BAB I
PENDAHULUAN

1.1 Latar Belakang

Semakin canggihnya teknologi di bidang komputasi dan telekomunikasi khususnya internet membuat informasi mudah untuk didapatkan oleh banyak orang. Kemudahan ini mendorong pertambahan jumlah informasi digital menjadi semakin banyak dan beragam. Informasi dapat berupa berita, dokumen, surat, cerita, laporan penelitian, data keuangan, dan lain-lain. Tidak dapat dipungkiri lagi informasi telah menjadi komoditi yang paling penting dalam dunia modern masa kini.

Bertambahnya jumlah dokumen teks yang dapat diakses di internet diikuti dengan meningkatnya kebutuhan pengguna akan perangkat pencarian informasi yang efektif dan efisien (Agusta, 2009). Sistem temu balik informasi bertujuan menghasilkan dokumen yang paling relevan berdasarkan *keyword* pada *query* yang diberikan pengguna. Dokumen dianggap relevan jika suatu dokumen cocok dengan pertanyaan pengguna.

Information Retrieval (IR) adalah proses penemuan dokumen tidak terstruktur (biasanya berupa teks) yang sesuai dengan kebutuhan informasi dari koleksi besar yang biasanya tersimpan dalam komputer (Manning, 2009). Akan tetapi, istilah-istilah yang terdapat dalam dokumen dan *query* sering memiliki banyak varian morfologik, sehingga pasangan istilah seperti “bergeser” dan “menggeser” tidak akan dianggap ekuivalen oleh sistem tanpa suatu bentuk *Natural Language Processing* (NLP).

Menurut konsep Luhn (1958), istilah sebaiknya dibedakan menurut jumlah kemunculannya dalam dokumen. Dalam konsep tersebut, istilah terbaik adalah istiah yang muncul dalam frekuensi menengah. Konsep tersebut mengasumsikan istilah yang jarang muncul adalah istilah yang kurang representatif, sedangkan istilah yang terlalu sering muncul akan membuat istilah tersebut menjadi tidak penting. Pembobotan dasar dilakukan dengan menghitung frekuensi kemunculan istilah dalam dokumen karena dipercaya bahwa frekuensi kemunculan istilah merupakan petunjuk sejauh mana istilah tersebut mewakili isi dokumen (Hamzah, 2009).

Pembobotan kata diharapkan dapat menemukan kembali informasi yang paling relevan dengan indeks istilah terbaik. Metode pembobotan kata berdasarkan kombinasi TF-IDF (*term frequency-inverse document frequency*) memberikan bobot lebih kepada istilah yang lebih penting. Istilah yang lebih penting yang dimaksud adalah istilah yang jika muncul pada sebuah dokumen maka dokumen tersebut dapat dianggap relevan dengan *query*. Dengan menggunakan metode pembobotan kata dapat diperoleh informasi penting dari suatu dokumen berdasarkan kata-kata yang ada dalam dokumen tersebut.

Penambahan istilah pada query juga diperlukan untuk meningkatkan performa dalam *information retrieval* atau dikenal dengan istilah *Query Expansion* atau perluasan query (Nugroho, 2009). Dalam konteks *web search engine*, hal ini termasuk evaluasi input user dan memperluas query pencarian untuk mendapatkan dokumen yang cocok dengan query (Qiu, 1993). Proses perluasan dalam sistem ini dilakukan dengan menggunakan kamus tesaurus. Metode yang dilakukan dalam perluasan adalah dengan mencari sinonim dalam bentuk *unstemmed-term* dari query. Pencarian sinonim tidak memperhatikan tiap relasi dari sinonim set (synset) yang ditemukan dalam tesaurus, dan hanya akan diambil maksimal 5 arti dari tiap istilah yang sinonimnya ditemukan.

Kata yang dimasukkan pengguna akan melewati proses ekstraksi kata (tokenisasi), eliminasi kata-kata yang tidak layak dijadikan kata kunci pencarian (*stop words*), *query expansion* dan pengembalian kata menjadi bentuk dasar (*stemming*) untuk memperkuat pencarian. Proses ekstrasi kata (tokenisasi) yaitu proses pemisahan kata dari dokumen dengan menggunakan karakter spasi sebagai tanda pemisahnya (Wibowo, 2010). Kata yang telah diproses beserta kata tambahan hasil *query expansion* akan melalui proses *stemming*, kemudian akan digunakan dalam pencarian. Pencarian diusulkan terdiri atas tiga bagian utama yaitu pengubahan dokumen dari berbagai format menjadi format teks (.txt) (*document parsing*), *document pre-processing* dan pembobotan kata pada masing-masing dokumen. *Document parsing* berfungsi untuk memudahkan proses ekstraksi kata dari dokumen. *Document pre-processing* melakukan ekstraksi kata dari dokumen (tokensisasi dokumen), eliminasi *stop words* dan *stemming*. Ketika

pencarian dilakukan, aplikasi melakukan proses pengurutan dari hasil bobot masing-masing dokumen.

Informasi berupa dokumen teks yang tersedia di internet memiliki banyak variasi. Sebuah sistem temu balik informasi dapat melakukan pencarian berbagai informasi. Dalam penelitian ini, pencarian dilakukan terhadap dokumen hasil *crawling* dari beberapa situs yang sudah ditentukan. *Crawling* adalah proses mengumpulkan halaman dari web untuk diindeks dan membantu mesin pencari (Manning, 2009). Fokus utama penelitian ini, berupa dokumen hasil *crawling* dari situs berita, tanpa menutup kemungkinan untuk melakukan pencarian terhadap dokumen hasil *crawling* dari situs lain.

1.2 Perumusan Masalah

Pada sistem yang terbuka seperti internet, semakin lama jumlah dokumen yang tersedia akan semakin banyak. Proses pencarian menjadi hal yang sangat penting untuk mengelola data tersebut dalam jumlah yang besar. Bentuk morfologik istilah yang memiliki banyak varian tidak akan dianggap ekuivalen oleh sistem tanpa suatu bentuk *Natural Language Processing* (NLP). Jumlah kemunculan indeks istilah juga berpengaruh terhadap pentingnya indeks istilah tersebut dalam sebuah dokumen. Berdasarkan hal tersebut, bagaimana merancang sistem temu balik informasi yang mampu mencari informasi yang dibutuhkan secara efektif dan efisien berdasarkan bentuk *Natural Language Processing* (NLP) dari kata kunci pencarian dengan memberikan bobot pada setiap kata dengan metode pembobotan kombinasi TF-IDF, dan menampilkan serta mengurutkan data yang paling relevan terhadap kata kunci pencarian.

1.3 Tujuan Penelitian

Menghasilkan suatu sistem temu balik informasi khususnya berita yang didalamnya terdapat proses pengumpulan dokumen, pembobotan kata dan pencarian dokumen dengan menggunakan metode pembobotan kombinasi TF-IDF sehingga dapat ditemukan informasi yang dibutuhkan dengan efektif dan efisien.

1.4 Pembatasan Masalah

Pembatasan masalah dari penelitian yang akan dilakukan antara lain:

1. Penelitian ini difokuskan pada proses pengumpulan dokumen dan pencarian dokumen dengan menggunakan metode pembobotan kata.
2. Tipe dokumen yang digunakan adalah dokumen berita hasil *crawling* dari beberapa situs berita yang telah ditentukan yaitu www.detik.com, www.antaranews.com, pontianak.tribunnews.com .
3. Dokumen yang digunakan hanya dokumen yang berbahasa Indonesia.

1.5 Sistematika Penulisan Skripsi

Adapun sistematika penulisan dari tugas akhir ini disusun dalam lima bab yang terdiri dari Bab I Pendahuluan, Bab II Tinjauan Pustaka, Bab III Metodologi Penelitian, Bab IV Hasil dan Analisis Sistem serta Bab V Penutup.

Bab I Pendahuluan adalah bab yang berisi latar belakang, perumusan masalah, tujuan penelitian, pembatasan masalah dan sistematika penulisan.

Bab II Tinjauan Pustaka adalah bab yang berisi landasan teori berhubungan dengan penelitian yang akan dilakukan dan uraian sistematis tentang hasil-hasil penelitian yang didapat oleh peneliti terdahulu.

Bab III Metodologi Penelitian adalah bab yang berisi tentang Bahan Penelitian, Alat yang dipergunakan, Metode Penelitian, Variabel atau Data, Analisis Hasil serta Diagram Alir Penelitian.

Bab IV Hasil Analisis dan Perancangan Sistem adalah bab yang berisi data hasil perancangan, pengujian, dan sebagainya yang telah dirancang pada Bab III. Setiap hasil yang disajikan akan dilakukan analisis untuk mengarah kepada suatu kesimpulan.

Bab V Penutup adalah bab yang berisi kesimpulan dari penelitian yang telah dilakukan dan saran/rekomendasi untuk perbaikan, pengembangan atau kesempurnaan/kelengkapan penelitian yang telah dilakukan

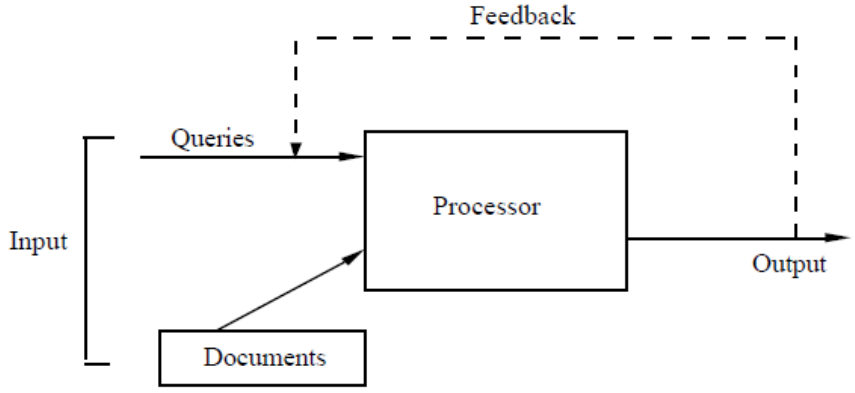
BAB II
TINJAUAN PUSTAKA

2.1 Sistem Temu Balik Informasi *(Information Retrieval)*

Pencarian informasi atau yang dikenal dengan istilah sistem temu balik informasi (*Information Retrieval*) digunakan untuk menemukan kembali (*retrieve*) secara otomatis informasi- informasi yang relevan terhadap kebutuhan pengguna dari suatu kumpulan informasi (Hadhiatma, 2010). Sistem temu balik informasi ini digunakan untuk mengurangi informasi yang terlalu banyak sehingga sulit untuk dikelola. Tujuan dari sistem temu balik informasi yang ideal adalah:

1. Menemukan seluruh dokumen yang relevan, atau
2. Menemukan dokumen relevan saja, artinya tidak terdapat dokumen tidak relevan pada dokumen hasil pencarian.

Sistem temu balik informasi terdiri dari tiga komponen utama, yaitu masukan (*input*), pemroses (*processor*), dan keluaran (*output*). Komponen-komponen ini digambarkan pada gambar 2.1 (Rijsbergen, 1979):



Gambar 2.1 Komponen Utama Sistem Temu Balik Informasi

1. *Input* adalah masukan yang diberikan oleh pengguna. Pengguna adalah pemilik kebutuhan akan informasi, yang kemudian menerjemahkan kebutuhannya menjadi sebuah *query*. *Input* harus berupa representasi yang tepat dari setiap dokumen dan *query* agar dapat diolah oleh pemroses.

2. Pemroses (*Processor*) adalah bagian yang paling krusial dalam sistem temu balik informasi. Tugas yang dilakukan oleh pemroses antara lain adalah:
   a. Menstrukturkan informasi dalam bentuk yang tepat, misalnya dengan pengindeksan dan klasifikasi.
   b. Melakukan proses temu balik, yaitu dengan menjalankan suatu strategi pencarian sebagai respon dari *query.*
3. *Output* adalah keluaran yang diberikan oleh pemroses. Output biasanya berbentuk informasi tentang suatu dokumen, dokumen itu sendiri, dan acuan ke dokumen lain (*citation*).

2.2 Sistem Temu Balik Informasi Berbasis Hiperteks

Berdasarkan kebutuhan informasi pengguna, sistem temu balik informasi mengemukakan model dan strategi pencarian untuk menemukan dokumen yang sesuai dengan kepentingan pengguna. Sebagai solusi pertama, sistem hiperteks telah menggunakan teknik pencarian klasik dengan mempertimbangkan dokumen sebagai entitas sendiri (Savoy, 1991).

Hiperteks dan temu-kembali informasi merupakan bidang penelitian yang berbeda satu dengan yang lain. Penggabungan kedua bidang ini dapat memecahkan masalah-masalah dalam bidang temu-kembali informasi (Hasibuan, 2001). Misalnya, sistem temu-kembali informasi yang didasarkan pada penggunaan operator boolean, mengandalkan kemampuan pemakai dalam memformulasikan *query*. Dengan adanya sistem hiperteks, hal ini dapat dipermudah dengan penyediaan antar muka yang memakai pencarian dengan metode *browsing*. Hiperteks membutuhkan lebih banyak *searching* sedangkan temu-kembali informasi membutuhkan lebih banyak *browsing*. Hal yang dimaksud adalah hiperteks akan semakin baik jika disertai dengan fasilitas pencarian, dan temu-kembali informasi membutuhkan *browsing* dalam melakukan pencarian yang efisien. Adapun maksud dari *searching* adalah berusaha mendapatkan atau mencapai tujuan spesifik sedangkan *browsing* adalah mengikuti suatu path sampai mencapai suatu tujuan.

Penggabungan sistem temu-kembali kedalam basis hiperteks lebih disebut *search engine*, dimana sistem ini dibagi dalam dua kategori berdasarkan sumber informasinya yaitu (Elvina, 2009):

1. *Worldwide Search Engine*

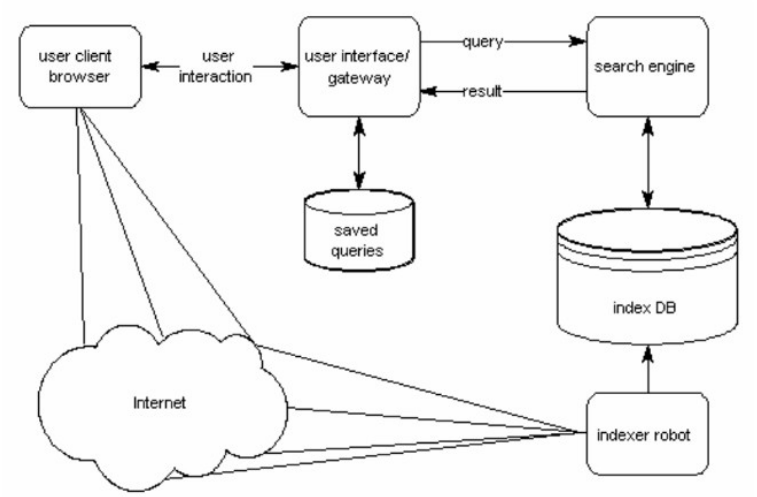
   Worldwide Search Engine adalah suatu sistem temu-kembali informasi yang mengambil data-data dari berbagai server di seluruh penjuru dunia. Data-data tersebut diambil melalui program yang disebut dengan "*robot*", "*bot*" atau "*crawler*". Program inilah yang melakukan pencarian data pada setiap server, yang kemudian dikirim ke server pusat pada selang waktu tertentu.

2. *Local Search Engine*

   Local Search Engine adalah suatu sistem temu-kembali informasi yang mengambil data-data dari server tertentu saja. Kata "*local*", yang berarti lokal atau setempat, memberi penekanan akan lokasi sumber data yang akan digunakan. *Local search engine* tidak dirancang untuk mengarungi belantara internet seperti *worlwide search engine*. Tujuan implementasi *local search engine* dimaksudkan untuk pencarian pada objek spesifik dan lebih kecil lingkupnya dibandingkan internet sendiri.

Penerapan *local search engine* dan *worlwide search engine* hanya akan mempengaruhi cara sistem pengindeksan dari temu-kembali. Sedangkan teknik *retrieval* dan rancangan penerapan teknik pada hiperteks akan sama saja.

Menurut Hasibuan (2001), Penelitian yang dilakukan Budi Yuwono (1995) menggunakan rancangan arsitektur seperti pada gambar 2.2.



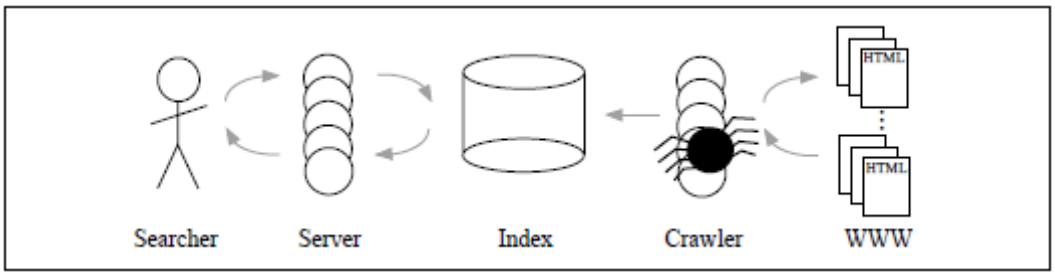
Gambar 2.2 Arsitektur Sistem Temu Balik Informasi

Arsitektur yang dirancang ini terdiri dari dua komponen utama yaitu: *Index Builder* dan *Search Engine*. *Index builder* merupakan sebuah sistem pengindeksan yang memanfaatkan "*robot*" atau "*crawler*" yang berkomunikasi dengan menggunakan HTTP (*Hypertext Transfer Protocol*) untuk mencari informasi yang akan diindeks. Sedangkan *search engine* merupakan teknik dari temu-kembali dalam menemukan dokumen dan sekaligus mengeksekusi algoritma peringkat dalam menampilkan dokumen. Sedangkan komunikasi antara pemakai dan *search engine* dalam memformulasikan *query* dilakukan melalui *user interface*. Setelah pemakai menemukan dokumen yang relevan dengan *query*, dapat langsung melakukan *browsing* ke sumber informasi dalam hal ini adalah alamat tempat www.

2.3 *Crawler*

Crawler merupakan program yang berjalan secara otomatis, berisi script program yang melakukan *crawling* melalui halaman website untuk mengumpulkan data berdasarkan indeks dari halaman web yang ditemukan (Sasongko, 2010). Tujuan dari crawler adalah dengan cepat dan efisien mengumpulkan banyak informasi dari halaman web yang berguna, berikut dengan struktur link yang terkoneksi dengan halaman web tersebut. *Crawler* paling umum digunakan dalam mesin pencari. Mesin pencari menggunakan *crawler* untuk mengumpulkan informasi yang terdapat pada halaman website sehingga ketika pengguna internet memasukkan kata kunci pencarian dapat dengan cepat memberikan informasi yang relevan kepada user.

Utami (2009) menyatakan bahwa sebuah *web crawler* akan berjalan menelusuri halaman *web* dan mengumpulkan dokumen-dokumen atau data-data di dalamnya. Selanjutnya *web crawler* akan membangun sebuah daftar indeks untuk memudahkan proses pencarian. Menurut Pinkerton (2000), arsitektur *web crawler* seperti pada gambar 2.3



Gambar 2.3 Arsitektur *web crawler*

Web crawler dimulai dengan sekumpulan URL, kemudian mendownload setiap halamannya, mendapatkan link dari setiap page yang dikunjungi kemudian mengulangi kembali proses *crawling* pada setiap link halaman tersebut. Proses *crawling* berlangsung terus sampai antrian URL kosong atau kalau kondisi berhenti sudah terpenuhi. Variasi algoritma dasar ini muncul dari kriteria pemilihan URL pada antrian yang akan diambil, sering juga disebut *crawling strategy* atau *crawler ordering* (Widyantoro, 2006). Penelusuran dapat difokuskan pada kedalaman (*depth-first spidering*) atau pada keluasan (*breadth-first spidering*) situs.

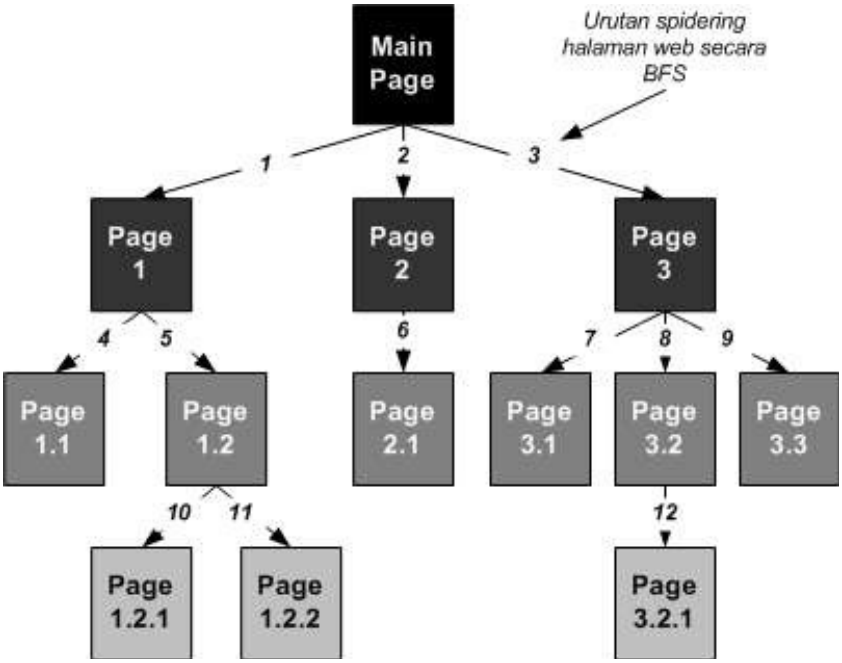
2.3.1 *Breadth First Spidering*

Ide dari *breadth first spidering* adalah proses penelusuran halaman web dimana seluruh halaman web yang berada disekitar halaman utama akan ditelusuri terlebih dahulu sebelum dilanjutkan ke halaman web berikutnya yang letaknya lebih jauh dari halaman utama tersebut (Gozali, 2004). *Breadth First spidering* atau BFS dimana halaman utama yang merupakan halaman web pada level-0 akan pertama diindeks. Pada halaman utama terdapat *link* ke tiga buah halaman pada level-1 yaitu *page-1*, *page-2* dan *page-3* yang akan diindeks berikutnya. Setelah proses *indexing* pada level-1 selesai barulah dilanjutkan pada level-2 dan selanjutnya.

