iaitu masalah *overload* dalam penyimpanan maklumat ke dalam pangkalan data dan keputusan carian yang dijanakan adalah kurang tepat. *UtmCrawler* akan mengembangkan kata kunci carian yang dimasukkan oleh pengguna dengan menggunakan teknik agoritma genetik (GA) supaya hasil carian yang dijana oleh *UtmCrawler* mempunyai nilai *precision* yang tinggi. Proses carian *UtmCrawler* dilakukan tanpa menggunakan latihan dan *Relevance Feedback* (RF) bagi mencapai keputusan carian yang berkesan. Kesimpulannya, dengan adanya *UtmCrawler* ini pengguna akan mendapatkan maklumat yang lebih tepat.

Kata Kunci: Web crawler, Algoritma genetik, Search engine, Agen, Precision

Abstract. The World Wide Web (WWW) had tremendous growth in size and changing information source. Its growth and change rates make the task of finding relevant and recent information harder. Search engines such as *Google, LookSmart, Altavista and Yahoo!* are needed for people to search through the web. Web Crawler is one of the most crucial components in search engines and it used to crawl all resources or information. However, web crawler is unable to find the page which is lies in the deep or invisible web. Besides that, the current search engines do not have the communication capabilities between the web crawler and the users who dispatched the crawler. Almost the result of finding is outdated. The main problem of the web crawler is the results of finding are imprecise. Information overload also has become a serious problem to the users. Therefore, we introduce a UtmCrawler which is an intelligent web crawler in order to solve the two problems. The UtmCrawler is a crawler that expands its initial keywords by using a genetic algorithm during the crawling which has a higher precision of search results. All retrieval processes done by the UtmCrawler are not based on a relevance feedback from the user. As a conclusion we have found that the UtmCrawler has been beneficial to the internet user.

Keywords: Web crawler, Genetic algorithms, Search engine, Agent, Precision

---

[1&2] Faculty of Computer Science and Information Systems (FSKSM), Universiti Teknologi Malaysia (UTM), Skudai, 81310, Johor, Malaysia,Tel: +607-55 32099/32211/32212 (ext.), Fax: +607-5532210, Email: aselamat@fsksm.utm.my, yee_way@yahoo.com

1.0 PENGENALAN

Dengan perkembangan teknologi maklumat yang kian mendadak, dunia semakin menuju kepada era ledakan maklumat. Pelbagai maklumat daripada seluruh pelusuk dunia dapat dicapai hanya di hujung jari sahaja dengan mudah, pantas dan terkini. Bilangan laman web yang dijumpai dalam internet adalah seribu juta dengan penambahan sebanyak 1.5 juta laman web setiap hari [1]. Oleh sebab sumber maklumat yang luas, maka proses carian maklumat dalam internet telah menjadi satu kerja yang susah dan memakan masa [2].

Enjin carian (*serch engine*) seperti *Google* [3], *Altavista* [4], *Yahoo!* [5] dan sebagainya telah diperkenalkan dan digunakan bagi memudahkan proses carian maklumat dalam internet. *Web crawler* adalah komponen yang sangat penting kepada enjin carian [6] dan ianya digunakan sebagai agen untuk mencari dan mengumpul sumber maklumat dalam internet. Namun, proses carian menggunakan *web crawler* menghadapi masalah *overload* [15, 16] di dalam penyimpanan maklumat yang diperolehi untuk disimpan di dalam pangkalan data [7] engine carian. Selain itu, keputusan carian yang dijanakan oleh *web crawler* juga kurang tepat [8]. Oleh sebab itu, *UtmCrawler* diperkenalkan di dalam kertaskerja ini bagi mengatasi masalah tersebut. Kertaskerja ini dibahagikan kepada beberapa bahagian seperti dibawah:

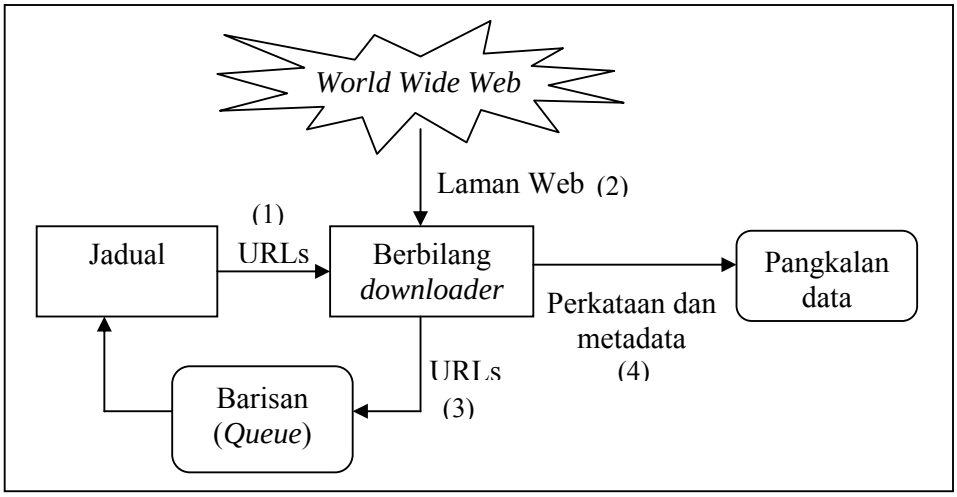
Bahagian 2 adalah kajian literatur tentang *web crawler*, teknik algoritma genetik (GA) dan topik yang berkaitan dengannya. Bahagian 3 akan membincangkan senibina *UtmCrawler* yang akan dibangunkan dan bahagian 4 akan menjelaskan metodologi yang akan digunakan untuk membangunkan prototaip *UtmCrawler*. Perbincangan dan keputusan dibincang pada bahagian 5.

2.0 KAJIAN LITERATUR

2.1 Pengenalan kepada *Web Crawler*

Web crawler merupakan satu agen yang melakukan operasinya di dalam internet. *Web crawler* juga merupakan satu elemen yang sangat penting kepada enjin carian tertentu. Enjin carian akan menggunakan maklumat yang diperolehi daripada *web crawler* untuk membina satu pangkalan data. *Web crawler* juga dikenali sebagai *web spider*, *web robot* atau *worm* yang digunakan untuk menguruskan dan mengumpul maklumat

bagi pangkalan data sesuatu enjin carian. Seperti yang ditunjukkan dalam Rajah 1, *web crawler* akan menjelajah dari satu pelayan ke pelayan yang lain untuk mendapatkan sumber maklumat yang diperlukan. Proses carian web crawler adalah ditandakan dengan turutan (1) hingga (4) di dalam Rajah 1.