Dalam mengikuti *link* pada sebuah alamat situs *web* yang dikunjungi, seberapa jauh kedalaman level yang diikuti *spider* tersebut diatur sendiri oleh pembuatnya. Ada *spider* yang mengindeks sebuah alamat situd *web* secara tuntas, ada yang mengikuti *link* tanpa menghiraukan letak dari halaman tersebut.

Beberapa *search engine* bahkan membatasi kedalaman *spidering* mereka untuk menghemat tempat penyimpanan yang dibutuhkan dan untuk menghindari

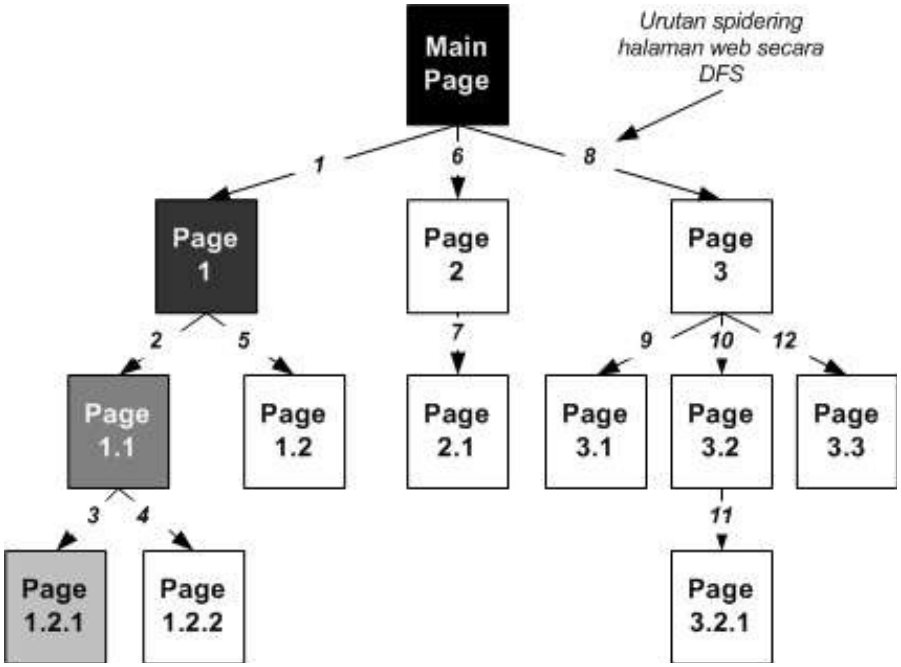
loop yang terjadi pada sebuah situs *web*. Kecepatan dan banyaknya informasi yang diperoleh *spider* juga dipengaruhi oleh kedalaman ini.



Gambar 2.4 *Breadth First Spidering*

2.3.2 Depth First Spidering

Alternatif lainnya adalah *depth first spidering.* Dalam hal ini, *spider* akan menelusuri halaman web dengan mengikuti *link* mulai dari *link* pertama pada halaman awal level 0 dilanjutkan pada *link* pertama pada level 1 dan begitu seterusnya sampai akhir dari *link* tersebut (Gozali, 2004). Selanjutnya proses secara *iterative* akan dilanjutkan dengan melakukan proses pengindeksan mulai dari link kedua yang berada pada halaman utama level 0 sampai seluruh link yang ada pada halaman utama level 0 tersebut selesai diindeks.



Gambar 2.5 *Depth First Spidering*

Kecepatan pertumbuhan informasi pada halaman web ini menyebabkan terbatasnya kemampuan dari *search engine* yang menggunakan kedua cara spidering tersebut diatas. Untuk mempercepat proses *spidering* serta memperbanyak informasi yang dimiliki pada database, beberapa situs bahkan memiliki lebih dari satu *spider*, misalnya, google.com memiliki beberapa *spider* yang bekerja sangat cepat dalam menumpulkan data tentang halaman situs *web*.

Cara lain yang dilakukan untuk mengatasi tersebut di atas adalah dengan melakukan *spidering* halaman web yang lebih menarik atau yang yang sesuai dengan suatu topik tertentu. Pendekatan ini disebut dengan istilah *focused spidering* atau *directed spidering*. *Focused spidering* mengeksplorasi halaman web yang difokuskan pada topik tertentu (Widyantoro, 2006). Pada *focused spidering* hanya sub set dari situs web yang akan diindeks yang sesuai dengan kriteria atau topik yang ditentukan.

Dalam melakukan *indexing* suatu halaman web, *crawler* harus dapat menentukan bagian-bagian dari dokumen yang dapat dijadikan untuk indeks. Diantaranya adalah *meta tags* atau *meta data*, yaitu suatu keterangan dalam halaman web yang tak terlihat pada *web browser*. *Meta data* ini yang menjelaskan isi dari halaman web tersebut dan berguna untuk membantu *automatic indexing*. *World Wide Web* (WWW) *Consortium* telah membuat daftar tentang *information*

and standardization proposals untuk *metadata*. Beberapa standar untuk *metadata* telah diumumkan diantaranya adalah *Dublin Core Metadata standard* dan *Warwick framework*.

Dublin Core Metadata terdiri dari 15 elemen yang ditujukan untuk memfasilitasi dan membantu mempercepat dan keakuratan otomatis indeks. Elemen-elemen itu adalah *title; creator; subject; description; publisher; contributors; date; resource type; format; resource identier; source; language; relation; coverage;* dan *rights.* Group ini juga telah membuat suatu cara untuk mempermudah penggabungan *metadata* ke dalam halaman web. Jika ini dilakukan secara keseluruhan dalam internet, dapat dipastikan *information retrieval* di web akan menjadi lebih akurat, tapi untuk dapat mengadopsi hal ini secara international merupakan hal yang sulit diwujudkan karena tidak semua situs web menggunakan standar tersebut.

2.4 *Indexing*

Indexing atau pengindeksan merupakan proses membangun basis data indeks dari koleksi dokumen (Bunyamin, 2008). *Indexing* dilakukan terhadap dokumen sebelum pencarian dilakukan. *Indexing* mencakup proses membaca file html, mengambil isinya, mengubahnya kedalam text biasa, menyimpulkan isi yang terkandung dalam file tersebut (Gozali, 2004).

Proses pengindeksan bisa secara manual ataupun secara otomatis. Adapun tahapan dari pengindeksan adalah sebagai berikut (Hasibuan, 2001):

1. *Parsing* dokumen yaitu proses pengambilan kata-kata dari kumpulan dokumen.
2. *Stoplist* yaitu proses pembuangan kata buang seperti: tetapi, yaitu, sedangkan, dan sebagainya.
3. *Stemming* yaitu proses penghilangan/pemotongan dari suatu kata menjadi bentuk dasar. Kata “diadaptasikan” atau “beradaptasi” mejadi kata “adaptasi” sebagai istilah.
4. *Term Weighting* dan *Inverted File* yaitu proses pemberian bobot pada istilah.

2.5 *Parsing*

Parser adalah sebuah sistem dasar yang memungkinkan pemahaman yang lebih baik terhadap kalimat dalam bahasa tertentu (Gusmita, 2008). Proses yang dilakukan oleh *parser* disebut *parsing. Parsing* merupakan proses pengambilan kata-kata dari kumpulan dokumen. Tujuan utama dari *parsing* adalah memeriksa apakah urutan token-token yang dihasilkan sesuai dengan tata bahasa dari bahasa yang bersangkutan (Suhartanto, 2006). Ada dua metode dalam *parsing* yaitu *parsing top-down* dan *parsing bottom-up*. *Parsing top-down* lebih mudah digunakan untuk pembentukan parser yang efisien. *Parsing bottom-up* dapat menangani tata bahasa yang lebih besar, tetapi implementasi *parser* ini lebih sukar dibandingkan *parsing top-down* (Suhartanto, 2006).

Parsing top-down dimulai dari simbol awal S sampai kalimat x. Ada 2 kelas metode *parsing top-down*, yaitu kelas metode dengan *backup* dan kelas metode tanpa *backup*. Contoh metode kelas dengan *backup* adalah metode *Brute-Force*, sedangkan contoh metode kelas tanpa *backup* adalah metode *recursive descent*.

a. Metode Brute-Force

Kelas metode dengan *backup*, termasuk metode *Brute-Force*, adalah kelas metode *parsing* yang menggunakan produksi alternatif, jika ada, ketika hasil penggunaan sebuah produksi tidak sesuai dengan simbol input (Himawan, 2011). Penggunaan produksi sesuai dengan nomor urut produksi. Metode *Brute-Force* tidak dapat menggunakan *grammar* rekursi kiri, yaitu grammar yang mengandung produksi rekursi kiri (*left recursion*). Produksi rekursi kiri akan menyebabkan *parsing* mengalami *looping* tak hingga. Dalam penelitian ini akan digunakan metode *Brute-Force*.

b. Metode Recursive-Descent

Kelas metode tanpa *backup*, termasuk metode *recursive descent*, adalah kelas metode *parsing* yang tidak menggunakan produksi alternatif ketika hasil akibat penggunaan sebuah produksi tidak sesuai dengan simbol input. Ketentuan produksi yang digunakan metode *recursive descent* adalah: jika terdapat dua atau lebih produksi dengan ruas kiri yang sama maka karakter pertama dari semua ruas kanan produksi tersebut tidak boleh sama.

2.6 *Query Expansion*

Teknik *Query expansion* digunakan dalam temu balik informasi. *Query expansion* merupakan suatu proses yang menambahkan sejumlah kata dari dokumen yang relevan terhadap *query* awal. Teknik *query expansion* bertujuan untuk meningkatkan pencarian pengguna dengan menambahkan *query* istilah baru pada *query* yang ada (Kanaan, 2008).

Terdapat tiga cara yang dapat digunakan dalam melakukan *query expansion* yakni: *manual*, *interactive*, dan *automatic* (Imran, 2009). *Manual query expansion* dan *interactive query expansion* memerlukan keterlibatan pengguna. Terkadang pengguna tidak dapat memberikan informasi yang cukup untuk melakukan *query expansion*, maka dibutuhkan suatu metode yang tidak memerlukan keterlibatan pengguna di dalamnya (*automatic*). *Automatic query expansion* (AQE) merupakan proses penambahan istilah atau frase pada *query* asli untuk meningkatkan kinerja temu balik tanpa intervensi dari pengguna (Imran, 2009).

Masalah utama dalam *query expansion* adalah bagaimana menghasilkan alternatif atau perluasan *query* untuk pengguna. Bentuk paling umum dari *query expansion* adalah analisa global menggunakan beberapa bentuk tesaurus (Manning, 2009). Untuk setiap istilah t dalam sebuah *query*, *query* dapat diperluas otomatis dengan sinonim dan kata yang berhubungan dengan t dari tesaurus.

2.7 *Natural Language Processing*

Bila dipandang dari sisi implementasi teknologinya maka pemrosesan bahasa lisan dan tulisan adalah sangat berbeda. Bahasa lisan lebih banyak melakukan pemrosesan bunyi atau suara, sedangkan bahasa tulisan lebih banyak melakukan pemrosesan symbol simbol tertulis. Teknologi yang berkaitan dengan pemrosesan bahasa alami ini sering disebut sebagai *speech and language technology*, *natural language processing technology*, *human language technology*, atau sering disingkat dengan teknologi bahasa, sedangkan dari segi keilmuan bidang ini dikenal sebagai bidang *natural language processing* atau *computational linguistic* (Utami, 2007).

Natural Language Processing (NLP) atau pengolahan bahasa alami merupakan salah satu bidang ilmu *Artificial Intelligence* (Kecerdasan Buatan) yang dikembangkan agar computer mengerti dan memahami bahasa alami yang diberikan dan member respon hasil pengolahan sesuai yang diinginkan (Handoko, 2009). NLP tidak bertujuan untuk mentransformasikan bahasa yang diterima dalam bentuk suara menjadi data digital dan/atau sebaliknya pula, melainkan bertujuan untuk memahami arti dari ucapan yang diberikan dalam bahasa alami dan memberikan respon yang sesuai, misalnya dengan melakukan suatu aksi tertentu atau menampilkan data tertentu (Suciadi, 2001).

Bahasa alami tumbuh secara alami untuk memenuhi kebutuhan komunikasi antar manusia. Bahasa alami tidak dirancang dengan memperhatikan berbagai kendala untuk kemudahan pemrosesan. Sebagai akibatnya, pemrosesan bahasa alami jauh lebih sulit dibandingkan bahasa buatan. Pemrosesan bahasa alami tidak mudah dilakukan. Beberapa karakteristik yang menyulitkan pemrosesan bahasa alami diantaranya adalah (Utami, 2007):

1. Sering terjadi ambiguitas dalam bahasa alami.

   Fenomena ini terjadi pada berbagai level implementasi bahasa, mulai dari simbol-simbol huruf dan tanda baca sebagai unit terkecil bahasa tulisan, tingkat kata, frasa, kalimat, bahkan paragraf. Simbol titik tidak selalu berfungsi sebagai tanda akhir kalimat, tetapi dapat menjadi bagian dari singkatan (misalnya Ir., Dr., Jl.) atau bagian dari bilangan. Contoh lainnya, kata “bisa” mungkin mempunyai pengertian “racun” atau “dapat”. Fenomena ini terjadi pula dalam penentuan jenis kata (part of speech), misalnya kata ”advanced” dapat berfungsi sebagai kata kerja aktif (bentuk lampau), kata kerja pasif, atau kata sifat.
2. Jumlah kosa kata dalam bahasa alami sangat besar dan berkembang dari waktu ke waktu.

2.8 *Stemming*

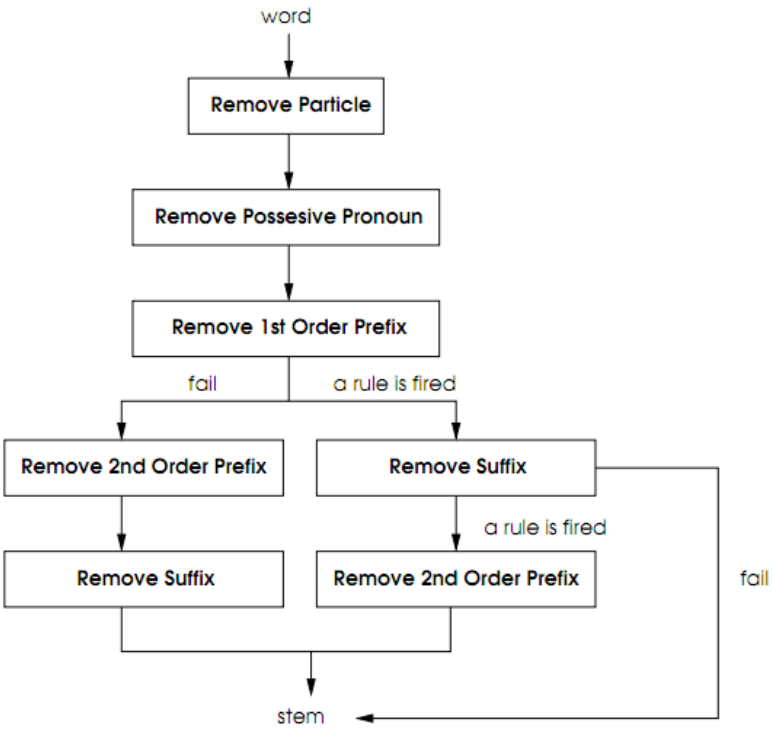
Stemming adalah salah satu cara yang digunakan untuk meningkatkan performa IR. *Stemming* digunakan untuk mencari kata dasar dari bentuk berimbuhan (Cholifah, TT). Algoritma *stemming* untuk bahasa yang satu berbeda

dengan algoritma *stemming* untuk bahasa lainnya. Sebagai contoh Bahasa Inggris memiliki morfologi yang berbeda dengan Bahasa Indonesia sehingga algoritma *stemming* untuk kedua bahasa tersebut juga berbeda. Proses *stemming* pada teks berbahasa Indonesia lebih rumit/kompleks karena terdapat variasi imbuhan yang harus dibuang untuk mendapatkan *root word* dari sebuah kata. Beberapa algoritma *stemming* Bahasa Indonesia telah dikembangkan sebelumnya. Penggunaan algoritma *stemming* yang sesuai mempengaruhi performa sistem IR (Agusta, 2009).

Algoritma *stemming* untuk beberapa bahasa telah dikembangkan, seperti Algoritma Porter untuk teks berbahasa inggris, Algoritma Porter untuk teks berbahasa Indonesia yang dikembangkan oleh Tala, Algoritma Nazief & Adriani untuk teks berbahasa Indonesia.

Sebuah penelitian oleh Agusta (2009) menyimpulkan bahwa proses stemming dokumen teks berbahasa Indonesia menggunakan Algoritma Porter membutuhkan waktu yang lebih singkat dibandingkan dengan stemming menggunakan Algoritma Nazief & Adriani. Oleh karena itu, dalam penelitian ini akan digunakan Algoritma Porter untuk teks berbahasa Indonesia yang dikembangkan oleh Tala. Adapun langkah-langkah algoritma ini adalah sebagai berikut (Tala, TT):

1. Hapus *Particle*.
2. Hapus *Possesive Pronoun* (kata ganti milik).
3. Hapus awalan pertama. Jika tidak ada lanjutkan ke langkah 4a, jika ada cari maka lanjutkan ke langkah 4b.
4. a. Hapus awalan kedua, lanjutkan ke langkah 5a.
   b. Hapus akhiran, jika tidak ditemukan maka kata tersebut diasumsikan sebagai *root word*. Jika ditemukan maka lanjutkan ke langkah 5b.
5. a. Hapus akhiran. Kemudian kata akhir diasumsikan sebagai *root word*.
   b. Hapus awalan kedua. Kemudian kata akhir diasumsikan sebagai *root word*.



Gambar 2.6 Porter *Stemming* untuk Bahasa Indonesia

Terdapat 5 kelompok aturan pada Algoritma Porter untuk Bahasa Indonesia ini. Aturan tersebut dapat dilihat pada Tabel 2.1 sampai Tabel 2.5.

Tabel 2.1 Kelompok Rule Pertama: *Inflectional Particles*

Suffix	Replacement	Measure Condition	Additional Condition	Examples
kah	NULL	2	NULL	bukukah → buku
lah	NULL	2	NULL	adalah → ada
pun	NULL	2	NULL	bukupun → buku

Tabel 2.2 Kelompok Rule Kedua: *Inflectional Possesive Pronouns*

Suffix	Replacement	Measure Condition	Additional Condition	Examples
ku	NULL	2	NULL	bukuku → buku
mu	NULL	2	NULL	bukumu → buku
nya	NULL	2	NULL	bukunya → buku

Tabel 2.3 Kelompok Rule Ketiga: *First Order of Derivational Prefixes*

Prefix	Replacement	Measure Condition	Additional Condition	Examples
meng	NULL	2	NULL	mengukur → ukur
meny	s	2	V...*	menyapu → sapu
men	NULL	2	NULL	menduga → duga menuduh → uduh
mem	p	2	V...	memilah → pilah
mem	NULL	2	NULL	membaca → baca
me	NULL	2	NULL	merusak → rusak
peng	NULL	2	NULL	pengukur → ukur
peny	s	2	V...	penyapu → sapu
pen	NULL	2	NULL	penduga → duga penuduh → uduh
pem	p	2	V...	pemilah → pilah
pem	NULL	2	NULL	pembaca → baca
di	NULL	2	NULL	diukur → ukur
ter	NULL	2	NULL	tersapu → sapu
ke	NULL	2	NULL	kekasih → kasih

Tabel 2.4 Kelompok Rule Keempat: *Second Order of Derivational Prefixes*

Prefix	Replacement	Measure Condition	Additional Condition	Examples
ber	NULL	2	NULL	berlari → lari
bel	NULL	2	ajar	belajar → ajar
be	NULL	2	K*er...	bekerja → kerja
per	NULL	2	NULL	perjelas → jelas
pel	NULL	2	ajar	pelajar → ajar
pe	NULL	2	NULL	pekerja → kerja

Tabel 2.5 Kelompok Rule Kelima: *Derivational Suffixes*

Suffix	Replacement	Measure Condition	Additional Condition	Examples
kan	NULL	2	prefix $\notin$ {ke, peng}	tarikkan → tarik (meng)ambilkan → ambil
an	NULL	2	prefix $\notin$ {di, meng, ter}	makanan → makan (per)janjian → janji
i	NULL	2	V|K...c_1c_1, $c_1 \neq s, c_2 \neq i$ and prefix $\notin$ {ber, ke, peng}	tandai → tanda (men)dapati → dapat pantai → panta

2.9 Pembobotan Kata

Dalam pencarian yang dilakukan oleh sistem temu balik informasi, semua istilah yang dicari tidak memiliki bobot yang sama. Untuk itulah dibutuhkan metode pembobotan kata agar pencarian lebih mudah difokuskan. *Term weighting* atau pembobotan kata yaitu setiap kata dalam dokumen diberikan bobot berdasarkan banyaknya kata pada tiap dokumen (Wibowo, 2010).

Bunyamin (2005) menyatakan bahwa salah satu cara untuk memberi bobot terhadap suatu kata adalah memberikan nilai jumlah kemunculan suatu kata (*term frequency*) sebagai bobot. Semakin besar kemunculan suatu kata dalam dokumen akan memberikan nilai kesesuaian yang semakin besar. Faktor lain yang diperhatikan dalam pemberian bobot adalah kejarangmunculan kata (*term scarcity*) dalam koleksi. Kata yang muncul pada sedikit dokumen harus dipandang sebagai kata yang lebih penting (*uncommon terms*) daripada kata yang muncul pada banyak dokumen. Pembobotan akan memperhitungkan faktor kebalikan frekuensi dokumen yang mengandung suatu kata (*inverse document frequency*).

Dalam penelitian ini, akan digunakan metode pembobotan TF-IDF (*term frequency-inverse document frequency*). Mengenai efektivitas kinerja dari sebuah *search engine* selalu dikaitkan dengan tingkat relevansi hasil pencarian. Untuk menemukan dokumen yang relevan, metode pembobotan TF-IDF memberikan bobot lebih kepada istilah yang lebih penting. Istilah yang lebih penting yang dimaksud adalah istilah yang jika muncul pada sebuah dokumen maka dokumen tersebut dapat dianggap relevan dengan *query*.

Term Frequency (TF) adalah algoritma pembobotan heuristik yang menentukan bobot dokumen berdasarkan kemunculan term (istilah). Semakin sering sebuah istilah muncul, semakin tinggi bobot dokumen untuk istilah tersebut, dan sebaliknya. Terdapat empat buah algoritma TF yaitu *Raw TF, Logarithmic TF, Binary TF, Augmented TF* (Safriadi, 2011). Dalam penelitian ini digunakan algoritma *Raw TF*. *Raw TF* diperoleh dari perhitungan frekuensi kemunculan suatu istilah pada dokumen.

IDF atau *Inverse Document Frequency* merupakan banyaknya istilah tertentu dalam keseluruhan dokumen. Pembobotan IDF dapat dihitung dengan persamaan:

$$idfj = \log \frac{n}{nj} \quad (1)$$

n = jumlah seluruh dokumen
nj = jumlah dokumen yang mempunyai istilah j

2.10 Mesin Pencari (*Search Engine*)

Salah satu aplikasi dari sistem temu kembali informasi adalah *search engine* atau mesin pencari yang terdapat pada jaringan internet. Pengguna dapat mencari halaman web yang dibutuhkan melalui *search engine*. *Search engine* tidak lain sebuah mesin pencari yang ulet dan teliti, yang melakukun eksplorasi atas informasi-informasi yang di-request tanpa memandang kapan, di mana dan oleh siapa itu dilakukan (Rafiudin, 2003).

Mesin pencari memungkinkan untuk meminta content media dengan kriteria yang spesifik (biasanya yang berisi kata atau frasa yang kita tentukan) dan memperoleh daftar file yang memenuhi kriteria tersebut (Soleh, 2010). Mesin pencari menggunakan indeks (yang sudah dibuat dan disusun secara teratur) untuk mencari file setelah pengguna memasukkan kriteria pencarian. Informasi yang ditampilkan mengandung atau berhubungan dengan suatu istilah spesifik.

2.11 JSP (*JavaServer Pages*) dan *Servlet*

Untuk membangkitkan halaman web sesuai dengan permintaan pemakai, para pengembang aplikasi web bisa menggunakan perangkat lunak seperti PHP, JSP, Perl dan ASP. Menurut Kadir (2004), *JavaServer Pages* (JSP) merupakan teknologi yang didasarkan pada bahasa java, yang dapat membentuk halaman-halaman web yang bersifat dinamis. Menurut Rickyanto (2002), JSP merupakan dokumen berbasis teks yang berjalan sebagai *servlet* tetapi mengizinkan pendekatan yang lebih alami untuk membuat *static content*. Sedangkan *servlet* merupakan *class* dalam bahasa pemrograman java yang secara dinamis memproses permintaan dan membangun tanggapan. Java sendiri didefinisikan sebagai teknologi dimana teknologi tersebut mencakup java sebagai bahasa

pemrograman yang memiliki sintaks dan aturan pemrograman sendiri, juga mencakup java sebagai platform dimana teknologi itu memiliki *virtual machine* dan *library* yang diperlukan untuk menulis dan menjalankan program yang ditulis dengan bahasa pemrograman java (Rickyanto, 2005).

Dua alasan penting yang membuat JSP banyak digunakan oleh para pengembang aplikasi web (Kadir, 2004):

1. JSP menggunakan bahasa java. Bagi para pemrogram yang telah mengenal java, membuat aplikasi web dengan JSP sangat mudah mengingat dasar JSP adalah bahasa java.
2. JSP mendukung multiplatform. JSP memang bukan satu-satunya perangkat lunak pembuat aplikasi web yang bersifat multiplatform. PHP, misalnya, juga bersifat multiplatform. Keunggulan dari dukungan adanya multiplatform adalah memungkinkan kode dapat dipindah-pindahkan ke berbagai platform tanpa perlu melakukan perubahan apapun pada kode tersebut. Sebagai contoh, kode JSP yang pada awalnya ditujukan untuk dijalankan pada Windows, dan kemudian dipindahkan ke lingkungan lain, misalnya Linux.

Detail pemrosesan oleh JSP *servlet engine* (Kadir, 2004) adalah sebagai berikut:

1. Melakukan pemilahan (*parsing*) kode JSP,
2. membangkitkan kode sumber *servlet*,
3. mengkompilasi kode sumber *servlet* menjadi sebuah kelas,
4. membuat instan *servlet*,
5. memberikan keluaran *servlet* ke web browser.

Kelebihan JSP dibandingkan teknologi web lain, yang menjadi pertimbangan sebagai bahasa pemrograman untuk aplikasi web antara lain:

a. Memisahkan presentasi statis dan isi yang dinamik

   Untuk menghasilkan aplikasi web tentunya *web developer* harus berurusan dengan tag-tag HTML maupun XML untuk menghasilkan halaman web. Agar menghasilkan tampilan halaman web yang baik, tentunya *web developer* juga harus mendesainnya sehingga harus menentukan kode HTML yang ingin dihasilkan. Dengan teknologi JSP hal ini dapat dilakukan, dimana dengan JSP maka *web programmer* dapat menyisipkan tag atau skriplet dengan data atau

isi dinamik yang akan ditampilkan pada bagian-bagian dari halaman web yang telah didesain.

b. Menekankan komponen *reusable*

Teknologi JSP memerlukan komponen yang *reusable* untuk melakukan pemrosesan yang lebih kompleks. Dengan komponen (*Javabeans*), *developer* dapat menggunakannya untuk operasi yang umum sehingga memungkinkan *sharing* dan distribusi komponen kepada publik atau komunitas di internet. Penggunaan komponen dapat mempercepat pembuatan aplikasi web karena proses logik yang diperlukan sudah tersedia dan langsung dapat digunakan.

c. Berbasis bahasa pemrograman java

Oleh karena JSP berbasis Java, maka aplikasi yang dibuat dengan JSP juga memiliki manajemen memori dan sekuritas yang baik. Selain itu, JSP juga mudah dipelajari dan dapat memanfaatkan pemrograman berorientasi objek dari java.

d. Bagian dari platform java

Oleh karena merupakan bagian dari platform Java, maka JSP juga memiliki karakteristik "*Write Once, Run Anywhere*" yaitu portabilitas yang tinggi.

e. Terintegrasi dengan *Java 2 Enterprise Edition* (J2EE)

Oleh karena JSP merupakan bagian integral J2EE, maka aplikasi JSP dapat dikembangkan ke aplikasi berskala Enterprise.

Servlet yang merupakan salah satu teknologi unggulan masa depan memiliki kelebihan-kelebihan sebagai berikut (Rickyanto, 2004):

1. Efisien dan baik dalam *performance*

Performance servlet baik dan efisien karena tidak ada proses pembuatan berulang untuk tiap request dari client. Setiap request ditangani oleh proses *servlet container*. *Servlet* telah dibuat dan dihancurkan berulang-ulang, tetapi tetap tersimpan pada memori untuk menangani request lain yang datang selanjutnya.

2. *Powerful*

Servlet memiliki kemampuan yang lengkap, antara lain mampu melakukan penanganan permintaan ke permintaan, penanganan *cookie* dan *session*, akses

database dengan *Java Database Connectivity* (JDBC), *caching*, serta *library* yang lengkap untuk pembuatan aplikasi web.

3. Aman

   Servlet memiliki fasilitas *security* yang baik dan merupakan bagian dari teknologi java yang sudah dari asalnya didesain dengan *security* yang baik.

4. Portabilitas

   Teknologi *java servlet* portabel karena dijalankan di berbagai *servlet container*, *application server*, maupun system operasi.

5. Proses pengembangan yang lebih cepat

   Dengan menggunakan *servlet, library java* yang lengkap dan komponen atau bean yang sudah ada dapat digunakan.

6. *Robust*

   Java mampu mencegah adanya *memory leaking* dan dapat mengidentifikasi *error* sebelum program dijalankan.

7. Tangguh

   Servlet merupakan teknologi java yang memiliki penanganan memori yang baik dan *garbage collection*, sehingga menjadi aplikasi web yang tangguh dan stabil.

8. Telah digunakan dan diakui di dunia

   Servlet merupakan teknologi java yang telah diterima dan digunakan di berbagai belahan dunia. Komponen, solusi dan dukungan yang ditawarkan baik yang gratis (*open source*) maupun komersial dapat ditemukan.

9. Murah

   Murah karena (*Java Development Kit*) JDK Java gratis untuk diunduh. *Servlet* dan *JSP Container* juga banyak yang gratis, misalnya Tomcat.

Servlet bekerja dengan dimuat ke *Java Virtual Machine* (JVM) oleh *servlet container* apabila terjadi permintaan pertama kali oleh *client*. Proses penanganan permintaan dijalankan sebagai *thread* dari web server atau *servlet container*. Setelah dimuat, maka *servlet* tetap ada di memori untuk menangani permintaan berikutnya. Setiap kali menangani permintaan, *servlet container* membandingkan *timestamp* dari *servlet* dalam memori dengan file *class java servlet*. Apabila

timestamp file java *servlet* lebih baru, maka secara otomatis *servlet container* akan memuat *servlet* yang baru dari *class servlet*.

Servlet secara normalnya bekerja pada lingkungan *multithreading*. Suatu *web container* dapat menghasilkan *instance servlet* yang memiliki banyak *thread*. Permintaan yang terjadi ditangani oleh *thread* dari *instance* tersebut. Sebenarnya *servlet* dapat diatur supaya bekerja dalam lingkungan *Single ThreadModel*, tetapi pada *Servlet 2.4* mekanisme ini mengalami *deprecation* yang berarti tidak didukung lagi pada versi *servlet* berikutnya.

2.12 Pengujian Perangkat Lunak

Pengujian merupakan kegiatan penting dalam rekayasa perangkat lunak. Pengujian digunakan untuk mengamati pelaksanaan sistem perangkat lunak dalam memvalidasi apakah berperilaku sebagaimana dimaksud dan mengidentifikasi potensi kerusakan. Pengujian secara luas digunakan dalam industri sebagai jaminan kualitas, dengan langsung mengamati perangkat lunak dalam pelaksanaan, memberikan umpan balik secara realistis dan melengkapi teknik analisis lainnya. *Pervasiveness*, kompleksitas dan kekritisan perangkat lunak tumbuh tak henti-hentinya, memastikan bahwa ia berperilaku sesuai dengan tingkat kualitas yang diinginkan dan kehandalan menjadi lebih penting, semakin sulit dan mahal (Bertolino, 2007).

Glen Myers menyatakan sejumlah aturan yang berfungsi sebagai sasaran pengujian (Pressman, 2002):

1. Pengujian adalah proses eksekusi program dengan tujuan untuk menemukan kesalahan.
2. *Test case* yang baik adalah *test case* yang memiliki probabilitas tinggi untuk menemukan kesalahan yang belum pernah ditemukan sebelumnya.
3. Pengujian yang sukses adalah pengujian yang mengungkap semua kesalahan yang belum ditemukan sebelumnya.

Sehingga tujuan dari pengujian adalah mendesain pengujian yang secara sistematis mengungkap kelas kesalahan yang berbeda dan melakukannya dengan jumlah waktu dan usaha yang minimum. Jika pengujian dilakukan secara sukses, maka akan ditemukan kesalahan didalam perangkat lunak, manfaat lain dari

pengujian adalah menunjukkan bahwa fungsi perangkat lunak telah bekerja sesuai dengan spesifikasi, dan persyaratan kinerja telah dipenuhi. Sebagai tambahan, data yang dikumpulkan pada saat pengujian dilakukan memberikan indikasi yang baik mengenai reliabilitas perangkat lunak dan beberapa menunjukkan kualitas perangkat lunak secara keseluruhan. Tetapi pengujian tidak dapat memperlihatkan tidak adanya cacat, pengujian hanya dapat memperlihatkan bahwa ada kesalahan pada perangkat lunak.

2.12.1 *White Box Testing*

Pengujian *white box* merupakan pendekatan terhadap pengujian yang diturunkan dari pengetahuan struktur dan implementasi perangkat lunak. Pengujian *white box* biasanya diterapkan untuk unit program yang relative kecil seperti subrutin atau operasi yang terkait dengan suatu objek (Sommerville, 2003).

Dengan menggunakan metode pengujian *white box*, perekayasa sistem dapat melakukan *test case* yang (Pressman, 2002):

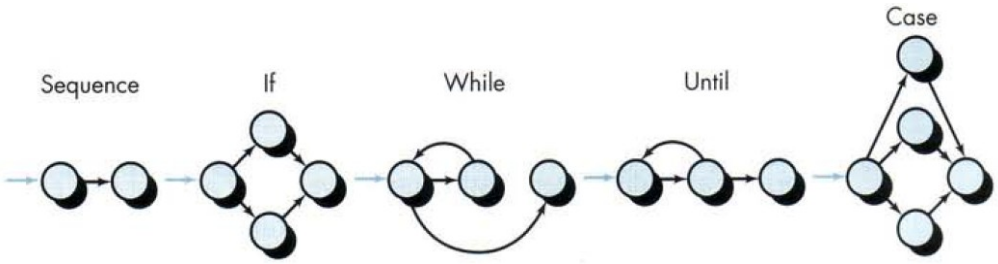
1. Menjamin bahwa seluruh independent *paths* dalam modul telah dilakukan sedikitnya satu kali,
2. melakukan seluruh keputusan logikal baik pada sisi *true* maupun *false*,
3. melakukan seluruh perulangan sesuai batasannya dan batasan operasionalnya,
4. menguji struktur data internal untuk menjamin validitasnya.

Pengujian *basis path* adalah pengujian *white box* yang diusulkan pertama kali oleh Tom McCabe. Metode ini memungkinkan pengukuran kompleksitas logis dari desain prosedural dan menggunakannya sebagai pedoman untuk menetapkan *basis set* dari semua jalur eksekusi. Konsep utama *basis path* yaitu tiap *basis path* harus diidentifikasi, tidak boleh ada yang terabaikan (setidaknya dites 1 kali) dan kombinasi dan permutasi dari suatu *basis path* tidak perlu dites.

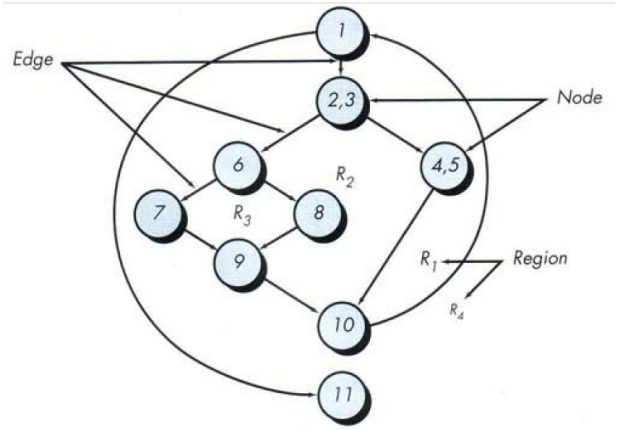
1. Notasi Diagram Alir

   Notasi yang digunakan untuk menggambarkan jalur eksekusi adalah notasi diagram alir (atau grafik program), yang menggunakan notasi lingkaran (simpul atau *node*) dan anak panah (*link* atau *edge*). Notasi ini menggambarkan aliran kontrol logika yang digunakan dalam suatu bahasa pemrograman. Berdasarkan gambar 2.7 dan 2.8, tiap lingkaran, disebut *flow graph node*, yang mewakili satu atau lebih pernyataan prosedural. Suatu

proses yang berurutan yang digambarkan dalam bentuk kotak pada *flow chart* atau suatu keputusan yang digambarkan dalam bentuk belah ketupat pada *flow chart* dapat diwakili oleh satu *node*. panah pada *flow graph*, disebut *edges* atau *links* (hubungan), mewakili alur pengiriman kendali dan merupakan analogi dari panah pada *flow chart*. Suatu *edge* harus diakhiri dengan suatu *node*, bahkan bilamana *node* tersebut tidak mewakili suatu pernyataan prosedural sekalipun. Area yang dibatasi oleh *edges* dan *nodes* disebut *regions*. Bila menghitung *regions*, harus juga mengikutkan area di luar dari grafik sebagai bagian dari *regions*.



Gambar 2.7 Notasi Diagram Alir



Gambar 2.8 Diagram Alir

2. Kompleksitas Siklomatis (*Cyclomatic Complexity*)

   Kompleksitas siklomatis adalah matriks perangkat lunak yang memberikan pengukuran kuantitatif terhadap kompleksitas logis suatu program, nilai yang didapat akan menentukan jumlah jalur independen dalam himpunan path, serta akan memberi nilai batas atas bagi jumlah pengujian yang harus dilakukan, untuk memastikan bahwa semua pernyataan telah dieksekusi sedikitnya satu kali. Jalur independen adalah jalur yang terdapat dalam program yang mengintroduksi sedikitnya satu rangkaian pernyataan proses atau kondisi baru.

Adapun penghitungan siklomatis dapat diperoleh dari:

- *Region* (daerah muka) grafik alir
- CC = E - N + 2

  E adalah jumlah *edge,* dan N adalah jumlah *node*
- CC = P +1

  P adalah simpul predikat

Simpul predikat adalah penggambaran suatu node yang memiliki satu atau lebih masukan (*input*), dan lebih dari satu keluaran (*output*).

3. Matriks Grafis

   Bentuk struktur data yang sering digunakan untuk mengambarkan pengujian adalah dengan matriks grafis. Matriks grafis adalah matriks bujursangkar yang berukuran sama dengan jumlah simpul pada grafik alir. Inputan dalam matriks harus bersesuaian dengan arah sisi dengan simpul. Matriks grafis selanjutnya disebut sebagai matriks koneksi, dan digambarkan serupa dengan matriks ketetanggaan dengan memperhatikan arah *in-out* dari *edge*.

2.12.2 Evaluasi Retrieval

Evaluasi suatu model *retrieval* oleh suatu sistem IR yang paling umum adalah ukuran *Recall* dan *Precision* (Rijsbergen, 1979). Safriadi (2011) menyatakan bahwa dokumen-dokumen yang ditampilkan oleh sistem temu balik informasi harus memenuhi persyaratan recall, Precision dan NIAP (*Non Interpolated Average Precision*). Recall didefinisikan dengan menemukan seluruh dokumen yang relevan dalam koleksi dokumen. Recall dapat dihitung dengan persamaan:

$$Recall = \frac{jumlah\,dokumen\,relevan\,ditemukan}{jumlah\,dokumen\,relevan\,dalam\,koleksi} \qquad (2)$$

Nilai recall tertinggi adalah 1, yang berarti seluruh dokumen dalam koleksi berhasil ditemukan.

Precision didefinisikan dengan menemukan hanya dokumen yang relevan saja dalam koleksi. Precision dapat dihitung dengan persamaan:

$$Precision = \frac{jumlah\,dokumen\,relevan\,ditemukan}{jumlah\,dokumen\,ditemukan} \qquad (3)$$

Nilai precision tertinggi adalah 1, yang berarti seluruh dokumen yang ditemukan adalah relevan.

NIAP adalah penggabungan dari recall dan precision, yang dapat dihitung dengan persamaan:

$$NIAP = \sum_{i=1}^{n} \frac{Precision\ pada\ dokumen\ ke-i}{Jumlah\ dokumen\ relevan\ dalam\ koleksi} \quad (4)$$

Di mana *n* menunjukkan jumlah dokumen yang dicari hingga seluruh dokumen relevan ditemukan. NIAP menghitung kinerja sistem berdasarkan rata-rata presisinya saat suatu dokumen relevan ditemukan (Mandala, 2006). Jadi saat suatu dokumen relevan ditemukan, nilai presisi dokumen tersebut akan dihitung. Kemudian seluruh nilai presisi tersebut akan dijumlahkan dan dibagi dengan jumlah dokumen relevan dalam koleksi dokumen. Nilai NIAP tertinggi adalah 1, yang berarti seluruh dokumen relevan berhasil ditemukan dengan seluruh dokumen relevan tersebut ditempatkan pada urutan teratas dalam hasil pencarian. Nilai NIAP akan digunakan untuk mengecek kebenaran hasil pencarian dari perangkat lunak yang dibangun.

2.13 Telaah Pustaka

Dalam penelitian ini banyak digunakan pustaka sebagai bahan referensi dan acuan. Beberapa penelitian diantaranya:

1. Aplikasi Server *Crawling* Berita Online Pada *Handphone* yang Mendukung WAP (Badrullami, 2010)

   Aplikasi yang dibuat dalam penelitian ini berupa aplikasi *crawling* dengan memanfaatkan RSS dari situs berita dan mengkategorisasi berita dengan *text mining* dan analisis korelasi. Pada aplikasi ini, *crawling* dilakukan hanya pada *website* penyedia berita yang memiliki fitur RSS dan berita berbahasa Indonesia saja. Penelitian ini memiliki output bahwa pengaksesan berita pada server WML menjadi lebih cepat apabila dibandingkan langsung ke situs aslinya karena sistem meminimalisir image yang ditampilkan pada server, sehingga load data dari server menuju klien menjadi lebih lancer dan analisis korelasi mampu mengkategorikan berita yang didapat dengan nilai error 20% sampai dengan 30%.

2. Uji Relevansi dan Performansi Sistem Temu Balik Informasi pada Giggle *Search Engine* (Safriadi, 2011)

   Dalam penelitian ini, dibuat mesin pencari bernama "Giggle *Search Engine*" yang menggunakan beberapa metode pembobotan berupa metode TF, IDF, Normalisasi dan kombinasi dari ketiga metode tersebut. Giggle *Search Engine* kemudian diuji performansinya dengan menghitung *recall, precision* dan NIAP terhadap dataset ADI dan CISI. Penelitian ini memiliki output bahwa metode pembobotan yang memiliki performansi paling tinggi adalah *Raw Term Frequency* untuk koleksi dokumen yang besar dan metode Normalisasi untuk koleksi dokumen yang kecil.

3. *Query Expansion* dengan Menggabungkan Metode Ruang Vektor dan Wordnet pada Sistem *Information Retrieval* (Nugroho, 2009)

   Dalam penelitian ini, diimplementasikan perluasan terhadap vektor *query*. Perluasan dilakukan dengan menggunakan WordNet pada istilah-istilah penyusun *query* agar hasil dari sistem dapat ditingkatkan. Penelitian ini hanya membahas query berbahasa Inggris. Penelitian ini memiliki output bahwa penggunaan *query expansion* berhasil meningkatkan jumlah dokumen yang diterima oleh sistem namun tidak menaikkan nilai *precision* karena rangking dokumen relevan yang dikembalikan turun karena semakin banyak dokumen tidak relevan yang diterima oleh sistem dan penggunaan sinonim dari WordNet untuk memperluas *query* dengan mengambil *part of speech noun* bagian sinonim tidak membantu dalam meningkatkan nilai *precision*.