Rajah 1: Senibina *web crawler* berperingkat tinggi [9]

2.2 Pengenalan kepada Algoritma Genetik

Algoritma Genetik (GA) adalah teknik pencarian tegap yang telah terbukti sebagai satu kaedah yang baik untuk penyelesaian masalah pengoptimaan yang susah. Ia mula diperkenalkan oleh Holland pada tahun 1975 [4]. Nama algoritma genetik berasal dari analogi di antara struktur kompleks melalui vektor komponen and struktur genetik pada kromosom.

GA adalah berprinsip kepada ciri-ciri penurunan dan evolusi. Dalam proses GA, individu ditakrifkan sebagai potensi cara penyelesaian. Kualiti sesuatu individu dinilaikan dengan menggunakan satu fungsi yang dikenali sebagai *fitness function*. Setiap generasi akan menghasilkan generasi yang baru melalui proses *selection,* proses *crossover* dan proses *mutation* [10]. Setiap generasi yang baru dihasilkan merupakan individu yang lebih berkualiti berbanding dengan generasi nenek moyangnya.

Di samping itu, GA juga digunakan untuk mencari cara penyelesaian yang paling optima bagi masalah tertentu. Contohnya, penggunaan GA dalam carian laluan terpendek bagi perjalanan bus, penggunaan teknik GA untuk menyelesaikan masalah penjadualan dan lain-lain [11]. Teknik GA akan cuba mendapatkan cara penyelesaian yang paling optima di kalangan cara-cara penyelesaian sesuatu masalah.

2.3 Fitness Function

Fitness function adalah satu fungsi GA yang digunakan untuk menentukan pelaksanaan atau *fitness* setiap individu dalam *population*. Individu yang mempunyai nilai *fitness* yang tinggi akan mempunyai kebarangkalian yang tinggi dipilih untuk proses *selection*. Contoh-contoh *fitness function* adalah seperti pekali *Jaccard*, *cosine similarity*, *Haming distance* [4] dan lain-lain.

$$fitness(d_j) = \frac{1}{n} \bullet \sum_{k=1}^{n} \left[\frac{|\, d_j \cap d_q \,|}{|\, d_j \cup d_q \,|} \right] \qquad \textbf{(1)}$$

Merujuk kepada formula **(1)** adalah pekali *Jaccard* di mana d_j adalah dokumen j, n adalah bilangan katakunci yang terdapat dalam dokumen j, k adalah satu perwakilan nombor dan d_q adalah dokumen q.

Cosine similarity adalah satu ukuran yang mengukur kesamaan dua vektor yang berdimensi n dengan mencari sudut antaranya. Diberi dua vektor A dan B, *Cosine similarity* (θ) dikira seperti yang ditunjukkan dalam formula **(2)** dengan menggunakan hasil *dot product* dan *magnitude* seperti dibawah:

$$\theta = \arccos \frac{A * B}{|\, A \,| * |\, B \,|} \qquad \textbf{(2)}$$

Hamming distance antara dua string yang mempunyai kepanjangan yang sama adalah jumlah perbezaan simbol antara dua string tersebut. Contohnya, *Hamming distance* antara 10111101 dan 10010101 adalah 2, *Hamming distance* antara 2143896 dan 2233796 adalah 3.

2.4 Strategi *Crawling*

Kecekapan *web crawler* untuk mencari sumber maklumat dalam internet bergantung kepada penggunaan algoritma carian yang sesuai. Kebanyakkan penyelidik menggunakan algoritma carian yang berjenis graf di mana mempertimbangkan laman web sebagai nod (*node)* dan perhubungannya sebagai sisi (*edge*). Secara umumnya, terdapat dua jenis algoritma carian berjenis graf iaitu carian *uninformed search* dan carian *informed search* [10].

Carian menggunakan teknik *uniformed search* merupakan algoritma yang mudah dan juga dikenali sebagai carian buta kerana ia melakukan carian tanpa melibatkan sebarang maklumat yang mengenai tentang keadaan semasa untuk bergerak dari satu nod ke satu nod seterusnya. Contoh carian jenis *uniformed search* adalah seperti teknik *breadth-first*, teknik *depth-first* dan lain-lain [10]. Teknik *breadth-first* merupakan algoritma yang biasa digunakan oleh *web crawler* untuk mengumpul semua laman web pada peringkat tertentu sebelum kepada peringkat seterusnya.

Carian berpanduan (*informed search*) merupakan carian yang menyimpan maklumat mengenai setiap sasaran nod yang dicapai seterusnya dan jarak nod terhadap capaian sasaran boleh dijangkakan. Maklumat tersebut seperti jumlah pautan yang terdapat di dalam sesebuah laman web (*number of in-link)*, skor kiraan *PageRank*, frekuensi kata kunci dan *query* carian digunakan sebagai heuristik untuk menentukan nod yang mana dicapai dahulu. Contoh carian berpanduan ialah seperti teknik *best-first, hill-climbling,* A* [2] dan lain-lain. Teknik *best-first* adalah teknik carian yang biasa digunakan.

Selain dariapada *breath-first* dan teknik *best-first*, algoritma carian lain yang biasa digunakan dalam proses *crawling* adalah strategi *crawling* dengan menggunakan teknik *backlink-count* [12], *batch-pagerank* [12], *partial-pagerank* [12] dan *on-line page important computation (OPIC)* [13].

2.4.1 Teknik *Breadth-first*

Teknik *breadth-first* adalah *first-in-first-out* (FIFO) di mana nod pertama yang dimasukkan akan dikeluarkan dahulu [10]. Kebanyakan *web crawler* menggunakan teknik melebar dahulu untuk proses *crawling* kerana algoritma carian ini bersifat mudah dan tidak kompleks. Semua URL pada aras kedalaman pertama akan dijelajah dahulu sebelum menjelajah kepada aras kedalaman seterusnya. Nilai heuristik tidak digunakan untuk menentukan URL yang mana dikehendaki dijelejah seterusnya.

2.4.2 Teknik *Best-first*

Teknik *best-first* adalah algoritma carian ruang yang menggunakan heuristik untuk menyusun URL dan menentukan nod yang mana dicapai dahulu [10]. URL yang berkaitan akan disusun pada bahagian depan sesuatu *query* , manakala URL yang kurang berkaitan akan disusun pada bahagian belakang sesuatu *query* untuk dilayari. Taraf URL ini ditentukan oleh fungsi darjat (*rank function*). URL yang baru dimasukkan ke dalam senarai *query* akan kehilangan peluang untuk dijelajah sekiranya ia mempunyai nilai had tertinggi (*ceiling value*).