5. Mesin Pencari Dokumen Teks (Februariyanti, 2010)

   Penelitian ini menggunakan algoritma indeks *inverted* untuk proses indeks kata (*term*), cosine similaritas untuk menghitung kesamaan kata dalam dokumen. Dokumen yang digunakan pada penelitian adalah dokumen teks abstrak skripsi mahasiswa. Hasil uji menunjukan bahwa algoritma dapat digunakan untuk menghitung tingkat similaritas (kesamaan) dokumen berdasarkan kata kunci yang diinputkan oleh pengguna. Pemilihan kata kunci sangat mempengaruhi hasil pencarian dokumen teks.

6. *Using a Query Expansion Technique to Improve Document Retrieval* (Aly, 2008)

   Penelitian ini mencoba mengimplementasikan teknik *query expansion* untuk meningkatkan temu balik dokumen. Penelitian ini menggunakan 3 dataset yaitu CISI, NPL dan CACM. Penelitian ini memiliki output bahwa efektivitas temu balik (*retrieval effectiveness*) sangat tinggi ketika teknik *query expansion* digunakan.

BAB III
METODOLOGI PENELITIAN

3.1 Bahan Penelitian

Bahan penelitian yang digunakan berupa berita hasil *crawling* website penyedia berita yang telah ditentukan sebelumnya yaitu www.detik.com, www.antaranews.com, pontianak.tribunnews.com.

3.1.1 Alat yang Dipergunakan

Alat yang digunakan dalam penelitian ini berupa alat penelitian, perangkat lunak dan perangkat keras. Adapun alat yang dipergunakan adalah sebagai berikut.

1. Alat Penelitian

   Alat penelitian yang digunakan dalam penelitian ini adalah:

   - *Entity Relationship Diagram (ERD)*, untuk menjelaskan hubungan antardata dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antarrelasi.
   - *Data Flow Diagram (DFD)*, untuk menggambarkan aliran data pada sistem yang terdiri dari dua bagian utama yaitu sistem input data dan sistem analisis data.
   - Metode *Crawling* Dokumen, untuk menggambarkan proses dan komponen yang digunakan dalam *crawling* dokumen.
   - Metode *Indexing* Dokumen, untuk menggambarkan proses dan komponen yang digunakan dalam *indexing* dokumen. Dalam *indexing* dokumen digunakan metode pembobotan kombinasi TF-IDF untuk menghasilkan index yang akan digunakan dalam pencarian.
   - Metode *Searching* Dokumen, untuk menggambarkan proses dan komponen yang digunakan dalam *searching* dokumen.

2. Perangkat Lunak

   Perangkat lunak yang digunakan dalam penelitian ini adalah:

   - Sistem operasi Windows 7 Professional
   - Apache Tomcat Version 6.0.26 sebagai *web server*
   - Bahasa pemograman Java dengan *interface* JSP

- Basisdata MySQL 5.0.27
- Notepad++ sebagai aplikasi untuk *source code editor*
- Netbeans 6.9.1

3. Perangkat Keras

Perangkat keras yang digunakan dalam penelitian ini adalah

1. Laptop sebagai client dengan spesifikasi:
   - Intel Pentium Core 2 Duo 2.2 GHz
   - Harddisk 320 GB
   - RAM DDR2 2.00 GB
2. Personal Komputer sebagai server dengan spesifikasi:
   - Intel Core 2 Duo 1.86 GHz
   - Harddisk 150 GB
   - RAM DDR2 2.00 GB

3.1.2 Metode Penelitian

Metode penelitian yang akan dilakukan antara lain yaitu:

1. Studi Literatur

   Studi literatur dilakukan untuk memperoleh informasi mengenai sistem temu balik informasi, gambaran sistem yang ada dan yang akan diimplementasikan dalam sistem yang akan dirancang serta memperoleh data-data yang diperlukan untuk tugas akhir ini.

2. Analisis Sistem

   Analisis sistem yang dilakukan mencakup analisis mengenai skema *crawling* dokumen, skema *indexing* dokumen dan skema pencarian dokumen berdasarkan hasil studi literarur yang telah dilakukan.

3. Metode perancangan aplikasi

   Metode perancangan aplikasi terdiri dari perancangan arsitektur sistem, perancangan proses menggunakan *Data Flow Diagram (DFD)*, perancangan basis data menggunakan *Entity Relationship Diagram* (*ERD*), serta perancangan antarmuka.

4. Metode pengujian aplikasi

Metode pengujian yang digunakan adalah metode *white box* dan evaluasi *retrieval*. Metode white box digunakan untuk menguji validitas proses yang terdapat dalam aplikasi. Evaluasi *retrieval* digunakan untuk mengetahui kinerja aplikasi dalam menemukan dokumen yang relevan.

3.1.3 Variabel atau Data

1. Data Primer

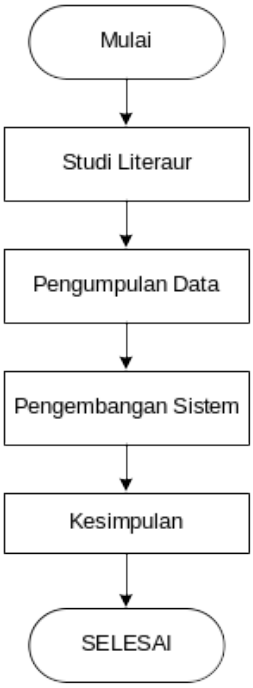
Data primer adalah data yang pertama kali diperoleh secara langsung dari objek penelitian. Dalam penelitian ini, yang menjadi data primer yaitu data yang diperoleh dari beberapa situs berita yang telah ditentukan sebelumnya.

2. Data Sekunder

Data sekunder adalah data yang tidak diperoleh langsung dari objek penelitian, yaitu dari studi pustaka dan referensi mengenai ilmu pengetahuan yang mendukung penelitian.

3.1.4 Bagan Alir Penelitian dan Pengembangan Sistem

Langkah-langkah penelitian yang akan dilakukan dapat dilihat pada Gambar 3.1.



Gambar 3.1 Bagan Alir Penelitian

a. Studi literatur

Studi literatur dilakukan guna memperoleh teori-teori pendukung serta kemungkinan asumsi yang digunakan dan berperan sebagai referensi dalam mencari pendekatan secara teoritis dari permasalahan yang diangkat yang bersumber antara lain pada buku atau bahan pustaka, karya ilmiah, *website* dan lain sebagainya.

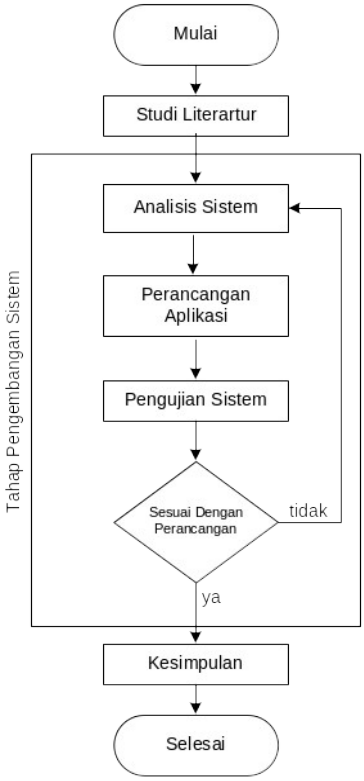
b. Pengumpulan data

Pengumpulan data dilakukan untuk memperoleh informasi mengenai gambaran sistem yang ada dan yang akan diterapkan dalam sistem yang akan dirancang serta memperoleh data yang diperlukan.

c. Pengembangan Sistem

Pengembangan sistem dilakukan untuk menyusun dan mengimplementasikan sistem agar sesuai dengan kebutuhan pengguna. Pengembangan sistem terdiri atas beberapa tahap yaitu analisis sistem, perancangan aplikasi, pengujian.

Langkah-langkah pengembangan sistem yang akan dilakukan dapat dilihat pada Gambar 3.2.



Gambar 3.2 Bagan Alir Pengembangan Sistem

Langkah- langkah dalam pengembangan sistem yang akan dilakukan dapat dijelaskan sebagai berikut:

a. Analisis Sistem

Analisis sistem dilakukan berdasarkan hasil observasi dan pengumpulan data yang dilakukan. Analisa kebutuhan sistem dilakukan untuk menentukan fitur-fitur apa saja yang terdapat pada sistem serta merancangnya agar sesuai dengan kebutuhan.

b. Perancangan Aplikasi

Perancangan aplikasi berupa perancangan *interface*, perancangan basis data menggunakan *Entity Relationship Diagram* (*ERD*), serta perancangan proses dan *output*.

c. Pengujian Sistem

Sistem yang telah diimplementasi kemudian diuji. Apabila *output* data yang dihasilkan oleh sistem tidak sesuai dengan perancangan maka akan dianalisa dan dilakukan perancangan kembali.

3.2 Perancangan Sistem

Dalam melakukan perancangan sistem terdapat tahapan-tahapan yang dilakukan yaitu:

1. Perancangan arsitektur sistem

   Tahapan perancangan arsitektur sistem meliputi gambaran umum sistem dan bagan alir sistem.

2. Perancangan diagram arus data

   Tahapan perancangan diagram arus data terdiri dari perancangan diagram konteks sistem, diagram *overview* sistem dan diagram rinci sistem.

3. Perancangan basis data

   Merancang entitas pembentuk sistem, atribut entitas, dan menggambarkan hubungan antar entitas pembentuk sistem dalam bentuk diagram hubungan antar entitas.

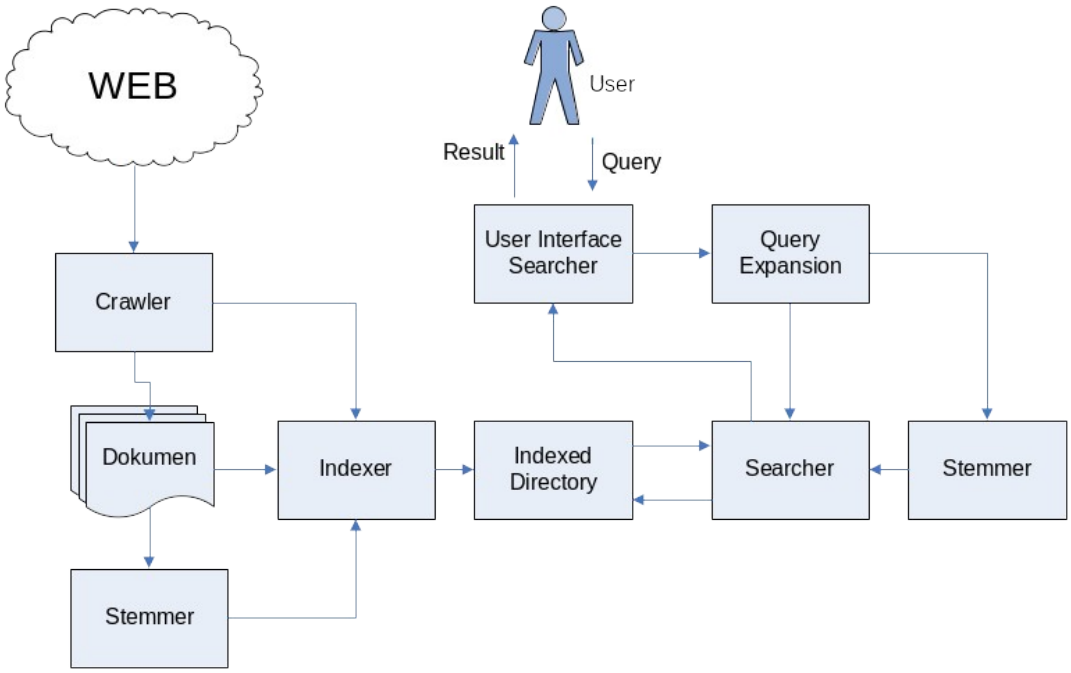
4. Perancangan antarmuka sistem

   Tahapan perancangan antarmuka sistem yaitu perancangan struktur antarmuka sistem serta perancangan *layout* dan komponen antarmuka sistem.

3.2.1 Arsitektur Sistem

3.2.1.1 Gambaran Umum Sistem

Gambaran Umum Sistem yang akan dibuat digambarkan melalui desain arsitektur sistem dapat dilihat pada Gambar 3.3.



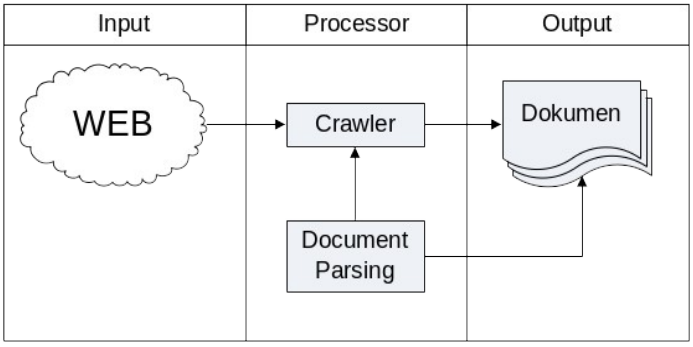
Gambar 3.3 Desain Arsitektur Sistem

Gambar 3.3 merupakan gambaran umum dari arsitektur sistem. Secara garis besar, sistem terdiri dari lima komponen utama yaitu *crawler*, *indexer, stemmer*, *query expansion* dan *searcher*.

Crawler berkelana di web dan berfungsi untuk mengumpulkan segala informasi tentang suatu halaman web. Informasi tentang halaman web tersebut didapat dari kata-kata yang terdapat di dalam halaman web tersebut. *Web crawler* digunakan untuk melakukan penjelajahan dan pengambilan halaman-halaman web yang ada di Internet (Sulastri, 2010).

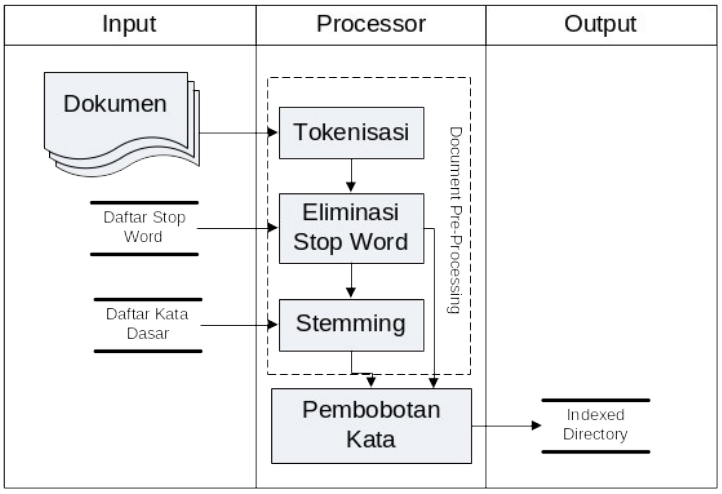
Metode yang dilakukan oleh *crawler* dapat dilihat pada Gambar 3.4. Proses yang dilakukan *crawler* akan menghasilkan dokumen berupa *metadata* yang dikumpulkan dari beberapa situs. Dokumen yang berupa metadata akan melalui proses parsing sehingga berita akan terekstraksi menjadi dokumen.

Document parsing berupa pengambilan kata-kata dalam dokumen hasil *crawling* yang digunakan untuk memudahkan dalam ekstraksi kata. Dokumen tersebut akan digunakan dalam *indexing* dokumen oleh *indexer*. *Crawler* juga mengumpulkan link dari metadata dan disimpan dalam dokumen untuk dijelajahi kemudian.



Gambar 3.4 Metode *Crawling* Dokumen

Dalam *indexing* yang dilakukan *indexer*, terdiri atas dua (2) bagian yaitu *document pre-processing* dan pembobotan kata pada masing-masing dokumen. Metode yang digunakan dalam *indexing* seperti yang terlihat pada Gambar 3.5.

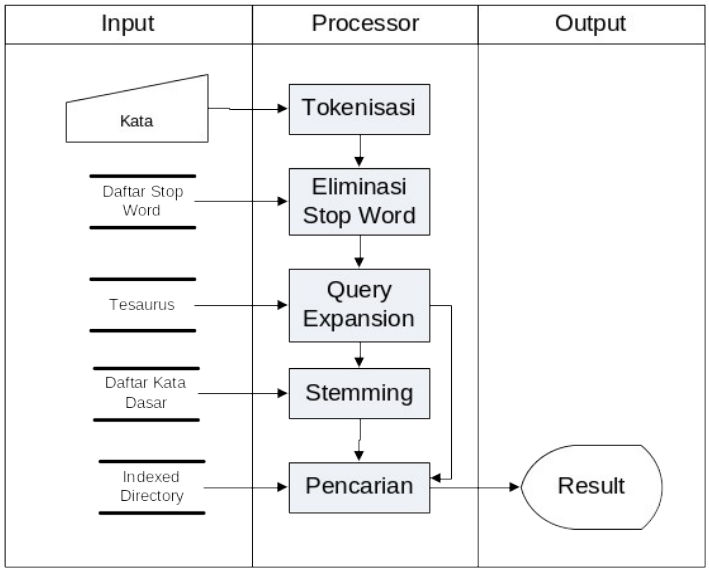


Gambar 3.5 Metode *Indexing* Dokumen

Document pre-processing melakukan ekstraksi kata dari dokumen (tokensisasi dokumen), eliminasi *stop words* dan *stemming*. Proses eliminasi *stop words* digunakan untuk menghilangkan kata-kata buang yang memiliki frekuensi kemunculan yang tinggi seperti: tetapi, yaitu, sedangkan, dan sebagainya. Proses *stemming* digunakan untuk mencari bentuk kata dasar dari kata dalam dokumen untuk selanjutnya dihitung bobotnya. Dalam proses *stemming*, digunakan algoritma porter untuk teks berbahasa Indonesia yang dikembangkan oleh Tala.

Dalam pengindeksan dokumen digunakan metode pembobotan kata berdasarkan *term frequency* (TF), *inverse document frequency* (IDF) dan kombinasi keduanya.

Proses pencarian merupakan proses menemukan kembali informasi (dokumen) yang relevan terhadap *query* kata kunci yang diberikan oleh pengguna dan menyediakan daftar dokumen terurut yang relevan dengan *query* tersebut. Dalam proses pencarian yang dilakukan *searcher*, *user* akan memasukkan kata yang akan dicari. Kata tersebut akan melewati beberapa proses sehingga menjadi kata kunci pencarian. Proses tersebut yaitu proses ekstraksi kata (tokenisasi), eliminasi kata-kata yang tidak layak dijadikan kata kunci pencarian (*stop words*), perluasan query (*query expansion*) dan pengembalian kata menjadi bentuk dasar (*stemming*) untuk memperkuat pencarian. Daftar dokumen yang memiliki keterkaitan akan diurutkan berdasarkan frekuensi kemunculan kata kunci. Daftar terurut inilah yang akan dikembalikan kepada pengguna. Metode yang digunakan dalam *searching* seperti yang terlihat pada Gambar 3.6.



Gambar 3.6 Metode *Searching* Dokumen

3.2.1.2 Bagan Alir Sistem

Bagan alir sistem merupakan bagan yang menunjukkan alur kerja atau apa yang sedang dikerjakan di dalam sistem secara keseluruhan dan menjelaskan urutan dari prosedur-prosedur yang ada di dalam sistem.

Pada Gambar 3.7 menjelaskan mengenai proses *crawling* dokumen yang terjadi pada administrator. Administrator harus melakukan proses *login* terlebih

dahulu sebelum dapat melakukan proses-proses lainnya. Setelah itu administrator dapat memulai *crawling*.

Proses *crawling* mengimplementasikan algoritma *depth first spidering* untuk pemilihan link selanjutnya dan *focused spidering* untuk meningkatkan kinerja *crawler*. *Depth first spidering* diimplementasikan dengan mengikuti *link* mulai dari *link* pertama pada halaman awal level 0 dilanjutkan pada *link* pertama pada level 1 dan begitu seterusnya sampai tidak ada *link* lagi *atau* link sudah mencapai batas maksimal *link* yang ditemukan, kemudian mengikuti *link* kedua pada halaman awal level 0 hingga *link* terakhir. *Focused spidering* diimplementasikan dalam *crawling* dengan membatasi domain link yang akan dijelajahi hanya jika sesuai dengan domain utama yang dimasukkan sehingga domain yang dijelajahi tepat pada situs penyedia berita yang diharapkan. Adapun langkah-langkah algoritma yang digunakan dalam proses *crawling* adalah sebagai berikut:

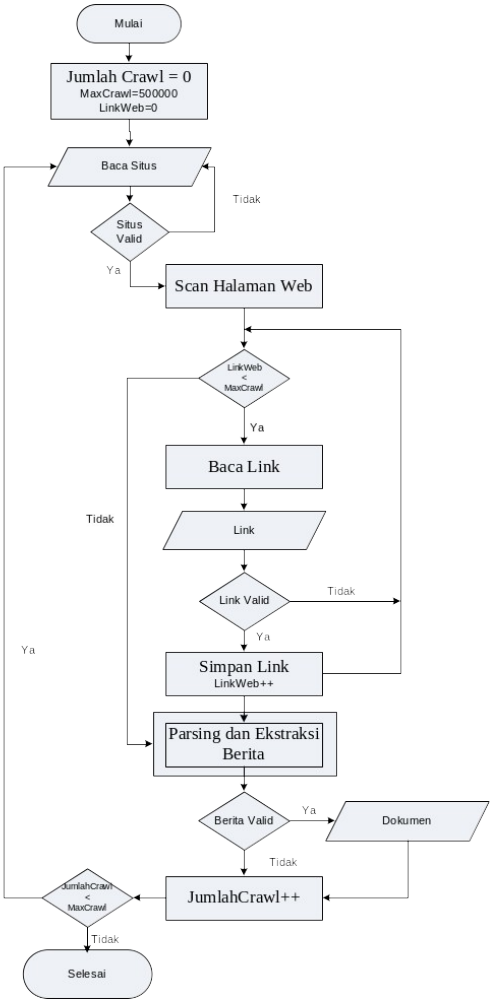
1. Set variabel jumlah halaman yang telah melalui *crawling* sama dengan 0, variabel jumlah maksimum sama dengan 10000, dan variabel jumlah link yang telah disimpan sama dengan 0.
2. Baca Situs.
3. Validasi apakah situs valid. Jika situs valid lanjutkan ke langkah 4, jika tidak valid maka ulangi ke langkah 2.
4. Scan halaman web berdasarkan alamat situs.
5. Cek jumlah link yang telah disimpan, jika kurang dari jumlah maksimal lanjutkan ke langkah 6, jika tidak lanjutkan ke langkah 8.
6. Baca link yang ada di halaman web
7. Simpan link ke database. Jumlah link yang telah disimpan ditambah 1.
8. Parsing dan ekstraksi berita.
9. Validasi berita, jika valid lanjutkan ke langkah 10, jika tidak lanjutkan ke langkah 11.
10. Simpan berita dalam dokumen.
11. Jumlah halaman yang telah melalui *crawling* ditambah 1.
12. Cek jumlah halaman yang telah melalui *crawling*. Jika kurang dari jumlah maksimal ulangi ke langkah 1, jika tidak lanjutkan ke langkah 11.