2.4.3 Teknik *Backlink-count*

Teknik *backlink-count* adalah teknik di mana laman web yang mempunyai nilai bilangan hubungan yang terbanyak berhubung dengannya akan dilayari dahulu [12]. Oleh sebab itu, laman web yang seterusnya dilayari adalah laman web yang paling berkaitan dengan laman web yang telah dimuatturunkan.

2.4.4 Teknik *Batch-pagerank*

Web crawler akan memuatturun sejumlah *K* laman web dan semua laman web tersebut akan dinilaikan dengan menggunakan kiraan *pagerank*. Sejumlah *K* laman web yang akan dimuatturun seterusnya adalah laman web yang mempunyai nilai *pagerank* yang tertinggi. Nilai *pagerank* ini akan dikira setiap kali terdapat URL yang baru di*download*. Teknik *batch-pagerank* adalah lebih baik jika berbanding dengan teknik *backlink-count* [12].

2.4.5 Teknik *Partial-pagerank*

Teknik *partial-pagerank* adalah sama seperti teknik *batch-pagerank*, tetapi semasa proses mengira semula nilai *pagerank*, satu nilai *pagerank* sementara disetkan kepada

laman web yang baru dimuatturun [12]. Nilai *pagerank* sementara ini adalah sama dengan total *normalized rankings* sesuatu laman web yang berhubung dengannya. Dengan itu, laman web yang terkini dijumpai dapat *crawl* oleh *web crawler* dengan secepat mungkin. Laman web yang di*download* adalah laman web yang nilai *partial pagerank* yang tertinggi.

2.4.6 Teknik *On-line Page Important Computation (OPIC)*

Dalam teknik OPIC, semua laman web permulaan mempunyai nilai *cash* yang sama. Setiap kali laman web tertentu *crawl* oleh *web crawler*, nilai *cash*nya akan dibahagikan kepada kalangan laman web yang berhubung dengannya [13]. Laman web yang belum dijelajahi oleh *web crawler* akan mempunyai jumlah nilai *cash* daripada laman web yang berhubung dengannya. Strategi ini hampir sama dengan *pagerank*, tetapi pengiraannya adalah tidak diulangi. Oleh sebab itu, teknik dengan menggunakan strategi ini adalah lebih pantas.

2.5 Masalah *Web Crawler*

Kebanyakan *web crawler* adalah hanya menggunakan bahagian tajuk laman web sahaja untuk proses pengindeksan. Bahagian tajuk adalah bahagian yang ditulis atau tidak ditulis oleh pengaturcara web dalam dokumen tersebut. Sebanyak 20% dokumen yang dijelajah oleh *web crawler* tertentu tidak mempunyai bahagian tajuk dalam dokumen tersebut [8]. Proses pengindeksan yang hanya menggunakan tajuk dokumen tersebut akan menyebabkan bahagian penting seperti isi kandungan dokumen tidak diambil kira semasa proses tersebut dilakukan. Tambahan pula, maklumat pada bahagian tajuk tersebut tidak boleh mengambarkan isi kandungan dokumen tersebut dengan sepenuhnya. Oleh sebab itu, keputusan carian yang dijanakan oleh *web crawler* adalah kurang tepat.

Cara penyelesaian bagi masalah ini adalah menjalankan proses pengindeksan dengan menggunakan maklumat pada bahagian tajuk dan isi kandungan dokumen tersebut (*full text indexing*). Proses pengindeksan yang menggunakan tajuk dokumen dan isi kandungan dokumen dapat mengambarkan dokumen tersebut dengan lebih menyeluruh. Dengan itu, *web crawler* dapat membekalkan maklumat yang lebih tepat kepada pengguna. Selain itu, penggunaan teknik GA juga boleh digunakan untuk

menjanakan keputusan carian yang lebih tepat. GA akan mengembangkan kata kunci yang dimasukkan oleh pengguna supaya hasil carian mempunyai nilai *precision* yang tinggi [14].

Proses pencarian sumber maklumat bermula dengan URL permulaan yang dimasukkan oleh pengguna. *Web crawler* akan meneroka semua laman web yang berhubung dengan laman web permulaan. Kemudian, bilangan *web crawler* akan bertambah berdasarkan bilangan laman web yang baru diteroka dan seterusnya. Web *crawler* akan menjelajah ke dalam kesemua laman web pada semua kedalaman. Oleh sebab itu, bilangan *web crawler* yang menjalani fungsi carian akan semakin bertambah banyak. Maklumat yang dikumpul dan diperolehi oleh *web crawler* juga akan banyak untuk disimpan ke dalam pangkalan data komputer pengguna [7]. Tetapi hal ini akan membebankan pangkalan data komputer pengguna.

Salah satu cara menyelesaikan masalah *overload* adalah penggunaan *robots exclusion protocol*. *Robots exclusion protocol* juga dikenali sebagai *robots.txt protocol* boleh digunakan oleh pentadbir web untuk menentukan bahagian-bahagian pelayan (*web server*) yang menyimpan laman web tidak akan boleh dicapai oleh *web crawler*. *Robots.txt protocol* juga boleh digunakan oleh pentadbir web untuk menentukan bahagian-bahagian dokumen yang berguna supaya dicapai oleh *web crawler*. Fungsi ini akan mengelakkan *web crawler* daripada memperolehi maklumat yang tidak berguna demi menjamin kualiti proses *indexing*.

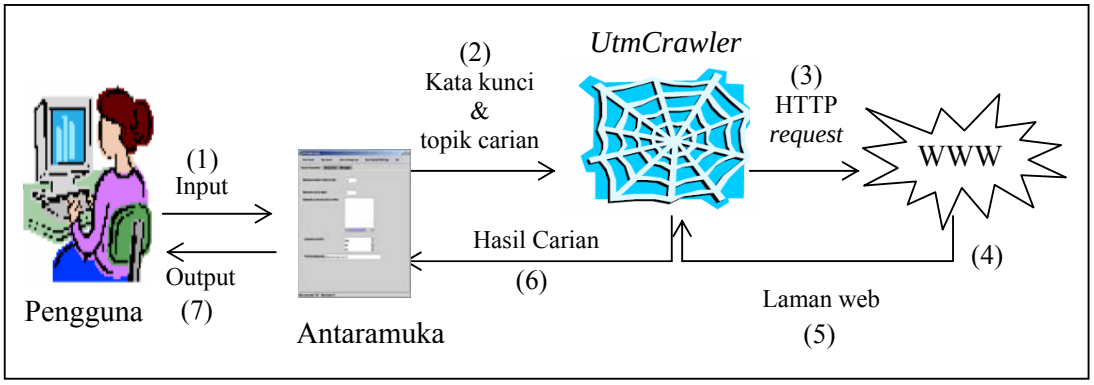
Sela masa antara permintaan antara pengguna dan sistem juga pelu dipertimbangkan supaya proses carian dijalankan dengan lebih lancar. Sela masa tersebut perlulah dalam satu tempoh masa menunggu yang singkat semasa proses *crawling* dilakukan. Contohnya, kajian Cho, J. et al. [15] telah meletakkan jumlah tetap 10 saat sebagai sela masa yang perlu ada pada *web crawler* bagi mencapai dokumen tertentu. Selain itu, cara pengumpulan maklumat yang sistematik juga diperlukan. Maklumat yang diperolehi oleh *web crawler* pada aras tertentu akan disimpan dalam satu fail. Kemudian, fail-fail pada aras tersebut akan dikumpul dan disimpan ke dalam satu fail dan seterusnya. Dengan demikian, hasil hantaran akhir oleh *web crawler* kepada pangkalan data adalah hanya satu fail yang bermaklumat

sistematik demi mengatasi masalah *overload* dalam penyimpanan maklumat ke dalam pangkalan data.

UtmCrawler yang adibangunkan telah menggunakan teknik agoritma genetik (GA) untuk mengatasai masalah hasil carian yang dijanakan oleh *UtmCrawler* adalah kurang tepat. Satu cara pengumpulan maklumat yang sistematik dan berstruktur digunakan untuk mengatasi masalah *overload* dalam penyimpanan maklumat ke dalam pangkalan data.

3.0 SENIBINA *UTMCRAWLER*

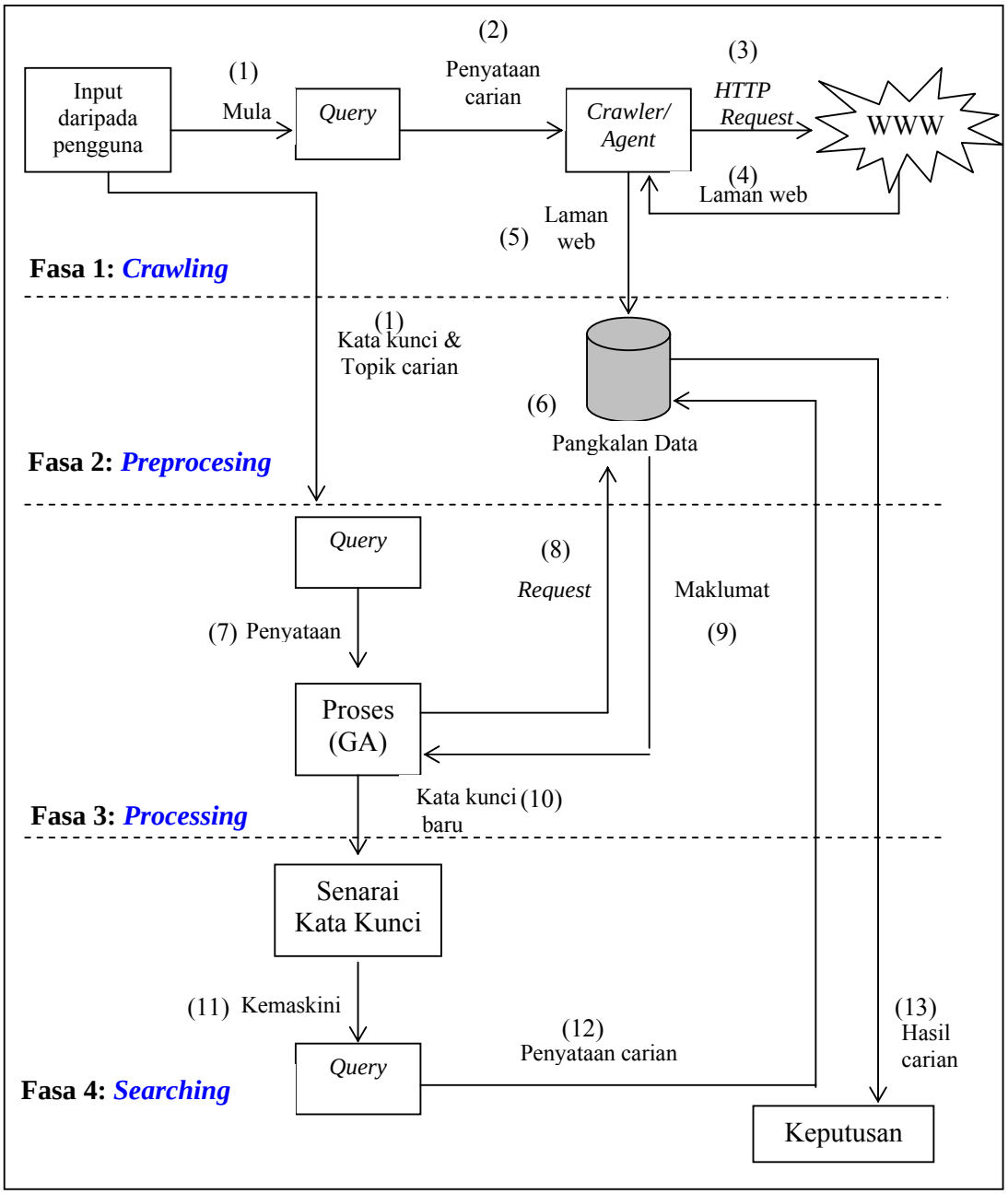
Seperti yang ditunjukkan pada Rajah 2, pengguna akan memasukkan input kepada *UtmCrawler* melalui antaramuka yang dibangunkan dengan menggunakan GUI dan Java. Kemudian, *UtmCrawler* akan bermula menjalankan proses carian dengan menghantar *HTTP request* kepada WWW yang dinamik. Laman web yang diperolehi oleh *UtmCrawler* akan diproses dan disimpan dalam pangkalan data pengguna. Hasil keputusan yang mempunyai nilai *precision* yang tinggi akan dihantar dan dipaparkan kepada pengguna melalui antaramuka *UtmCrawler*. Rajah 2 menunjukkan senibina dan proses yang dilakukan oleh *UtmCrawler* secara teliti dari proses (1) hingga proses (7) bagi mencapai isi kandungan laman web yang dikehendaki oleh pengguna.