13. Proses *crawling* selesai.

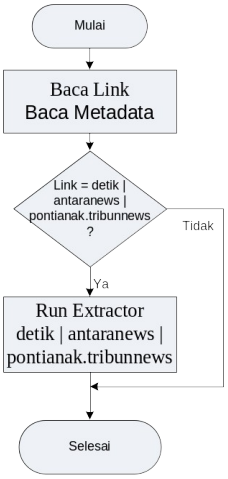
Crawling dimulai dengan membaca alamat situs, apabila alamat situs tersebut valid maka sistem akan membaca halaman web tersebut. Jika situs tersebut tidak valid maka sistem akan menolak untuk melakukan proses selanjutnya dan meminta dimasukkan kembali situs yang valid. Setelah membaca halaman web, sistem akan mengecek apakah alamat situs yang dimasukkan belum melebihi batas maksimal. Jika belum melebihi batas maksimal, sistem akan mengekstrak semua link yang terdapat dapat halaman web dan menyimpan link yang valid. Link yang valid yaitu link yang memiliki format alamat situs, bukan merupakan file, bukan termasuk ke dalam alamat yang diblok dan link yang belum ada didalam tabel dokumen. Kemudian sistem akan melakukan parsing dan ekstraksi berita. Proses ekstarksi berita dalam crawling dokumen digambarkan pada gambar 3.8. Jika berita ditemukan dan valid, berita akan disimpan dalam *database* dokumen. Proses *crawling* dari tahap baca situs akan diulangi jika masih ada alamat situs yang belum dijelajahi dan jumlah situs yang dijelajahi belum mencapai batas maksimal. Selanjutnya, dokumen yang dihasilkan *crawler* akan diproses dalam *indexer* dokumen.

Selanjutnya dokumen berita akan melalui proses *document pre-processing* yang meliputi ekstraksi kata dari dokumen (tokensisasi dokumen), eliminasi *stop words* dan *stemming*. Setelah melewati proses *document pre-processing*, dokumen akan diindeks dengan pembobotan kata. Proses indexing dokumen dapat dilihat pada gambar 3.9.

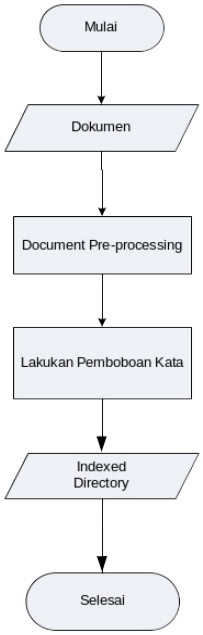
Pada Gambar 3.10 menjelaskan mengenai proses yang terjadi pada *user*. *User* hanya perlu untuk menginputkan kata kunci pencarian. Setelah itu sistem akan memproses dan menampilkan daftar hasil pencarian. Hasil pencarian dokumen yang ditampilkan kepada *user* diurutkan berdasarkan bobot kemunculan kata kunci dalam suatu dokumen. Pada Gambar 3.11 menjelaskan secara detail proses pengolahan kata kunci yang terdapat pada proses *searching* dokumen.



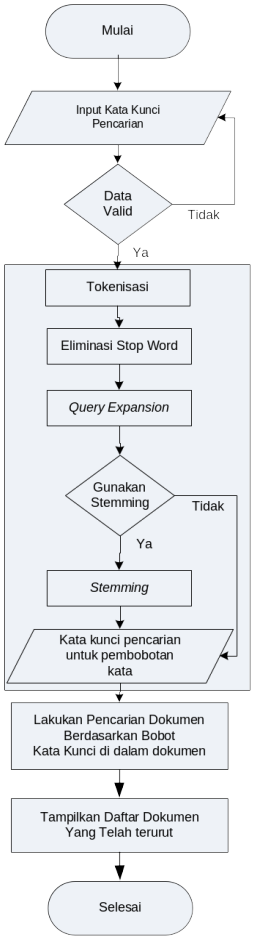
Gambar 3.7 Bagan Alir untuk *Crawling* Dokumen



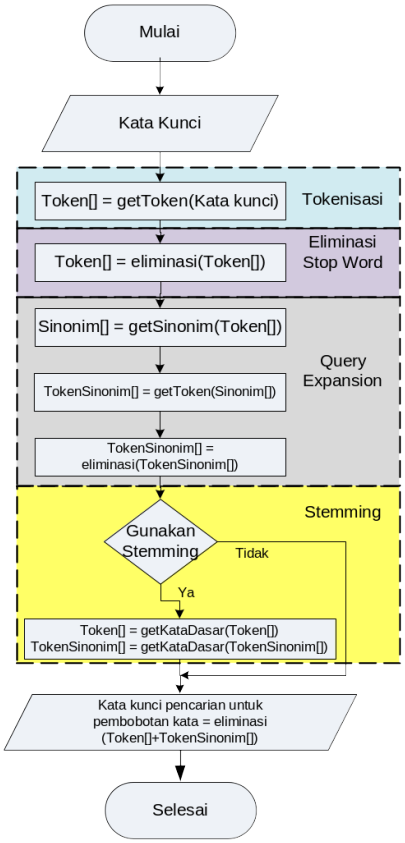
Gambar 3.8 Bagan Alir untuk *News Extractor* dalam *Crawling* Dokumen ~~Dokumen~~



Gambar 3.9 Bagan Alir untuk ~~Dokumen~~ *Indexing* Dokumen



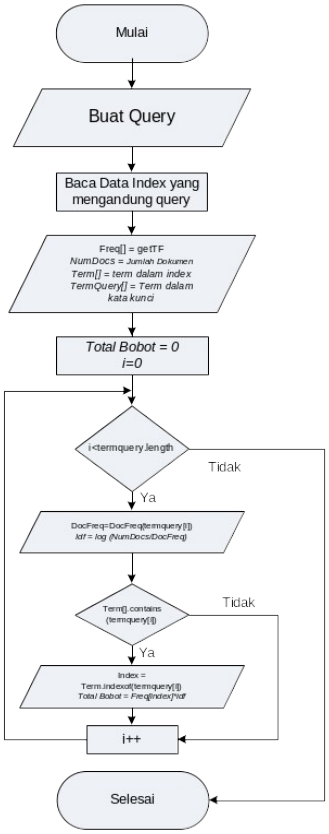
Gambar 3.10 Bagan Alir untuk *Searching* Dokumen



Gambar 3.11 Bagan Alir Pengolahan Kata Kunci untuk *Searching* Dokumen

Pada Gambar 3.12 menjelaskan mengenai proses yang terjadi ketika *searching* dokumen. Kata kunci yang dihasilkan pada proses *searching* dokumen akan diubah menjadi *query*. *Query* akan digunakan untuk membaca data index dan memanggil beberapa variabel dalam data index. Untuk semua term yang terdapat dalam *query*, dihitung nilai tf dan nilai idf untuk mengkalkulasi perkalian nilai tf dan idf sehingga jika semua nilai dari perkalian tf dan idf dijumlahkan, maka dihasilkan total bobot untuk sebuah dokumen.

Dalam mengoptimalkan proses pencarian ada beberapa hal yang dilakukan terhadap kata kunci. Kata kunci tersebut akan melewati beberapa proses sehingga menjadi kata kunci pencarian untuk pembobotan kata. Proses tersebut yaitu proses ekstraksi kata (tokenisasi), eliminasi kata-kata yang tidak layak dijadikan kata kunci pencarian (*stop words*), perluasan *query* (*query expansion*), dan pengembalian kata menjadi bentuk dasar (*stemming*) untuk memperkuat pencarian. Tahap selanjutnya sistem akan melakukan pencarian dokumen berdasarkan bobot kata kunci di dalam dokumen. Dokumen yang isinya memiliki bobot kemunculan kata kunci paling banyak akan diurutkan paling atas.



Gambar 3.12 Bagan Alir untuk Pencarian Dokumen berdasarkan Bobot

3.2.2 Perancangan Diagram Arus Data

3.2.2.1 Diagram Konteks Sistem

Diagram konteks adalah diagram yang memberikan gambaran umum terhadap kegiatan yang berlangsung dalam sistem. Gambar 3.13 berikut ini menunjukkan diagram konteks dari sistem.

Berdasarkan gambar diagram tersebut, entitas luar yang terlibat langsung dalam sistem adalah admin dan *user*. Berikut ini merupakan pembagian *privileges* untuk masing-masing entitas:

1. Admin

   Admin merupakan entitas yang mempunyai *previleges* tertinggi dalam sistem. Admin mempunyai akses penuh terhadap semua data admin. Admin juga mempunyai kewenangan untuk memanajemen situs, menambahkan kata-kata penting untuk proses *stemming*, menambahkan *stop word* dan menambahkan kata-kata dalam tesaurus. Admin dapat menambah, mengubah, dan menghapus data admin, kata dasar, *stop word* dan kata dalam tesaurus.

2. *User*

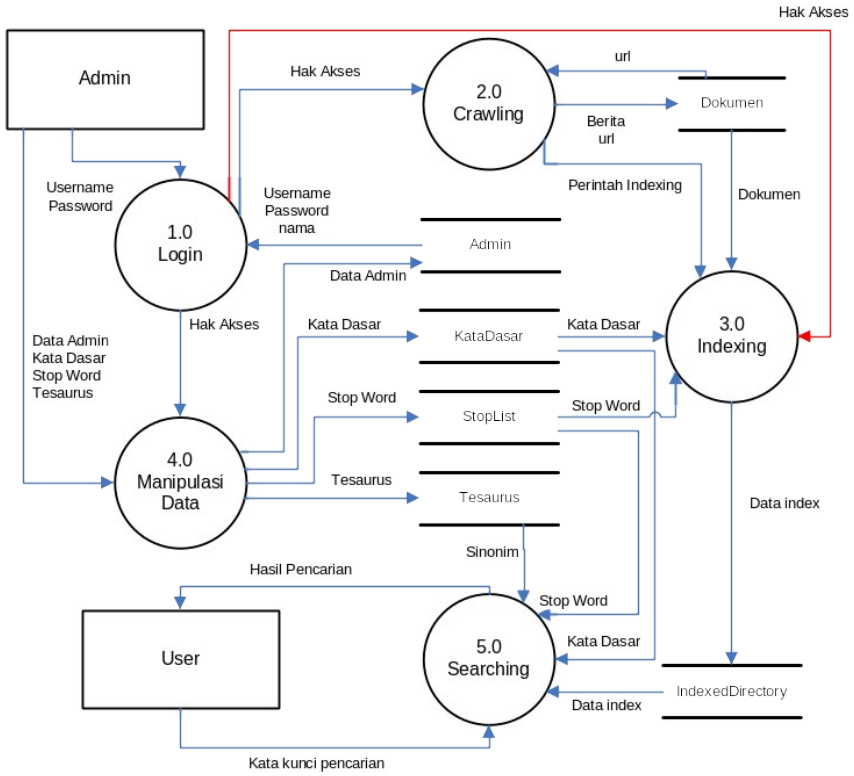
   Setiap orang yang dapat mengakses sistem ini secara *online* dapat dikatakan sebagai *user*. *User* diberi kebebasan untuk mencari dokumen yang mereka butuhkan tanpa adanya batasan oleh jumlah dan waktu. Tentunya keinginan setiap pengguna adalah mencari dokumen yang tersimpan dalam sistem ini dan dapat menemukannya secara mudah dan relevan. Pengguna hanya perlu untuk memasukan kata kunci pencarian yang berkaitan dengan dokumen yang dicari.

Gambar 3.13 Diagram Konteks Sistem

3.2.2.2 Diagram *Overview*

Diagram *overview* adalah diagram yang menjelaskan urutan-urutan proses dari diagram konteks. Pada diagram ini proses/tahapan dibagi menjadi empat proses yaitu:

1. Proses 1.0 *Login* adalah proses autentifikasi admin untuk memberikan akses terhadap proses *crawling, indexing* dan manipulasi data.
2. Proses 2.0 *Crawling* adalah proses pengumpulan dokumen dengan menjelajahi situs yang dikendalikan oleh admin.
3. Proses 3.0 *Indexing* adalah proses pengindeksan dokumen yang dikendalikan oleh admin.
4. Proses 4.0 Manipulasi Data merupakan proses mengelola data *stop word* dan kata dasar untuk digunakan oleh proses *indexing* dan *searching*.
5. Proses 5.0 *Searching* adalah proses pencarian dokumen yang dilakukan oleh *user* sistem untuk menemukan dokumen yang relevan dengan yang diinginkannya.



Gambar 3.14 Diagram *Overview* Sistem

3.2.2.3 Diagram Rinci Sistem

Diagram rinci sistem berisi penjelasan secara lebih terperinci dari diagram nol/*overview* sistem.

1. Proses 1.0 Login

   Proses ini terbagi lagi menjadi 3 proses sebagai berikut.

   a). Proses 1.1 *Input username* dan *password*

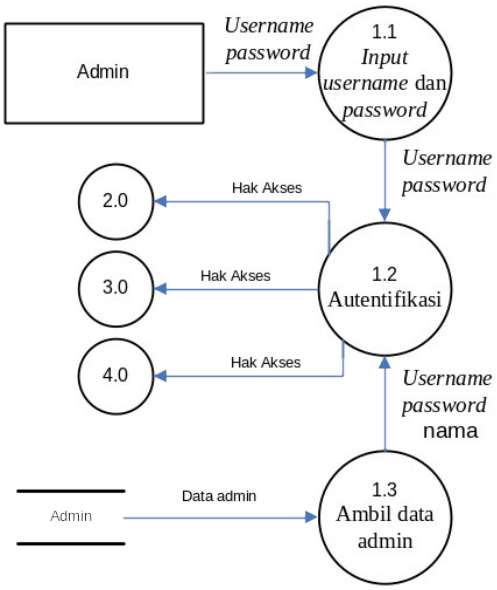
   Proses ini meminta inputan dari admin dan menyampaikan username dan password yang dimasukkan admin ke proses selanjutnya.

   b). Proses 1.2 Autentifikasi

   Proses ini mencocokkan username dan password dari proses sebelumnya dengan data yang diambil oleh proses ambil data admin dari *database*. Kemudian proses ini akan mengeluarkan autentifikasi benar atau tidaknya admin yang sedang login serta memberikan hak akses terhadap proses-proses lain.

   c). Proses 1.3 Ambil data admin

   Proses ini mengambil data admin dari *database* berdasarkan username yang dimasukkan.



Gambar 3.15 Diagram Rinci Proses 1.0 Level 2

2. Proses 2.0 *Crawling*

Proses ini terbagi lagi menjadi 8 proses sebagai berikut.

a). Proses 2.1 Baca daftar situs

Proses ini membaca url yang belum dirayapi dalam *database* kemudian mengirimkannya kepada proses selanjutnya.

b). Proses 2.2 Mulai *crawling*

Proses ini memulai *crawling* hingga semua seluruh *metadata* telah diunduh. Proses ini juga memberikan perintah pada proses 3.0 untuk memulai *indexing* dari data hasil *crawling* yang telah disimpan dalam *database*.

c). Proses 2.3 Ekstraksi Link

Proses ini mengekstraksi link yang terdapat dalam *metadata* hasil *crawling* dari proses 2.2 ke dalam *database*.

d). Proses 2.4 Cek Link

Proses ini memvalidasi link hasil proses 2.3 valid atau tidak. Link yang valid yaitu link yang memiliki format alamat situs, bukan merupakan file, bukan termasuk alamat yang diblok dan link belum ada didalam *database*.

e). Proses 2.5 Simpan Link

Proses ini menyimpan link valid hasil proses 2.4 ke dalam *database*.

f). Proses 2.6 Parsing dan Ekstraksi Berita

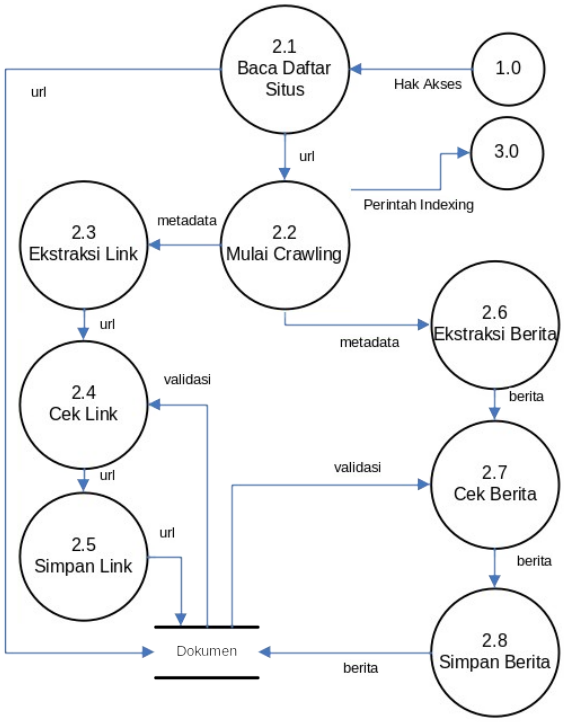
Proses ini melakukan *parsing metadata* dan mengekstraksi berita yang dari dokumen hasil *crawling* dari proses 2.2 ke dalam *database*.

g). Proses 2.7 Cek Berita

Proses ini memvalidasi berita hasil proses 2.6 valid atau tidak. Berita yang valid yaitu berita yang mempunyai komponen yang lengkap dan tidak sama dengan berita lain yang telah disimpan dalam *database*.

h). Proses 2.8 Simpan Berita

Proses ini menyimpan berita valid hasil proses 2.7 ke dalam *database*.



Gambar 3.16 Diagram Rinci Proses 2.0 Level 2

3. Proses 3.0 *Indexing*

Proses ini terbagi lagi menjadi 5 proses sebagai berikut.

a). Proses 3.1 Membaca dokumen

Proses ini membaca *database* berisi data hasil *crawling*. Proses ini akan dimulai jika mendapatkan masukan dari proses 2.0 berupa perintah *indexing*.

b). Proses 3.2 Tokenisasi

Pada proses ini, teks hasil proses 3.1 akan diekstraksi menjadi kata-kata yang berdiri sendiri.

c). Proses 3.3 Eliminasi *stop word*

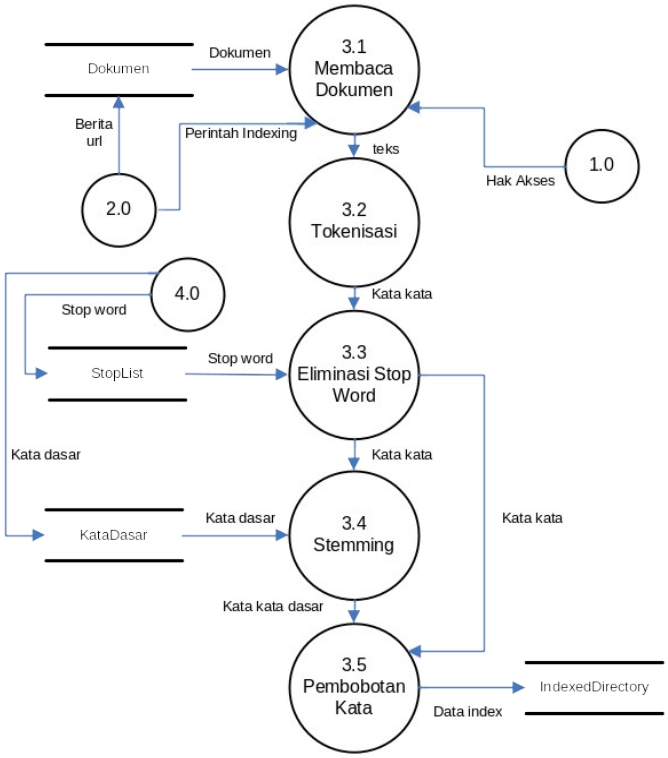
Pada proses ini, setiap kata hasil proses 3.2 akan dikenali. Jika kata tersebut termasuk dalam *stoplist*, maka kata tersebut akan dihapuskan.

d). Proses 3.4 *Stemming*

Pada proses ini, setiap kata hasil proses 3.3 akan diproses dalam *stemmer* untuk mengubah setiap kata menjadi bentuk dasar dari kata tersebut.

e). Proses 3.5 Pembobotan kata

Pada proses ini, setiap kata hasil proses 3.4 akan diberi bobot berdasarkan frekuensi kata dan frekuensi dokumen.



Gambar 3.17 Diagram Rinci Proses 3.0 Level 2

4. Proses 4.0 Manipulasi data

Proses ini terbagi lagi menjadi 3 proses sebagai berikut.

a). Proses 4.1 *Input* data

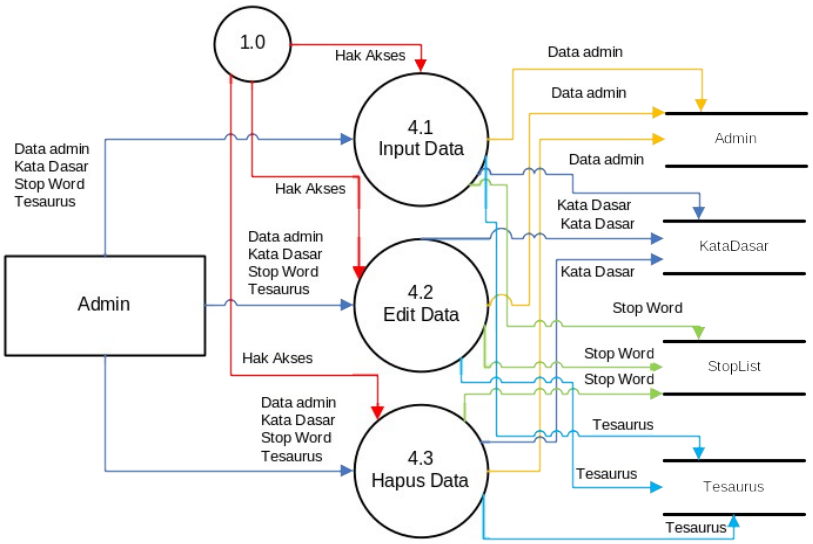
Proses ini menyimpan data dalam *database* yaitu data kata dasar, *stoplist*, tesaurus dan admin.

b). Proses 4.2 *Edit* data

Proses ini mengubah data dalam *database* yaitu data kata dasar, *stoplist*, tesaurus dan admin.

c). Proses 4.3 Hapus data

Proses ini menghapus data dalam *database* yaitu data kata dasar, *stoplist*, tesaurus dan admin.



Gambar 3.18 Diagram Rinci Proses 4.0 Level 2

5. Proses 5.0 *Searching*

Proses ini terbagi lagi menjadi 6 proses sebagai berikut.

a). Proses 5.1 Membaca kata kunci

Proses ini menerima input kata kunci dari *user* yang akan digunakan pada seluruh proses pencarian.

b). Proses 5.2 Tokenisasi

Pada proses ini, kata kunci yang dibaca pada proses 5.1 akan diekstraksi menjadi kata-kata yang berdiri sendiri.

c). Proses 5.3 Eliminasi *stop word*

Pada proses ini, setiap kata hasil proses 5.2 akan dikenali. Jika kata tersebut termasuk dalam *stoplist*, maka kata tersebut akan dihapuskan.

d). Proses 5.4 *Query expansion*

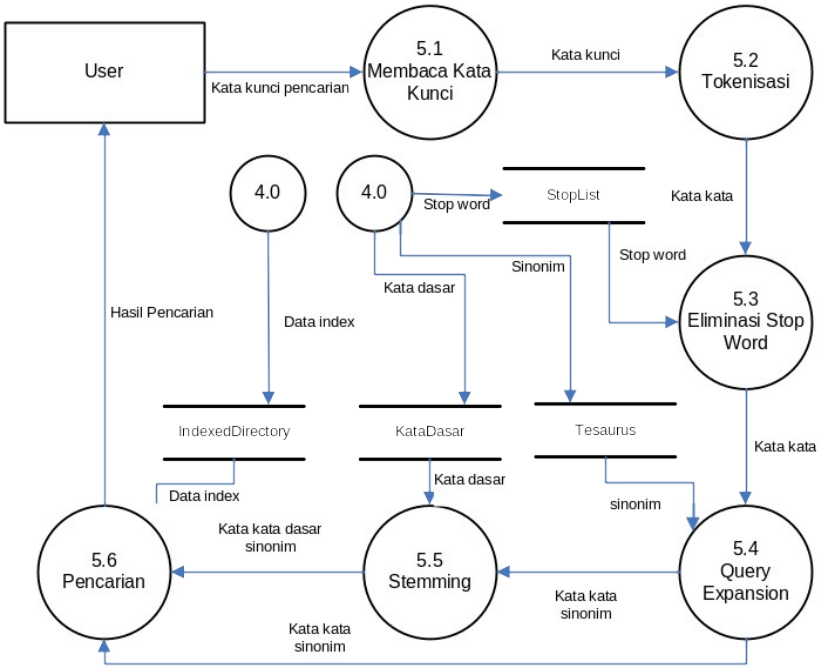
Pada proses ini, setiap kata hasil proses 5.3 akan dikenali dan dicari synset kata tersebut dalam tesaurus. Kata hasil proses 5.3 ditambah synset kata tersebut menjadi *output* pada proses ini dan akan digunakan dalam pencarian.

e). Proses 5.5 *Stemming*

Pada proses ini, setiap kata hasil proses 5.4 akan diproses dalam *stemmer* untuk mengubah setiap kata menjadi bentuk dasar dari kata tersebut.

f). Proses 5.6 Pencarian

Pada proses ini, setiap kata hasil proses 5.5 akan digunakan sebagai *query* untuk pencarian yang dilakukan terhadap *indexed directory*.



Gambar 3.19 Diagram Rinci Proses 5.0 Level 2

3.2.3 Perancangan Basis Data

3.2.3.1 Spesifikasi Tabel Basis Data Tabular

Sistem ini menggunakan tabel-tabel yang dibuat menggunakan *database* MySQL. Sistem ini menggunakan 6 tabel, yaitu tb_katadasar, tb_stoplist, dokumen, tesaurus, admin dan *lastindex*. Sistem ini juga membuat *indexed directory* menggunakan Apache Lucene yang namanya sesuai dengan tanggal dan waktu indexing dilakukan serta metode pengindeksan yang digunakan.

Tabel kata dasar merupakan tabel referensi yang digunakan untuk menyimpan kata dasar. Kata dasar yang tersimpan akan digunakan dalam autentifikasi kata dasar pada proses *stemming*. Dalam sistem yang dirancang, digunakan kata dasar yang berasal dari kamus bahasa Indonesia dari pusat bahasa departemen pendidikan nasional yang diterbitkan tahun 2008 versi elektronik yang kemudian telah dikonversi menjadi kata dasar dalam file .babylon oleh

Steven Haryanto (http://steven.haryanto.org). File .babylon kemudian dapat dibaca sebagai file .txt untuk diinputkan ke dalam database. Spesifikasi tabel kata dasar dapat dilihat pada Tabel 3.1.

Tabel 3.1 Spesifikasi Tabel Kata Dasar

Nama Field	Tipe	Fungsi	Key
Id_ktdasar	int(10)	Menyimpan nomor id kata dasar	*
Katadasar	varchar(20)	Menyimpan kata dasar	
Tipe_katadasar	varchar(20)	Menyimpan tipe kata dasar	

Tabel *stoplist* merupakan tabel referensi yang digunakan untuk menyimpan kata yang termasuk kata buang (*stop word*). Kata Buang akan digunakan dalam proses eliminasi *stop word*. Spesifikasi tabel *stoplist* dapat dilihat pada Tabel 3.2.

Tabel 3.2 Spesifikasi Tabel *Stoplist*

Nama Field	Tipe	Fungsi	Key
Id_Stoplist	int(10)	Menyimpan nomor id kata buang	*
Stoplist	varchar(50)	Menyimpan kata buang	

Tabel Dokumen merupakan tabel referensi yang digunakan untuk menyimpan data dokumen hasil *crawling* situs. Data dalam tabel dokumen akan digunakan dalam proses *indexing*. Spesifikasi tabel dokumen dapat dilihat pada Tabel 3.3.

Tabel 3.3 Spesifikasi Tabel Dokumen

Nama Field	Tipe	Fungsi	Key
Id	int(12)	Menyimpan nomor id URL	*
Url	Text	Menyimpan URL	
Status	Int(1)	Menyimpan status *indexing*	
Judul	Tinytext	Menyimpan judul dokumen	
Date	Tinytext	Menyimpan tanggal dokumen dibuat	
Reporter	Tinytext	Menyimpan reporter dokumen	
Isi	Mediumtext	Menyimpan isi dokumen	

Tabel tesaurus merupakan tabel referensi yang digunakan untuk menyimpan sinonim kata dan antonim kata. Sinonim kata yang terdapat dalam tabel tesaurus akan digunakan dalam proses *query expansion* terhadap kata kunci yang dimasukkan pengguna. Dalam sistem yang dirancang, digunakan tesaurus yang berasal dari tesaurus bahasa Indonesia pusat bahasa dari pusat bahasa departemen pendidikan nasional yang diterbitkan tahun 2008 versi elektronik. Tesaurus dari pusat bahasa berupa file .pdf yang didistribusikan per bab berdasarkan abjad sehingga berjumlah 26 file yang kemudian telah dikonversi menjadi file .babylon oleh Steven Haryanto. File .babylon kemudian dapat dibaca sebagai file .txt untuk diinputkan ke dalam database. Spesifikasi tabel tesaurus dapat dilihat pada Tabel 3.4.

Tabel 3.4 Spesifikasi Tabel Tesaurus

Nama Field	Tipe	Fungsi	Key
Kata	Varchar(45)	Menyimpan kata	*
Bentuk	Varchar(45)	Menyimpan bentuk kata	
Sinonim	text	Menyimpan sinonim kata	
Antonim	Varchar(255)	Menyimpan antonim kata	

Tabel admin merupakan tabel referensi yang digunakan untuk menyimpan data-data mengenai admin. Tabel admin akan digunakan dalam autentifikasi admin pada proses login. Tabel admin juga menyimpan waktu login terakhir setiap admin. Spesifikasi tabel admin dapat dilihat pada Tabel 3.5.

Tabel 3.5 Spesifikasi Tabel Admin

Nama Field	Tipe	Fungsi	Key
Id	int(10)	Menyimpan nomor id admin	*
username	Varchar(45)	Menyimpan username	
Password	Varchar(45)	Menyimpan password	
Nama	Varchar(100)	Menyimpan nama	
Status	int(1)	Menyimpan hak akses	
Login_terakhir	datetime	Menyimpan data waktu terakhir login	

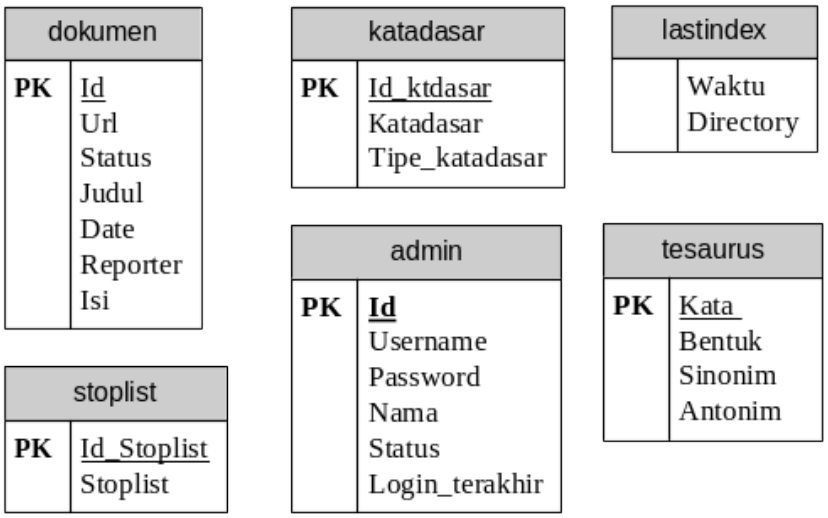
Tabel *lastindex* merupakan tabel referensi yang digunakan untuk menyimpan data proses pengindeksan berupa waktu dan directory penyimpanan indeks. Directory yang terdapat dalam tabel *lastindex* akan digunakan dalam proses *Searching* yang dilakukan pengguna. Spesifikasi tabel *lastindex* dapat dilihat pada Tabel 3.6.

Tabel 3.6 Spesifikasi Tabel *Lastindex*

Nama Field	Tipe	Fungsi	Key
Waktu	Datetime	Menyimpan tanggal dan waktu pengindeksan	
Directory	tinytext	Menyimpan directory tujuan pengindeksan	

3.2.3.2 Diagram Hubungan Antar Tabel Data Tabular

Model relasi antar tabel (model fisik) dari model konseptualnya, yang terbentuk beserta *fieldnya* dan *primary key* dapat dilihat pada Gambar 3.20.



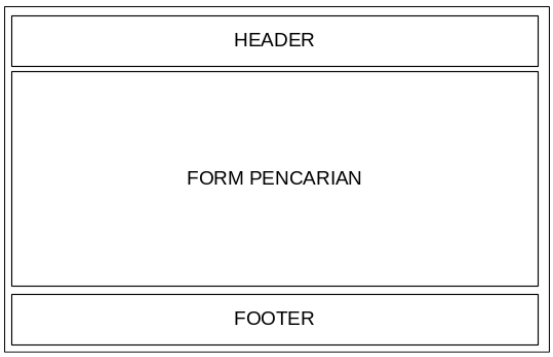
Gambar 3.20 Diagram Hubungan Antar Tabel

3.2.4 Perancangan Antarmuka Sistem

Antarmuka merupakan media interaksi dengan sistem. Pada bagian ini akan dijelaskan *layout* dan komponen antarmuka sistem. Secara umum sistem memiliki dua antarmuka utama yaitu antarmuka *user* dan antarmuka *admin*. Tampilan yang dimunculkan tergantung hak akses saat mengakses sistem.

3.2.4.1 *Form Index Page User*

Form index merupakan halaman utama dari sistem. Form ini terdiri dari beberapa bagian yaitu *header, form* pencarian dan *footer*. *Layout* dari *form index* dapat dilihat pada gambar 3.21.

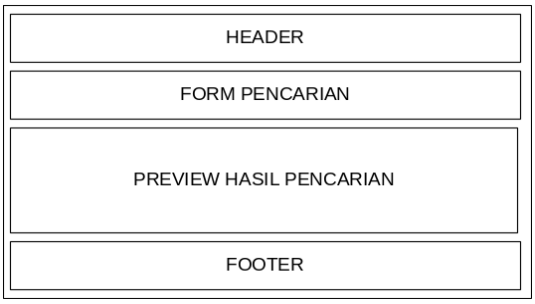


Gambar 3.21 *Layout* Antarmuka *Form Index Page User*

Menu header berisi logo dan Menu *form* pencarian terletak di bawah menu *header*. Menu footer menampilkan tahun pembuatan dan nama pembuat sistem.

3.2.4.2 *Form User Preview* Hasil Pencarian

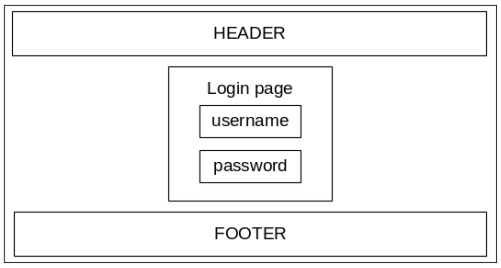
Pada gambar 3.22 menampilkan menu *preview* hasil pencarian yang digunakan untuk melihat informasi berita, penggalan isi berita serta link halaman situs berita.



Gambar 3.22 *Layout* Antarmuka *Form User Preview* Hasil Pencarian

3.2.4.3 *Form Login* Admin

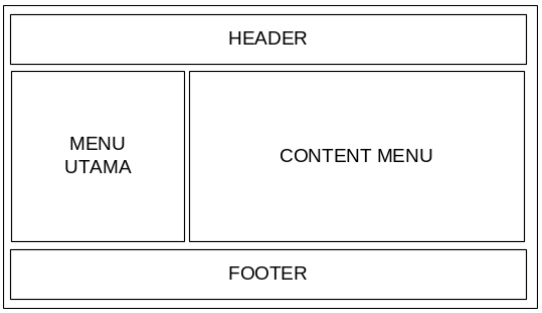
Gambar 3.23 menampilkan *layout form login* yang berfungsi untuk menjaga keamanan sistem dan hanya yang berhak (dalam hal ini admin) yang diizinkan masuk ke dalam sistem. *Form* ini berisi *username* serta *password*, apabila pengguna mengisikan data *username* dan *password* dengan benar maka petugas akan masuk ke halaman utama untuk melakukan proses-proses lainnya.



Gambar 3.23 *Layout* Antarmuka *Form Login* Admin

3.2.4.4 *Form Index Page* Admin

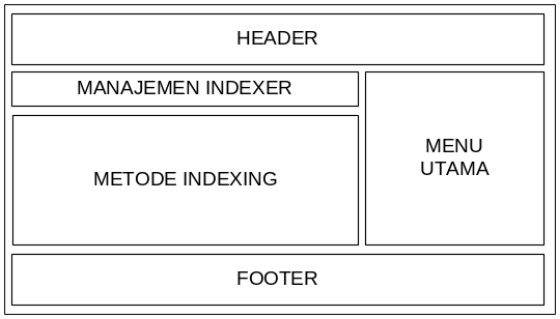
Gambar 3.24 menampilkan halaman utama untuk admin. Pada antarmuka tersebut terdapat dua bagian penting yaitu menu utama dan *content* dari menu utama. Menu utama berisi menu-menu untuk melakukan proses manajemen sistem. Menu-menu tersebut yaitu beranda, manajemen *indexer*, manajemen *searcher*, manajemen situs, manajemen kata dasar, manajemen *stop word* dan manajemen tesaurus.



Gambar 3.24 *Layout* Antarmuka *Form Index* Admin

3.2.4.5 *Form* Menu Admin Untuk Manajemen Pencarian

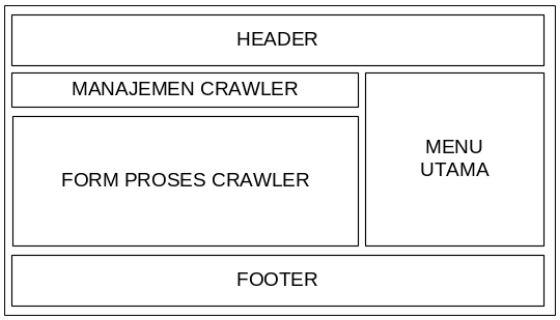
Gambar 3.25 menampilkan halaman dari menu manajemen pencarian untuk admin. Pada antarmuka tersebut ditampilkan pilihan metode *indexing* yang akan diterapkan.



Gambar 3.25 *Layout* Menu Admin untuk Manajemen Pencarian

3.2.4.6 *Form* Menu Admin Untuk Manajemen *Crawler*

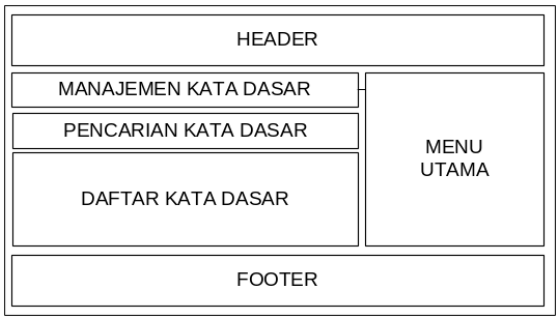
Gambar 3.26 menampilkan halaman dari menu manajemen *crawler* untuk admin. Pada antarmuka tersebut ditampilkan form untuk memulai *crawling*.



Gambar 3.26 *Layout* Menu Admin untuk Manajemen *Crawler*

3.2.4.7 *Form* Menu Admin Untuk Manajemen Kata Dasar

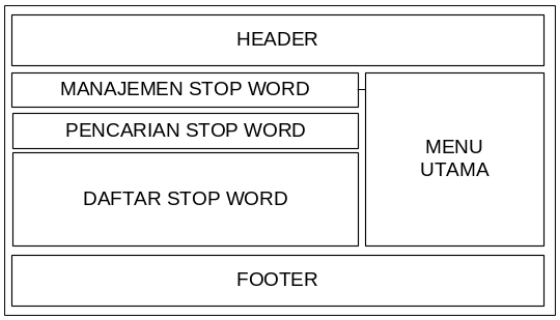
Gambar 3.27 menampilkan halaman dari menu manajemen kata dasar untuk admin. Pada antarmuka tersebut ditampilkan pilihan daftar kata dasar yang ada dalam sistem, menu untuk menambah, mengubah dan menghapus kata dasar serta menu pencarian kata dasar.



Gambar 3.27 *Layout* Menu Admin untuk Manajemen Kata Dasar

3.2.4.8 *Form* Menu Admin Untuk Manajemen *Stop Word*

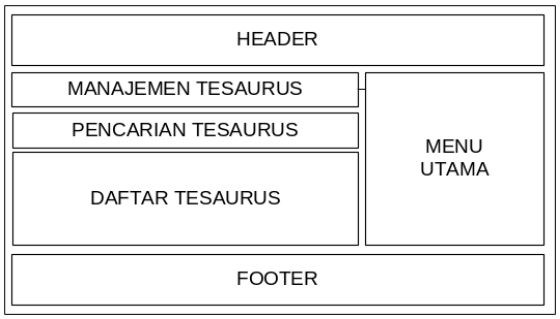
Gambar 3.28 menampilkan halaman dari menu manajemen *stop word* untuk admin. Pada antarmuka tersebut ditampilkan pilihan daftar *stop word* yang ada dalam sistem, menu untuk menambah, mengubah dan menghapus *stop word* serta menu pencarian *stop word*.



Gambar 3.28 *Layout* Menu Admin untuk Manajemen *Stop Word*

3.2.4.9 *Form* Menu Admin Untuk Manajemen Tesaurus

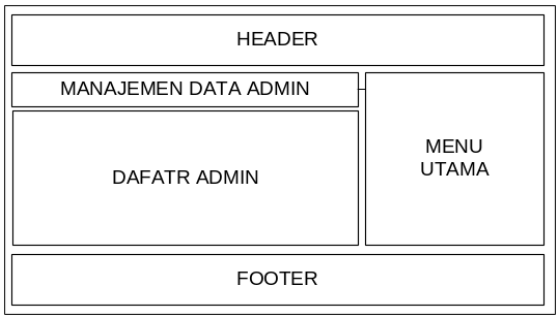
Gambar 3.29 menampilkan halaman dari menu manajemen tesaurus untuk admin. Pada antarmuka tersebut ditampilkan pilihan daftar tesaurus yang ada dalam sistem, menu untuk menambah, mengubah dan menghapus tesaurus serta menu pencarian tesaurus.



Gambar 3.29 *Layout* Menu Admin untuk Manajemen Tesaurus

3.2.4.10 *Form* Menu Admin Untuk Manajemen Data Admin

Gambar 3.30 menampilkan halaman dari menu manajemen data admin. Pada antarmuka tersebut ditampilkan pilihan daftar admin yang ada dalam sistem, menu untuk menambah, mengubah dan menghapus data admin.