Rajah 2: Gambarajah peringkat tinggi *UtmCrawler*

Setiap proses yang dirujuk di dalam Rajah 2 dapat dijelaskan secara terperinci di dalam Rajah 3 dimana fasa *crawling* adalah proses pencarian maklumat oleh agen dalam WWW. Fasa pra-pemprosesan *(preprocessing)* adalah penyusunan laman web

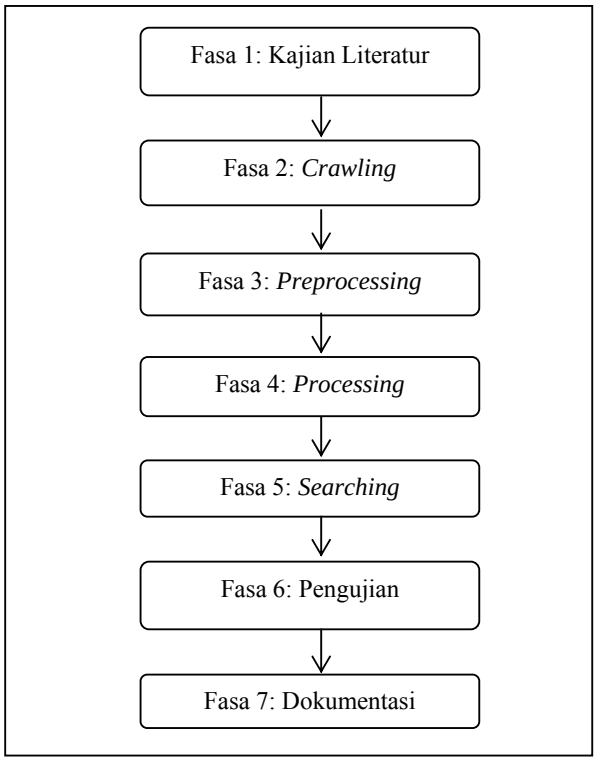
mengikut nombor rujukan dan penyimpanan maklumat laman web dengan sistematik (rujuk nombor (1) hingga (13) di dalam Rajah 3 untuk melihat proses yang terlibat). Proses GA dilaksanakan pada fasa *Processing* dan hasil modul GA adalah senarai kata kunci yang baru dan berkaitan dengan kata kunci permulaan. Pada fasa *Searching*, senarai kata kunci baru akan mengemaskinikan penyataan carian dan pencarian maklumat dalam pangkalan data. Hasil carian dipaparkan sebagai output kepada pengguna.



Rajah 3 Gambarajah senibina *UtmCrawler*

4.0 METODOLOGI

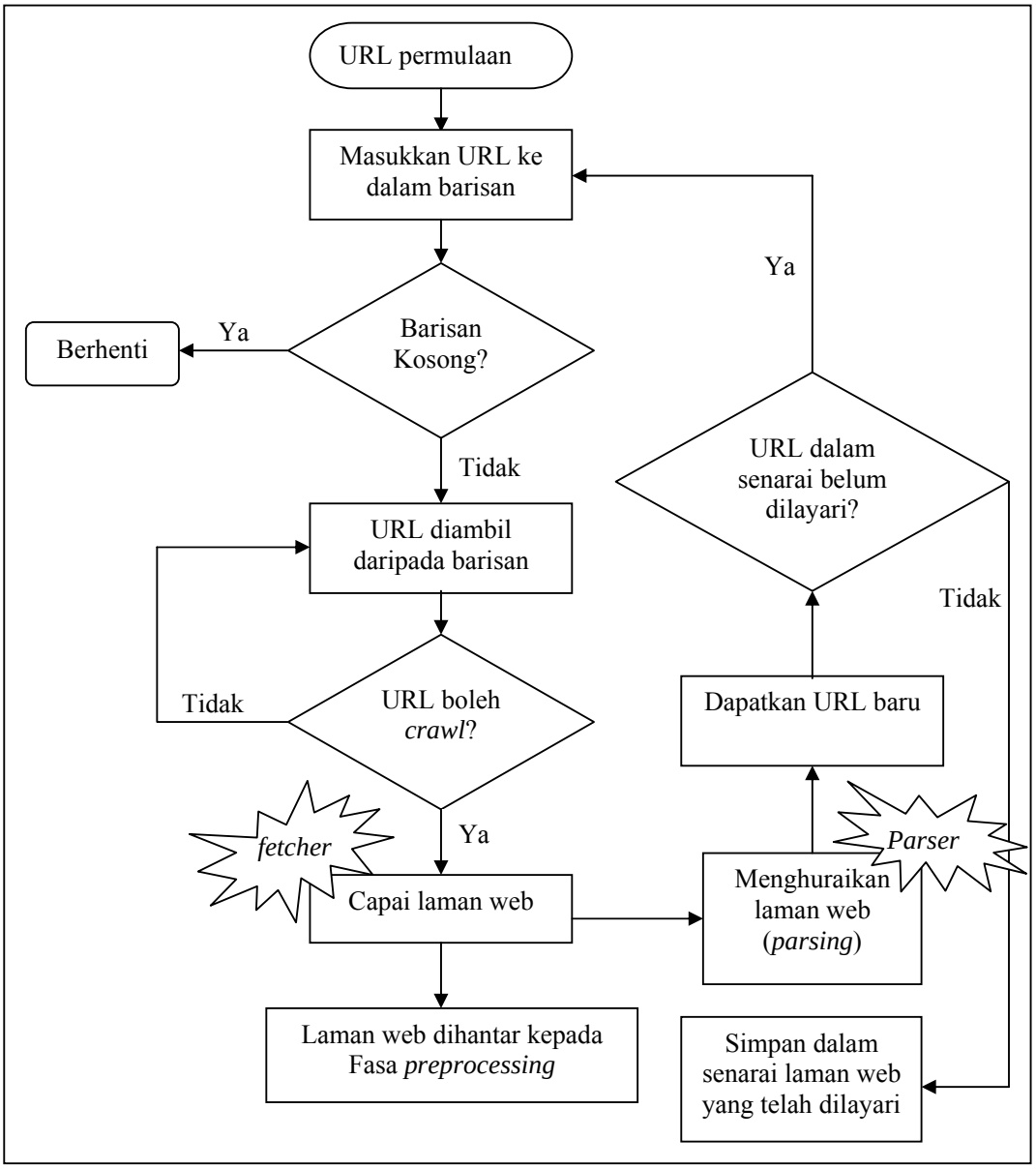
Rajah 4 menunjukkan metodologi yang akan digunakan dalam pembangunan *UtmCrawler* di mana terdiri daripada tujuh fasa utama iaitu fasa kajian literatur, fasa *crawling*, fasa *preprocessing*, fasa *processing*, fasa *searching*, fasa dokumentasi dan fasa pengujian.