Gambar 3.30 *Layout* Menu Admin untuk Manajemen Data Admin

BAB IV
HASIL ANALISIS DAN PERANCANGAN

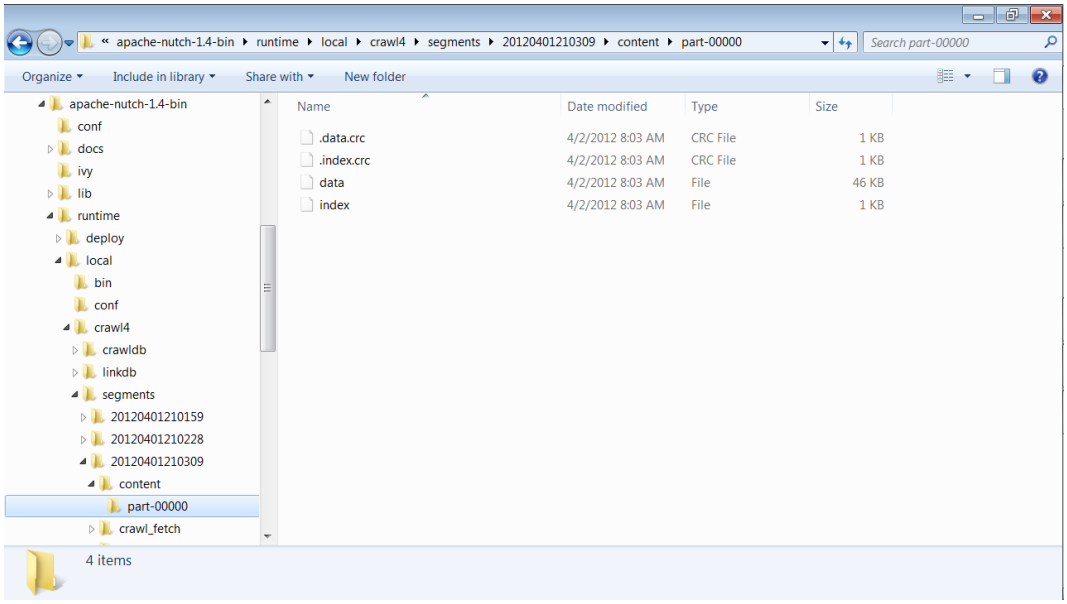
4.1 Hasil Analisis

Sistem temu balik informasi yang dirancang memiliki proses utama berupa penemuan kembali dokumen berita dari website penyedia berita *online*. Dokumen berita yang terdapat dalam website penyedia berita terus bertambah seiring waktu. Dokumen tersebut masih berupa dokumen yang tidak terstruktur sehingga sistem ini juga harus mengenali kandungan berita dalam dokumen tidak terstruktur tersebut. Berita yang dikenali dari judul, repoter, tanggal dan isi berita. Berita yang telah dikenali dapat diproses dan diindeks untuk mengenali varian morfologik dari istilah-istilah yang terdapat dalam berita.

Kata yang memiliki banyak varian morfologik diproses untuk mendapatkan kata dasar tanpa *stop word*. Proses yang digunakan untuk mengekstrak kata sehingga menjadi kata dasar yaitu proses *stemming*. Proses *stemming* yang diimplementasikan dalam sistem ini menggunakan algoritma porter untuk teks berbahasa Indonesia. Proses *stemming* dengan algoritma porter menemukan kata dasar dari sebuah kata berdasarkan aturan pemotongan dan daftar kata dasar dalam *database*.

Kemunculan istilah dalam dokumen memiliki bobot masing-masing sesuai jumlah kemunculannya pada seluruh dokumen. Istilah yang memiliki jumlah frekuensi kemunculan yang tinggi pada seluruh dokumen akan memiliki nilai *inverse term frequency* (idf) yang rendah. Bobot istilah tersebut menjadi rendah karena idf merupakan faktor pengali untuk bobot istilah. Sebaliknya, jumlah kemunculan istilah dalam sebuah dokumen (*term frequency*) akan meningkatkan bobot istilah karena merupakan faktor pengali. Semakin banyak istilah yang muncul dalam sebuah dokumen, bobot istilah tersebut akan semakin tinggi. Dengan demikian, pembobotan kata dapat membedakan pentingnya sebuah istilah berdasarkan bobotnya dan menemukan informasi yang paling relevan berdasarkan bobot tersebut. Dokumen yang telah memiliki bobot masing-masing diurutkan berdasarkan bobot tertinggi sehingga dokumen yang paling relevan terhadap *query* dapat ditampilkan.

Proses pengumpulan dokumen untuk pencarian dilakukan oleh *crawler* yang dibangun sendiri. Pertama, dilakukan pengujian pada penggunaan beberapa *crawler* diantaranya Nutch yang bekerja di lingkungan Linux. Penggunaan Nutch menghasilkan dokumen yang masih belum dapat diuraikan dan dimanfaatkan. File yang dihasilkan oleh Nutch berupa data berformat file yang bernama “data”, “index”, “.data.crc” dan “.index.crc” seperti pada gambar 4.1. Selanjutnya belum ditemukan cara menguraikan dan memanfaatkan hasil ini, sehingga digunakan alternatif untuk membuat sendiri *crawler* dengan pengenal berita khusus yang selanjutnya disebut News Extractor. News Extractor bertugas untuk menguraikan metadata hasil *crawling* sehingga dapat dikenali bagian-bagian dari berita secara lengkap yaitu judul, tanggal, reporter dan isi berita.



Gambar 4.1 File Hasil Program Nutch

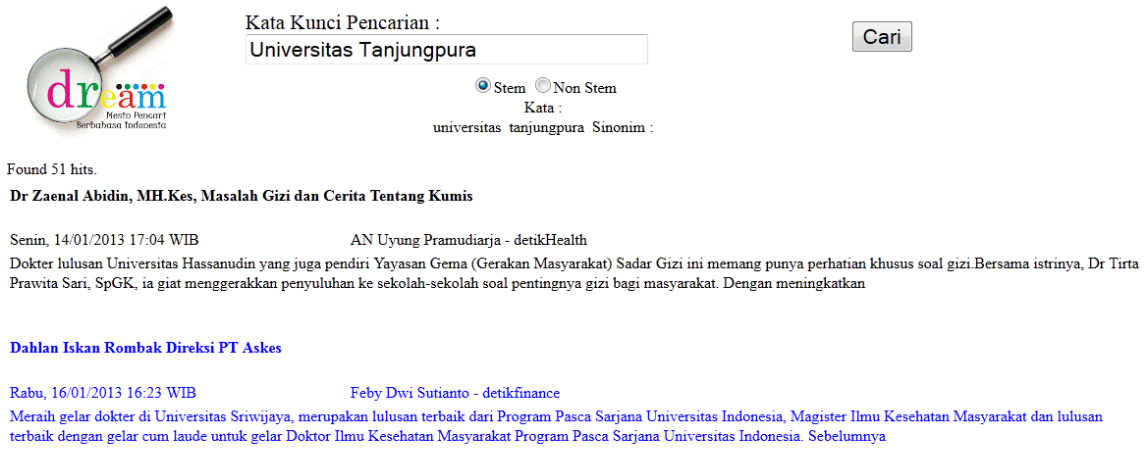
4.2 Hasil Perancangan

Sistem yang telah dirancang menggunakan JSP sebagai bahasa pemrograman, Apache Tomcat sebagai web server, MySql sebagai pengelola database serta JDK dan JRE sebagai *tools* pendukung. *User* dan Admin dapat mengakses sistem dengan web browser seperti Mozila Firefox dan Internet Explorer. Antarmuka halaman utama sistem ini dapat dilihat pada gambar 4.2.

Gambar 4.2 Antarmuka Halaman Utama untuk *User*

Antarmuka halaman utama untuk *user* berisi fitur pencarian yang dapat digunakan *user* untuk mencari berita. Pada bagian paling bawah terdapat 2 pilihan fitur pencarian antara lain stem dan non stem. Kata kunci yang ingin dicari dimasukkan ke form isian, kemudian memilih satu diantara 2 fitur pencarian dan akan dieksekusi ketika tombol cari diklik. Hasil eksekusi pencarian akan muncul halaman hasil pencarian. Antarmuka halaman hasil pencarian ini dapat dilihat pada gambar 4.3.

Antarmuka halaman hasil pencarian berisi *header* dan hasil pencarian. *Header* berisi kata kunci pencarian, 2 fitur pencarian, kata kunci yang dicari dan sinonim dari kata kunci yang dicari sebagai fitur *query expansion*. Kata kunci yang dicari di *header* akan sesuai dengan fitur pencarian yang dipilih sehingga jika fitur stem yang dipilih, maka kata kunci yang muncul akan berupa kata dasar.



Gambar 4.3 Antarmuka Halaman Hasil Pencarian

Hasil pencarian yang muncul berupa judul berita dan potongan isi berita. Kata dalam potongan isi berita yang sesuai dengan kata kunci akan disorot dengan warna hijau. Hasil pencarian yang ditampilkan terurut berdasarkan pembobotan tertinggi. Setiap halaman hasil pencarian akan menampilkan maksimal 20 data. Hasil pencarian disajikan sebagai link halaman sehingga ketika ingin membuka halaman sumber berita dapat mengklik langsung dibagian hasil pencarian.

4.2.1 Halaman *Login* Admin

Halaman *login* admin merupakan halaman yang mengautentifikasi admin. Autentifikasi akan digunakan untuk mengakses halaman admin yang digunakan untuk memanajemen pencarian. Untuk melakukan *login* ke dalam sistem, admin harus memasukkan *username* dan *password* yang sesuai. Antarmuka halaman *login* admin ini dapat dilihat pada gambar 4.4.

Gambar 4.4 Antarmuka Halaman *Login* Admin

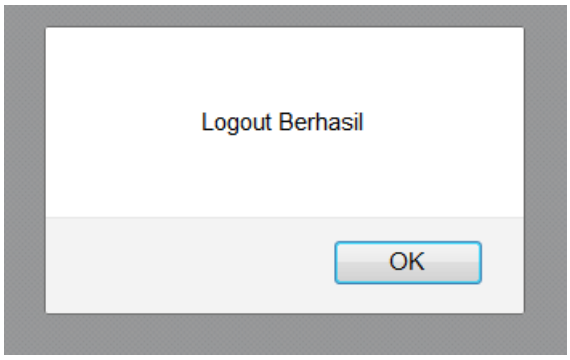
4.2.2 Halaman Admin

Halaman admin merupakan halaman yang digunakan untuk memanajemen pencarian baik memanajemen tabel pendukung dalam pencarian dan tabel admin maupun memanajemen proses *crawling* dan *indexing*. Halaman admin terdiri dari 4 komponen utama yaitu *header*, menu utama, *content menu* dan *footer*. *Header* berisi informasi admin yang sedang *login* dan tombol *logout* untuk keluar dari halaman admin. Menu utama berisi menu dalam memanajemen pencarian antara lain manajemen *crawler*, manajemen pencarian, manajemen admin dan manajemen kamus. Manajemen kamus terbagi menjadi tiga yaitu tesaurus, kata dasar dan *stop list*. *Content menu* berupa isi dari menu utama yang dipilih. *Footer*

berisi informasi website. Antarmuka halaman utama admin ini dapat dilihat pada gambar 4.5.

Gambar 4.5 Antarmuka Halaman Utama Admin

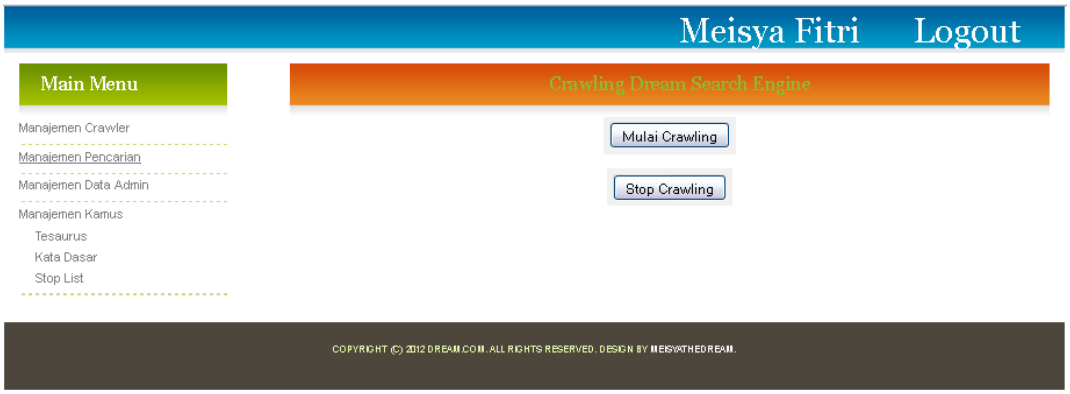
Admin yang telah masuk ke halaman admin dapat keluar dengan melakukan *logout*. *Logout* dimulai ketika tombol *logout* pada *header* diklik. Jika proses *logout* berhasil, akan muncul pesan seperti pada gambar 4.6, kemudian akan langsung diarahkan ke halaman *login*.



Gambar 4.6 Pesan Notifikasi *Logout*

4.2.2.1 Halaman Manajemen *Crawler*

Halaman manajemen *crawler* berisi *form crawling* berita. *Form crawling* berisi perintah untuk memulai dan menghentikan *crawling*. *Crawling* akan dilakukan terhadap situs yang sudah ditentukan sebelumnya. Data hasil *crawling* akan disimpan dalam *database* bernama dokumen. Antarmuka halaman manajemen *crawler* dapat dilihat pada gambar 4.7.

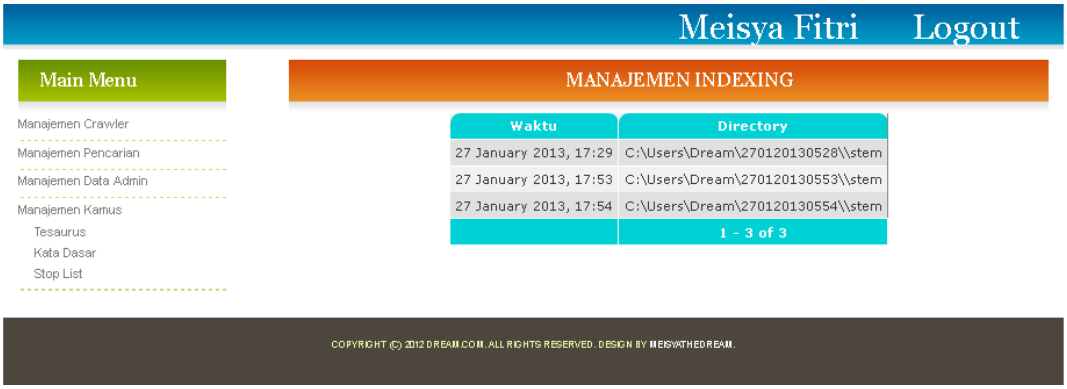


Gambar 4.7 Antarmuka Halaman Manajemen *Crawler*

4.2.2.2 Halaman Manajemen Pencarian

Halaman manajemen pencarian berisi informasi mengenai *indexing* dan *form indexing*. Informasi mengenai *indexing* berisi waktu *indexing* terakhir dan directori tujuan *indexing* yang tercatat dalam database serta menu lihat semua yang dapat digunakan untuk membuka halaman lihat data *indexing*. Halaman lihat data *indexing* berisi semua data *indexing* yang pernah dilakukan. *Form indexing* berisi pilihan tipe *indexing* yang akan digunakan yaitu stem atau non stem serta tombol untuk memulai *indexing*. *Indexing* akan dilakukan ke dalam folder yang dinamai dengan tanggal dan waktu *indexing* serta tipe yang digunakan. Antarmuka halaman manajemen pencarian dapat dilihat pada gambar 4.8 dan antarmuka halaman lihat data *indexing* dapat dilihat pada gambar 4.9.

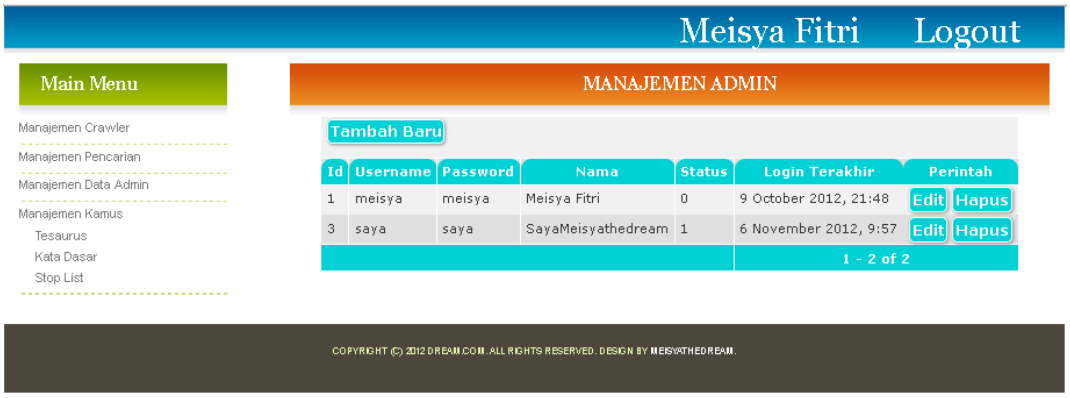
Gambar 4.8 Antarmuka Halaman Manajemen Pencarian



Gambar 4.9 Antarmuka Halaman Lihat Data *Indexing*

4.2.2.3 Halaman Manajemen Data Admin

Halaman manajemen data admin berisi data admin yang terdaftar. Data admin ditampilkan berupa tabel yang berisi informasi *username*, nama, status, *login* terakhir. Antarmuka halaman manajemen data admin dapat dilihat pada gambar 4.10. Terdapat tiga pilihan proses untuk manipulasi data admin yaitu tambah data, *edit* dan hapus yang juga tersedia pada halaman manajemen data admin.



Gambar 4.10 Antarmuka Halaman Manajemen Data Admin

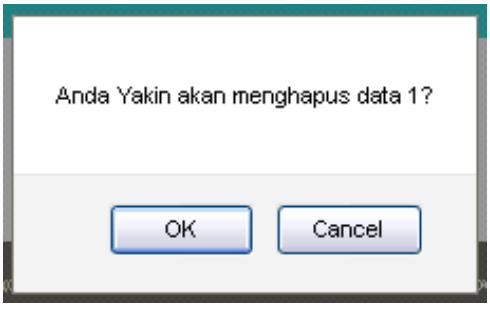
Admin dapat menambah admin baru untuk memanajemen pencarian. Untuk menggunakan fitur tambah baru, admin harus memasukkan *username*, *password*, nama, status dan *login* terakhir, seperti gambar 4.11. Admin dapat mengubah data admin yang dibatasi untuk data *username*, *password*, nama, status dan *login* terakhir. Pengubahan data admin dilakukan berdasarkan id admin yang tidak dapat ikut diubah.

Antarmuka halaman *edit* data admin dapat dilihat pada gambar 4.12. Admin juga dapat menghapus data admin dengan menekan tombol hapus pada halaman manajemen data admin. Ketika tombol hapus diklik, akan muncul

peringatan sebelum menghapus data. Jika benar akan menghapus data, maka admin dapat mengkonfirmasi dengan menekan tombol ok atau tombol *cancel* untuk membatalkan perintah hapus seperti pada gambar 4.13.

Gambar 4.11 Antarmuka Halaman Tambah Admin

Gambar 4.12 Antarmuka Halaman *Edit* Admin



Gambar 4.13 Pesan Peringatan Hapus Data Admin

4.2.2.4 Halaman Manajemen Kamus

Halaman manajemen kamus terdiri dari tiga bagian antara lain tesaurus, kata dasar dan *stop list*. Terdapat masing-masing tiga pilihan proses untuk manipulasi tesaurus, kata dasar dan *stop list* yaitu tambah data, *edit* dan hapus. Admin dapat menambah tesaurus untuk kata baru, kata dasar baru dan kata buang baru yang belum terdaftar pada aplikasi.

1. Halaman Tesaurus

   Halaman tesaurus berisi data tesaurus yang disajikan berupa tabel. Tabel yang ditampilkan berisi id, kata, bentuk, sinonim dan antonim. Antarmuka halaman tesaurus dapat dilihat pada gambar 4.14. Untuk menggunakan fitur tambah baru, admin diharuskan untuk memasukkan beberapa data yaitu kata, bentuk, sinonim dan antonim. Antarmuka halaman tesaurus fitur tambah baru dapat dilihat pada gambar 4.15. Fitur *edit* dapat mengubah data tesaurus berupa kata, bentuk, sinonim dan antonim berdasarkan id. Antarmuka halaman tesaurus fitur *edit* dapat dilihat pada gambar 4.16. Admin juga dapat menghapus data tesaurus dengan menekan tombol hapus. Ketika tombol hapus diklik, akan muncul peringatan sebelum menghapus data. Jika benar akan menghapus data, maka admin dapat mengkonfirmasi dengan menekan tombol ok atau tombol *cancel* untuk membatalkan perintah hapus seperti pada gambar 4.17.