Rajah 4 Metodologi pembangunan *UtmCrawler*

4.1 Fasa Kajian Literatur

Fasa ini melibatkan tiga tugas utama. Tugas pertama ialah merancangkan projek awalan. Matlamat projek, objektif projek, skop projek, latarbelakang masalah dan kepentingan projek dikaji dan dikenalpastikan. Tugas kedua pula adalah mengkaji dan menganalisa topik yang berkaitan dengan *web crawler* dengan teliti. Kajian tersebut akan merangkumi penelitian terhadap *web crawler*, strategi *crawling,* teknik algoritma genetik (GA) dan topik yang berkaitan dengannya. Tugas ketiga adalah merekabentuk *UtmCrawler* iaitu merekabentuk senibina *UtmCrawler*, merekabentuk antaramuka yang menghubungkan *UtmCrawler* dengan pengguna, merekabentuk aliran proses dan merekabentuk pangkalan data.

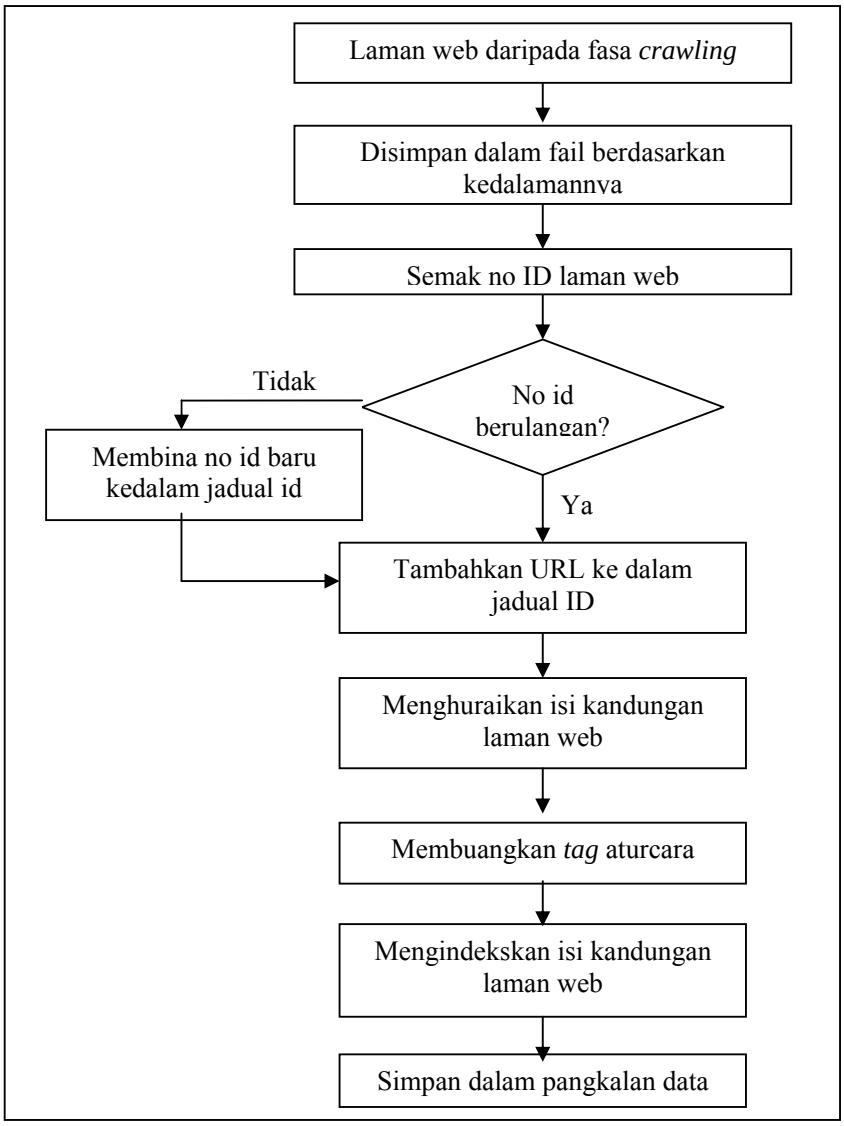


Rajah 5: Aliran kerja proses *UtmCrawler*

4.2 Fasa *Crawling*

Dalam fasa ini, proses *crawling* yang dilaksanakan oleh *UtmCrawler* adalah digunakan untuk mencari semua maklumat dalam internet. Teknik *breadth-first* dipilih sebagai strategi *crawling* kerana teknik tersebut bersifat mudah dan menjelajahi semua laman web dalam internet. *UtmCrawler* akan menjelajah dari satu pelayan ke pelayan yang lain untuk mendapatkan sumber maklumat yang diperlukan.

Rajah 5 menunjukkan aliran kerja modul *crawling* di mana *fetcher* dan *parser* adalah satu jenis agen yang mempunyai fungsi tertentu.

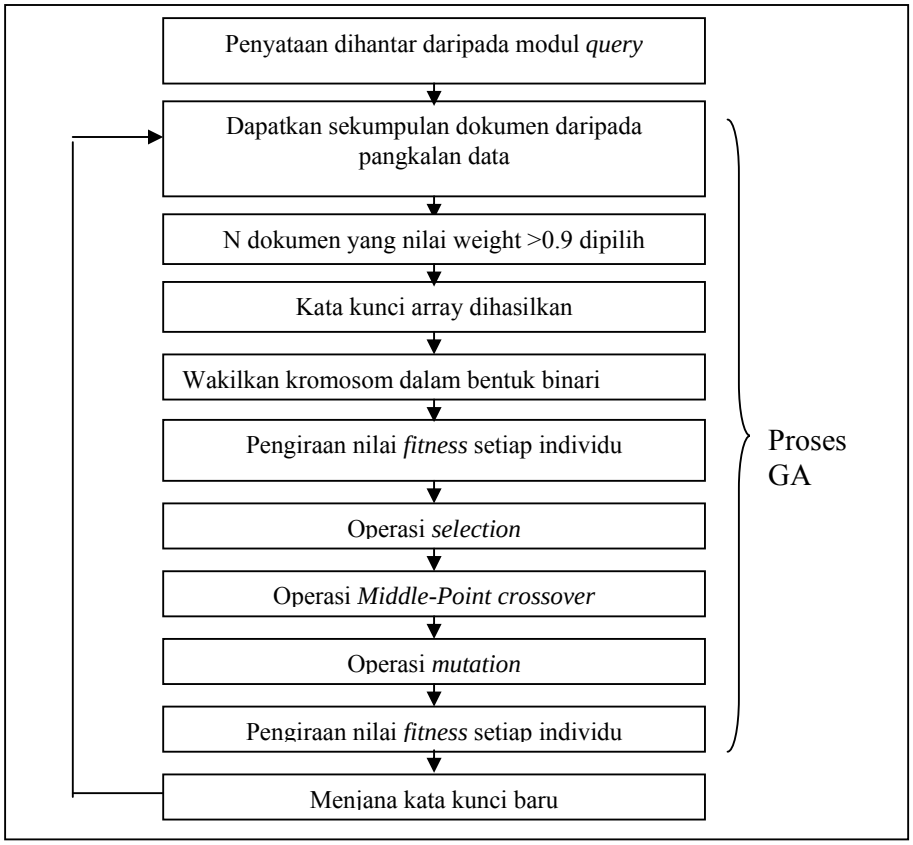


Rajah 6: Aliran kerja fasa *preprocessing*

4.3 Fasa *PreProcessing*

Dalam fasa ini, proses yang terlibat adalah proses menyusun laman web mengikut Idnya dan proses penyimpanan maklumat laman web dengan sistematik. Satu laman web mempunyai nombor rujukan *(ID)* yang unik, isi kandungan yang sama tetapi boleh memiliki nama URL yang berbeza. Dalam fasa ini, laman web disusunkan

berdasarkan Idnya supaya URL yang isi kandungan dan Idnya yang sama adalah tidak berulangan dalam senarai dan pangkalan data. Proses penyimpanan maklumat carian ke dalam pangkalan data dengan sistematik dapat mengurangkan beban pelayan dan pangkalan data dan memudahkan proses penjanaan rajah pokok carian. Rajah 6 menunjukkan aliran kerja modul *preprocesing.*



Rajah 7: Aliran kerja fasa *processing*

4.4 Fasa *Processing*

Dalam fasa ini, teknik GA digunakan untuk mengembangkan kata kunci carian permulaan. Contohnya, satu kata kunci baru iaitu *British* akan ditambahkan kepada kata kunci permulaan yang terdiri daripada *Iraq, Iraqi, US* dan *War* selepas melalui proses GA. Ini disebabkan kebanyakan laman web yang mencatatkan berita pasukan tentera *British* dalam negara *Iraq*. Proses perkembangan kata kunci carian permulaan ini dapat menjanakan keputusan carian yang mempunyai nilai *precision* yang tinggi. Nilai *Precision* digunakan untuk mengukur darjah ketepatan dokumen yang

diperolehkan oleh *UtmCrawler* adalah sama dengan *query* carian. Rajah 7 menunjukkan aliran kerja modul *processing.*

Contohnya, katakan kata kunci permulaan adalah *information* dan *retrieval*. Satu *request* dihantar kepada pangkalan data supaya memperolehi dokumen-dokumen untuk diproses. Pengiraan nilai pemberat (*weight*) terhadap semua dokumen-dokumen. Lima dokumen yang nilai pemberat lebih daripada 0.9 dipilih dan kata kunci setiap dokumen disusunatur dalam bentuk *array* seperti yang ditunjukkan pada Rajah 8. Pengiraan pemberat adalah berdasarkan skema pemberat *Lnu.Itu* [17]. Contoh isi kandungan lima dokumen adalah seperti dalam Jadual 1.

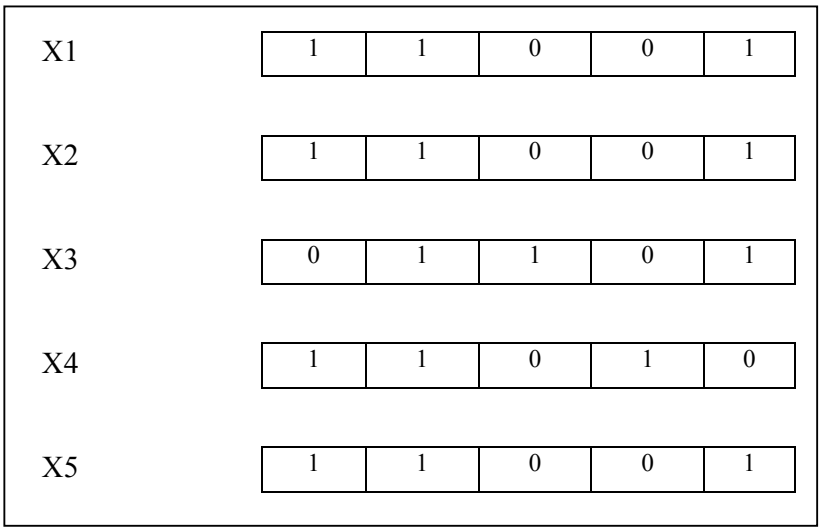
Jadual 1: Isi kandungan lima dokumen dan kata kunci masing-masing

Dokumen	**Isi kandungan laman web (dalam bentuk *TF)***	**Kata kunci**
Doc 1	*Data – 8, Retrieval-4, Information-2, Queries-1, Database-2*	*Data*
Doc 2	*Data – 2, Retrieval-2, Information-4, Computer-1*	*Information*
Doc 3	*Indexing-3, System-1, Retrieval-2, Information-5, IR-6*	*IR*
Doc 4	*Query-9, Information-3, Data-5*	*Query*
Doc 5	*Retrival-5, Information-4, Data-2*	*Retrieval*

Data	*Information*	*IR*	*Query*	*Retrieval*

Rajah 8: Kata kunci *array*

Nilai binari *array* yang panjangnya 5 diberikan kepada setiap individu. Nilai gen disetkan 1 sekiranya individu tersebut mempunyai *term* yang sama dengan kata kunci *array* dan sebaliknya. Rajah 9 menunjukkan perwakilan 5 dokumen tersebut dalam kromosom.



Rajah 9: Perwakilan dokumen dalam kromosom

Nilai *fitness* untuk setiap individu dalam *populatian* dikirakan. *Fitness function* yang digunakan adalah pengiraan Jaccard seperti yang ditunjukkan di dalam persamaan **(1)**. Nilai *fitness* setiap individu adalah seperti yang ditunjukkan dalam Jadual 2.

Jadual 2: Nilai *fitness* bagi *initial population*

Dokumen	Kromosom	Nilai fitness	Fitness/total fitness *100%
X1	11001	0.8	22.79%
X2	11001	0.75	21.37%
X3	01101	0.54	15.38%
X4	11010	0.52	14.81%
X5	11001	0.9	25.64%

Kromosom atau individu yang mempunyai nilai *fitness* yang tinggi akan dipilih dalam operasi *selection*. Dua individu yang mempunyai nilai *fitness* yang tinggi dipilih untuk melaksanakan proses *crossover*. Teknik *crossover* yang digunakan adalah *middle-point crossover*. Operasi *mutation* dilaksanakan setelah operasi *crossover* diselesaikan. Nilai *fitness* untuk setiap individu dalam *population* baru dikirakan. Jadual 3 menunjukkan nilai *fitness* untuk *population* baru.

Jadual 3: Nilai *fitness population* baru

Dokumen	Kromosom	Nilai fitness
X1	11001	0.95
X2	11001	0.95
X3	11001	0.95
X4	11001	0.95
X5	11011	0.8

4.5 Fasa *Searching*

Kata kunci baru yang dihasilkan daripada fasa *processing* digunakan untuk mengemaskini penyataan carian. Satu *Request* dihantar kepada pangkalan data untuk memulakan proses carian. Proses carian dalam pangkalan data berlaku berdasarkan penggabungan kata kunci permulaan dengan kata kunci baru. Hasil carian akan dihantar dan dipaparkan kepada pengguna melalui antaramuka *UtmCrawler*.

4.6 Fasa Pengujian

Benchmarking yang akan dijalankan adalah berdasarkan *fitness function* yang digunakan dalam proses GA pada fasa *Processing. Fitness function* yang terpilih adalah Jaccard Coefficient **(1)**, Cosine similarity **(2)** dan Haming distance. Keputusan perbandingan akan dinilai berdasarkan nilai *precesion*, *recall* dan *F1* di mana *F1* adalah purata bagi nilai *precesion* dan *recall*.

5.0 PERBINCANGAN DAN KEPUTUSAN

Daripada Jadual 3, X2 yang mempunyai nilai *fitness* yang paling tinggi dipilih untuk mengembangkan kata kunci permulaan. Kata kunci X2 mengandungi *Data, Information* dan *Retrieval*. Oleh sebab kata kunci *Information* dan *Retrieval* adalah kata kunci permulaan, maka kata kunci Data dipilih sebagai kata kunci baru pada generasi pertama proses GA. Proses GA diulangkan. Purata nilai *fitness* bertambah daripada 0.702 (Jadual 2) kepada 0.92 (Jadual 3)

Seperti yang ditunjukkan dalam Jadual 4, apabila kata kunci tersebut ditambahkan kepada kata kunci carian permulaan, keputusan carian yang dijanakan akan mempunyai nilai *precision* yang lebih tinggi jika berbanding dengan keputusan carian dengan kata kunci carian permulaan.

Jadual 4: Keputusan awal carian menggunakan *UTMCrawler*

Kata Kunci Lama	**Kata Kunci Baru**	**Peratusan precision dengan kata kunci lama**	**Peratusan precision dengan kata kunci baru**
Iraq, Iraqi, US, War	**British**	82%	98%
Presley, Rock, Music, Song	**Elvis**	60%	80%
France, French, Revolution, History	**Napoleon**	66%	70%

6.0 KESIMPULAN

Secara kesimpulannya, *UtmCrawler* yang dibangunkan menggunakan teknik agoritma genetik (GA) adalah untuk mengatasai masalah hasil carian yang dijanakan oleh *UtmCrawler* adalah kurang tepat. Keputusan carian yang dipaparkan mempunyai nilai *precision* dan nilai *fitness* yang tinggi dan tidak memerlukan sebarang latihan dan *Relevance Feedback* (RF) dalam proses carian dengan menggunakan GA. Selain itu, satu cara pengumpulan maklumat yang sistematik dan berstruktur digunakan untuk mengatasi masalah maklumat berlebihan *(overload)* dalam penyimpanan maklumat ke dalam pangkalan data. Cara penyimpanan maklumat ke dalam pangkalan data yang sistematik dapat mengurangkan beban *server* dan pangkalan data. Tambahan pula, peristiwa URL yang berulangan dalam senarai hasil carian dan pangkalan data tidak akan berlaku.

RUJUKAN

[1] Pierre, J. M. 2000. Practical issues for automated categorization of web pages. *ECDL 2000 Workshop on the Semantic Web.* Lisbon, Portugal.

[2] Pant, G. dan Menczer, F. 2002. MySpiders: Evolve Your Own Intelligent Web Crawlers. *Autonomous Agents and Multi-Agent Systems*, 5, 221–229, 2002, Kluwer Academic Publishers. Manufactured in The Netherlands.

[3] *www.google.com,* Google search engine, 2006.

[4] *www.altavista.com,* Altavista search engine, 2006.

[5] *www.yahoo.com,* Yahoo! search engine, 2006.

[6] Tsay, J-J. , Shih, C-Y. dan Wu, B-O. 2005. AutoCrawler- An Integrated System for Automatic Topical Crawler. *Proceeding of the fourth annual ACIS International Conference on Computer and Information Science.*

[7] Koster, M. 1995. Robots in the web: threat or treat ? *ConneXions*, 9(4).

[8] Pinkerton, B. 1994. Finding what people want: Experiences with the WebCrawler. *In Proceedings of the First World Wide Web Conference*, Geneva, Switzerland. 1994.

[9] Wikipedia, the free encyclopedia. (accessed September 2, 2006). http://en.wikipedia.org/wiki

[10] Luger, G.F. *Artificial Intelligence structures and Strategies for Complex Problem Solving* 4th ed. 2002. Addison-Wesley. Page 81-158

[11] Negnevitsky, M. *Artificial Intelligence: A Guide to Intelligence Systems*. 2nd ed. 2005. Addison-Wesley. Page 219-258.

[12] Cho, J., Garcia-Molina, H., Page, L. 1998. Efficient Crawling through URL Ordering. *Proc. the 7th International World-Wide Web Conference*. Brisbane, Australia, Apr 1998.

[13] Abiteboul, S., Preda, M., and Cobena, G. 2003. Adaptive on-line page importance computation. *In Proceedings of the twelfth international conference on World Wide Web*: 280-290.

[14] Shokouhi, M., Chubak, P., Raeesy, Z. 2005. Enchancing Focused Crawling with Genetic Algorithms. *Proceedings of the International Conference on Information Technology: Coding and Computing* (ITCC'05).

[15] Cho, J. dan Garcia-Molina, H. 2000. Synchronizing a database to improve freshness. *In Proceedings of ACM International Conference on Management of Data (SIGMOD)*, pages 117-128, Dallas, Texas, USA.

[16] Selamat, A., Omatu, S, and Yanagimoto, H. 2003. *Web News Categorization Using Neural Networks*, IEEJ Transactions on Electrical and Information Systems, Vol. 123, No. 5, pp. 1020-1026.

[17] Mitra, M., Singhal, A., Buckley, C. 1998 .Improving automatic query expansion, *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, Melbourne, Australia, pp. 206 - 214, ISBN:1-58113-015-5