Gambar 4.14 Antarmuka Halaman Tesaurus

Gambar 4.15 Antarmuka Halaman Tambah Tesaurus

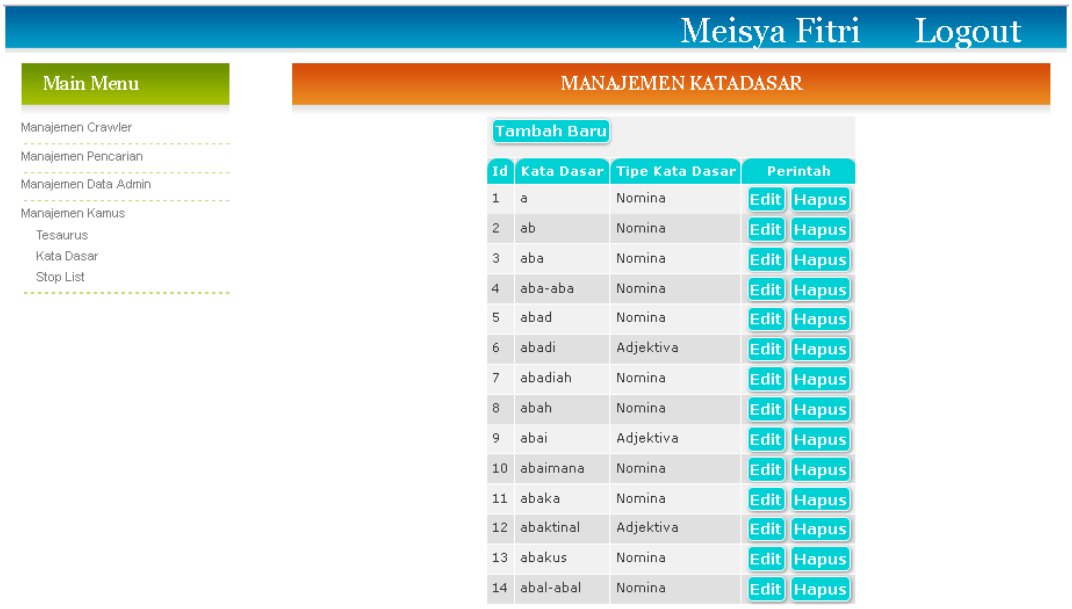
Gambar 4.16 Antarmuka Halaman *Edit* Tesaurus

Gambar 4.17 Pesan Peringatan Hapus Data Tesaurus

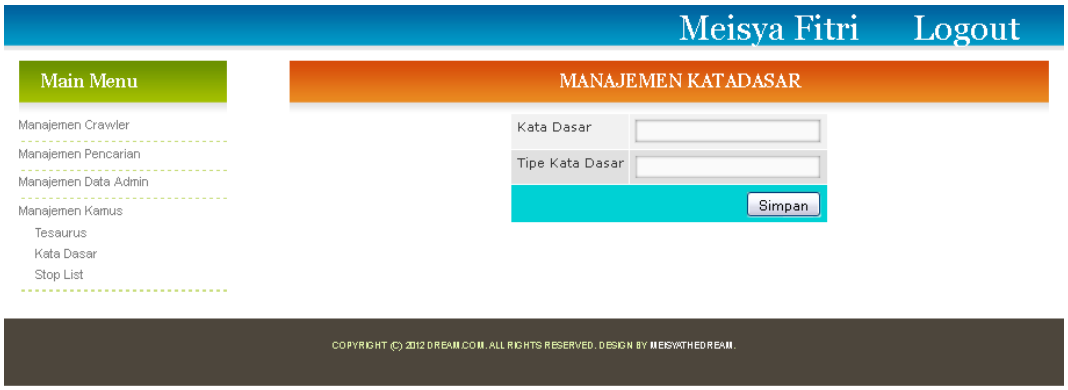
2. Halaman Kata Dasar

   Halaman kata dasar berisi data kata dasar yang disajikan berupa tabel. Tabel yang ditampilkan berisi id, kata dasar, dan tipe kata dasar. Antarmuka halaman kata dasar dapat dilihat pada gambar 4.18. Untuk menggunakan fitur tambah baru, admin diharuskan untuk memasukkan kata dasar dan tipe kata dasar. Antarmuka halaman kata dasar fitur tambah baru dapat dilihat pada gambar 4.19. Fitur *edit* dapat mengubah data kata dasar berupa kata dasar dan

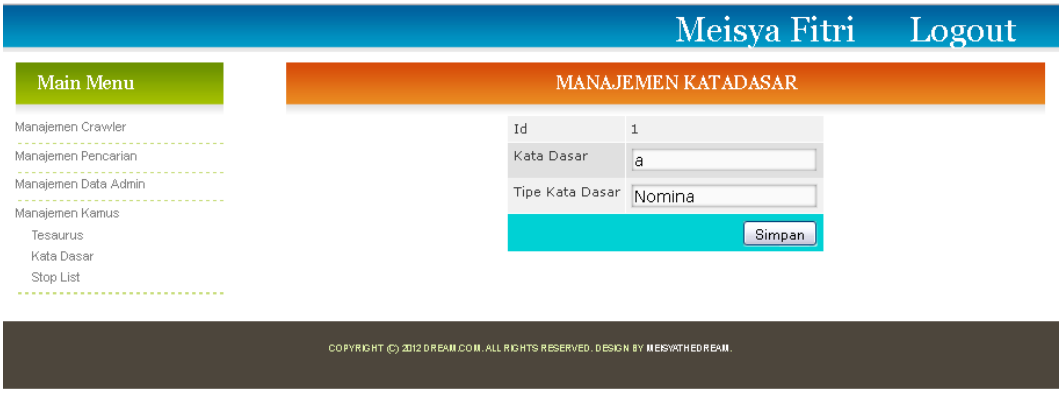
tipe kata dasar berdasarkan id. Antarmuka halaman kata dasar fitur *edit* dapat dilihat pada gambar 4.20. Admin juga dapat menghapus data kata dasar dengan menekan tombol hapus. Ketika tombol hapus diklik, akan muncul peringatan sebelum menghapus data. Jika benar akan menghapus data, maka admin dapat mengkonfirmasi dengan menekan tombol ok atau tombol *cancel* untuk membatalkan perintah hapus seperti pada gambar 4.21.



Gambar 4.18 Antarmuka Halaman Kata Dasar



Gambar 4.19 Antarmuka Halaman Tambah Kata Dasar



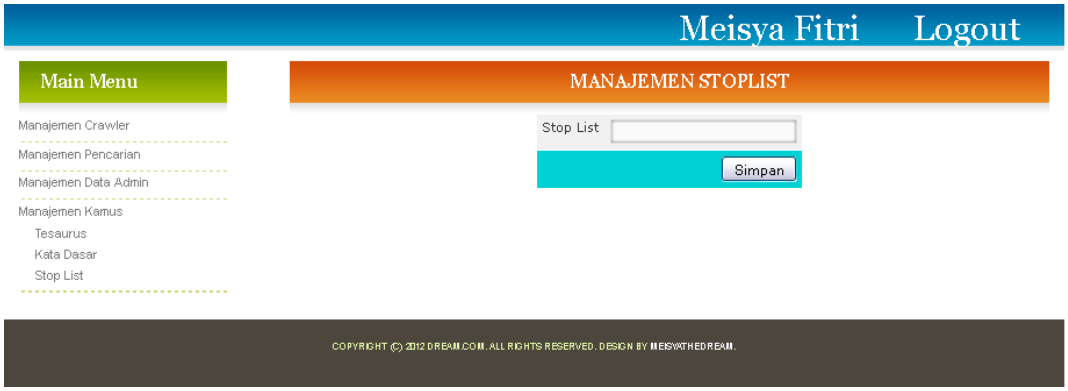
Gambar 4.20 Antarmuka Halaman *Edit* Kata Dasar

Gambar 4.21 Pesan Peringatan Hapus Data Kata Dasar

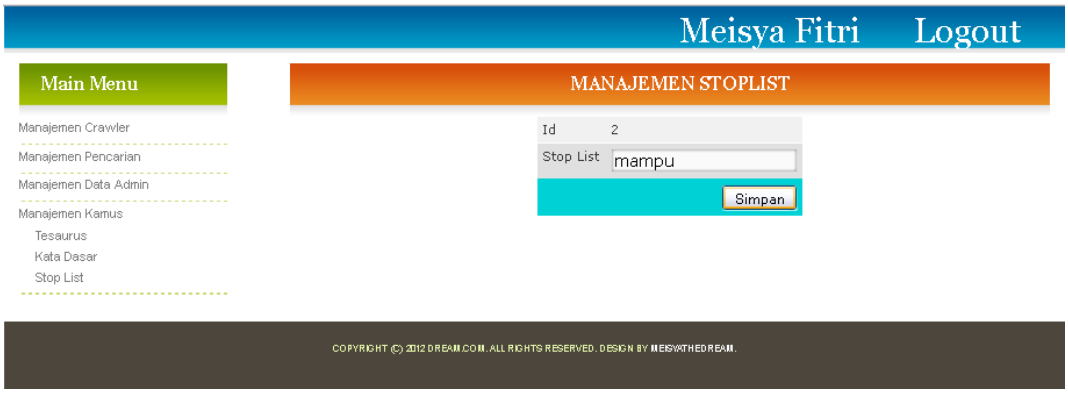
3. Halaman *Stop list*

   Halaman *stop list* berisi data kata buang yang disajikan berupa tabel. Tabel yang ditampilkan berisi id *stop list*. Antarmuka halaman *stop list* dapat dilihat pada gambar 4.22. Untuk menggunakan fitur tambah baru, admin diharuskan untuk memasukkan *stop list*. Antarmuka halaman *stop list* fitur tambah baru dapat dilihat pada gambar 4.23. Fitur *edit* dapat mengubah data *stop list* berupa *stop list* berdasarkan id. Antarmuka halaman *stop list* fitur *edit* dapat dilihat pada gambar 4.24. Admin juga dapat menghapus data *stop list* dengan menekan tombol hapus. Ketika tombol hapus diklik, akan muncul peringatan sebelum menghapus data. Jika benar akan menghapus data, maka admin dapat mengkonfirmasi dengan menekan tombol ok atau tombol *cancel* untuk membatalkan perintah hapus seperti pada gambar 4.25.

Gambar 4.22 Antarmuka Halaman *Stop list*



Gambar 4.23 Antarmuka Halaman Tambah *Stop list*



Gambar 4.24 Antarmuka Halaman *Edit Stop list*

Gambar 4.25 Pesan Peringatan Hapus Data *Stop list*

4.3 Hasil Pengoperasian Sistem

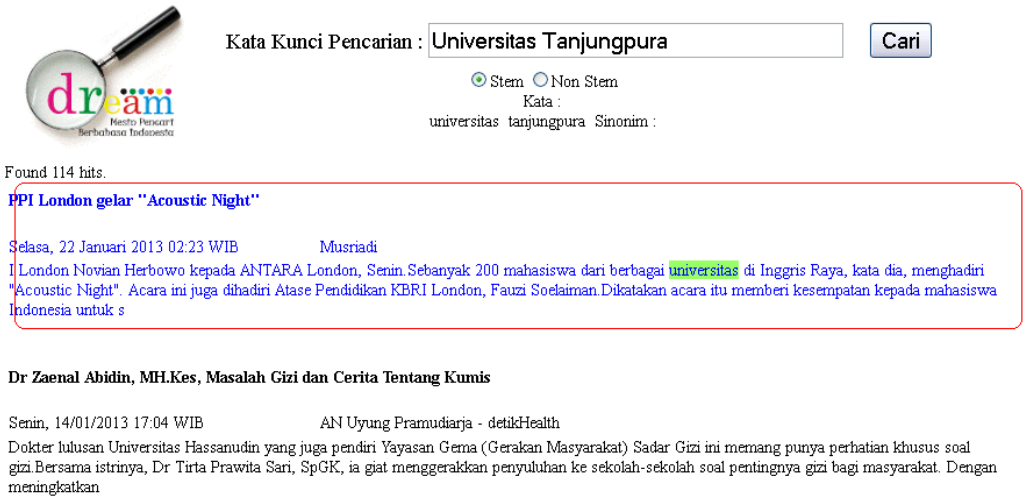
Berikut ini adalah contoh pengoperasian sistem dengan mengambil beberapa proses yeng dilakukan oleh *user* dan admin. Adapun cara pengoperasian sistem antara lain:

4.3.1 Pengoperasian Sistem sebagai *User*

Pada aplikasi ini *user* dapat melakukan pencarian dengan memasukkan kata kunci dan memilih fitur pencarian. Aplikasi akan mengeksekusi pencarian ketika tombol cari ditekan seperti pada gambar 4.26

Gambar 4.26 Antarmuka Halaman Pencarian

Setelah eksekusi pencarian aplikasi akan masuk ke dalam halaman hasil pencarian. Halaman hasil pencarian akan menampilkan hasil pencarian berdasarkan kata kunci yang telah terurut. Hasil pencarian ditampilkan berupa cuplikan berita yang memiliki kegunaan sebagai link ke sumber berita seperti pada gambar 4.27. Untuk masuk ke sumber berita asli, *user* hanya perlu mengklik pada hasil pencarian. Sumber berita asli akan dibuka melalui halaman baru di browser. Untuk mengubah kata kunci pencarian, *user* dapat memasukkan kata kunci yang baru di halaman hasil pencarian.



Gambar 4.27 Antarmuka Halaman Hasil Pencarian

4.3.2 Pengoperasian Sistem sebagai Admin

Pada aplikasi ini admin diharuskan untuk melakukan *login* untuk masuk ke dalam sistem. *Login* dilakukan dengan memasukkan *username* dan *password* yang valid pada halaman *login*. Jika *login* berhasil, admin akan masuk ke halaman utama admin. Untuk memanajemen aplikasi, admin dapat memilih menu utama yang terdapat dalam halaman admin.

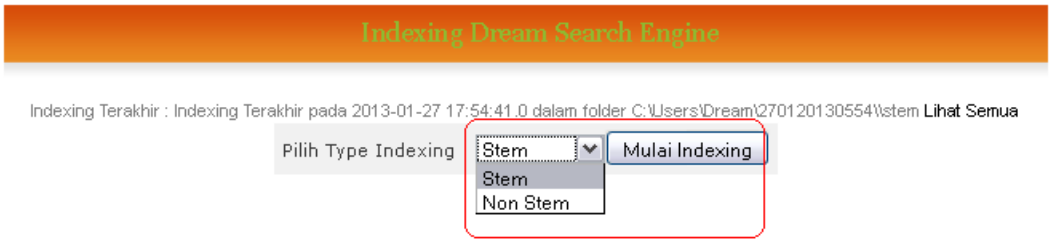
4.3.2.1 Pengoperasian *Crawler*

Pengoperasian *crawler* dilakukan setelah admin melewati proses *login*. Untuk memulai *crawling*, admin dapat masuk ke halaman manajemen *crawler* dengan menekan menu manajemen *crawler*. *Crawling* akan dimulai ketika admin menekan tombol ”Mulai Crawling” seperti pada gambar 4.28. ketika *crawling* selesai, aplikasi akan menampilkan pesan ” Crawling selesai,Mulai Indexing?” untuk mengkonfirmasi apakah admin akan melakukan proses *indexing* setelah proses *crawling* selesai. Jika tidak ingin melakukan *indexing*, admin dapat menekan tombol *cancel*. Sebaliknya, jika ingin melakukan *indexing*, admin harus menekan tombol ok. Selanjutkan akan muncul pesan ”Masukkan mode indexing yang digunakan:1. Stem 2. Non Stem” untuk memilih mode *indexing* yang digunakan. Jika admin ingin menggunakan mode stem, maka admin dapat memasukkan angka 1 pada pesan yang tampil, kemudian menekan ok. Untuk memilih mode non stem, masukkan angka 2. Jika ingin menghentikan *crawling* sebelum proses berakhir, admin dapat menekan tombol ”Stop Crawling”

Gambar 4.28 Antarmuka Halaman Manajemen *Crawler*

4.3.2.2 Pengoperasian *Indexer*

Pengoperasian *indexer* dilakukan setelah admin melewati proses *login*. Untuk memulai *indexing*, admin dapat masuk ke halaman manajemen pencarian dengan menekan menu manajemen pencarian. *Indexing* akan dimulai ketika admin menekan tombol ”Mulai Indexing” seperti pada gambar 4.29. Sebelum memulai *indexing*, admin dapat menentukan mode *indexing* yang akan digunakan dengan memilih pada *combobox*. Ketika *indexing* selesai, aplikasi akan menampilkan pesan ”selesai”. Kemudian directori tempat *file index* disimpan dan waktu terakhir *indexing* dilakukan akan tampil di halaman ini.



Gambar 4.29 Antarmuka Halaman Manajemen Pencarian

4.4 Hasil Pengujian Sistem

4.4.1 Hasil Pengujian Sistem Temu Balik Informasi

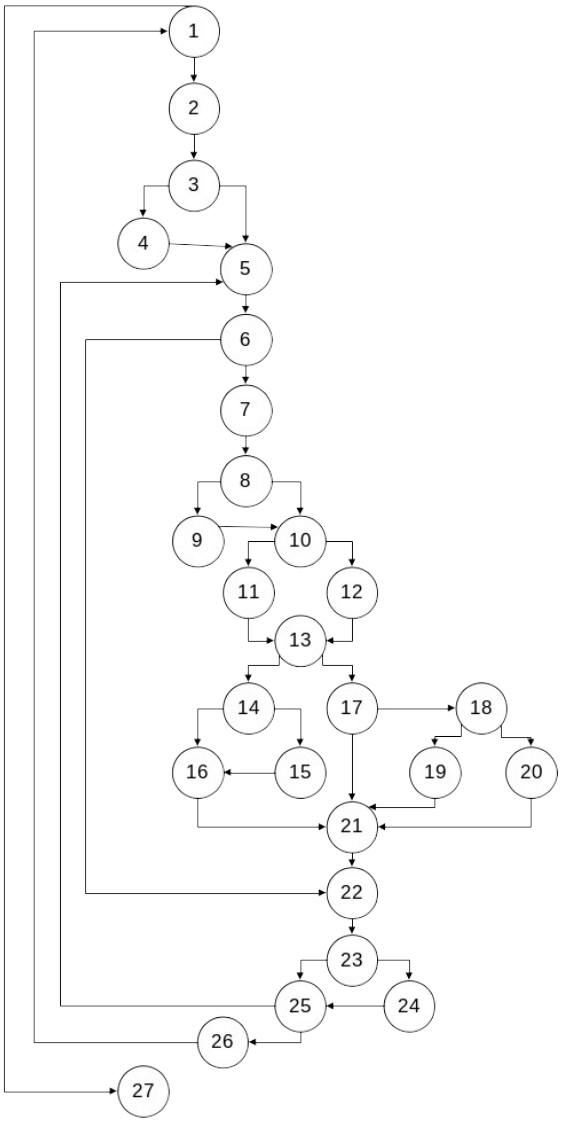
Berkenaan dengan Gambar 3.3 pada Bab III, sistem temu balik informasi dapat diuraikan pada beberapa bagian kode program berikut ini.

1. Prosedur *Crawling* website

```
Crawler cr = new Crawler();
int status = 0;
for (int q = 0; q<cr.getDomainUrlUtamaUntukCrawl().length;q++)                    1
    {
     int jmlCrawl = 0;
     String host1 = cr.getDomainUrlUtamaUntukCrawl()[q];
     alamat =host1;
     int msrto =cr.getmsRto()[q];
     linkweb = 100000;
     if (cr.isValidMyUrl(alamat,host1)&&!kon.cekrecorddokumen(alamat)) {          2,3
        String[] z = {alamat,"","","",""};                                       4
        kon.inserttable("dokumen", "", z);
        status = 1;
        }
        while (alamat == null ? "" != null : !alamat.equals("")){                 5
```

```
    String htmldump ="";
    status=4;
    if (cr.isValidMyUrl(alamat,host1)) {                                    6
      try {                                                                 7
       htmldump = cr.removetag(cr.scanweb(alamat,msrto).toString());}
      catch (MalformedURLException mue)
         {System.out.println(mue.toString());}
      catch (IOException ie) {System.out.println(ie.toString());}
      String namadomain = cr.gethost(alamat);
      if (linkweb<maxcrawl){                                                8
          List<String> ge = cr.getLink(htmldump);                           9
          insertlink(ge, namadomain,host1);
          }
      NewsExtractor NE= new NewsExtractor();
      String[] s = new String[5];
      if (htmldump == null ? "" != null : !htmldump.equals("")) {           10
          try {                                                             11
          String[] d = NE.extractor(htmldump, alamat);
          s[0] = alamat;
          s[1] = d[0].trim();
          s[2] = d[1].trim();
          s[3] = d[2].trim();
          s[4] = d[3].trim();
          } catch (Exception e) {System.out.println(e.toString());}
      }
      else {                                                                12
          status=3;
      }
      String id ="";
      if (kon.cekrecorddokumen(s[0])){                                      13
          if (s[1] == null ? "" != null : !s[1].equals(""))                 14
             if  (s[2] == null ? "" != null : !s[2].equals(""))
                if  (s[3] == null ? "" != null : !s[3].equals(""))
                   if  (s[4] == null ? "" != null : !s[4].equals("") )
                   {
                      kon.updaterecorddokumen(s);                           15
                   }
          status =1;                                                        16
      }
      else {                                                                17
          if (s[1] == null ? "" != null : !s[1].equals(""))                 18
             if  (s[2] == null ? "" != null : !s[2].equals(""))
                if  (s[3] == null ? "" != null : !s[3].equals(""))
                   if  (s[4] == null ? "" != null : !s[4].equals("") ){
                      kon.inserttable("dokumen", id, s);                    19
                      status =1;
                   }
          else {                                                            20
             s[1] = "";
             s[2] = "";
             s[3] = "";
             s[4] = "";
             kon.inserttable("dokumen", id, s);
             status=2;
          }
      }
      jmlCrawl++;                                                           21
    }
    kon.updatestatusdokumen(alamat,status);                                 22
    alamat = kon.geturl();
    if (jmlCrawl>=maxcrawl) {                                               23
       alamat = "";                                                         24
    }
  }                                                                         25
  kon.deletessemuadokumen0();                                               26
}                                                                           27
```

Berikut grafik aliran dari prosedur *crawling* website



Gambar 4.30 Grafik Aliran dari Prosedur *Crawling* Website

Menghitung kompleksitas siklomatik (*cyclomatic complexity*) dari grafik aliran yang diperoleh. Berdasarkan Gambar 4.30, kumpulan jalur independen pada grafik aliran dapat ditentukan sebagai berikut.

Jalur 1 : 1-27

Jalur 2 : 1-2-3-5-6-22-23-25-26-1-27

Jalur 3 : 1-2-3-4-5-6-22-23-25-26-1-27

Jalur 4 : 1-2-3-5-6-22-23-25-26-1-27

Jalur 5 : 1-2-3-5-6-22-23-24-25-26-1-27

Jalur 6 : 1-2-3-5-6-22-23-25-5-6-22-23-25-26-1-27

Jalur 7 : 1-2-3-5-6-7-8-9-10-11-13-14-16-21-22-23-25-26-1-27

Jalur 8 : 1-2-3-5-6-7-8-10-11-13-14-16-21-22-23-25-26-1-27

Jalur 9 : 1-2-3-5-6-7-8-10-12-13-14-15-16-21-22-23-25-26-1-27

Jalur 10 : 1-2-3-5-6-7-8-10-12-13-17-21-22-23-25-26-1-27

Jalur 11 : 1-2-3-5-6-7-8-10-12-13-17-18-19-21-22-23-25-26-1-27

Jalur 12 : 1-2-3-5-6-7-8-10-12-13-17-18-20-21-22-23-25-26-1-27

Dengan demikian, *cyclomatic complexity* dapat dihitung dengan tiga cara sebagai berikut.

1. Jumlah region grafik aliran sesuai dengan *cyclomatic complexity*. Grafik aliran mempunyai 12 *region*.
2. CC = jumlah edge – jumlah node + 2

   Dari Gambar 4.22 diketahui jumlah *edge* = 37 sedangkan jumlah *node* = 27, sehingga jumlah *cyclomatic complexity* dapat diperoleh sebagai berikut.

   CC = 37 *edge* – 27 *node* + 2 = 12
3. CC = jumlah *predicate node* + 1

   Dari Gambar 4.30 diketahui jumlah *predicate node* = 11, sehingga jumlah *cyclomatic complexity* dapat diperoleh sebagai berikut.

   CC = 11 *predicate node* + 1 = 12.

Jadi berdasarkan tiga cara perhitungan di atas, *cyclomatic complexity* untuk grafik aliran Gambar 4.30 adalah 12. Dengan demikian, hasil pengujian *white box* pada prosedur *crawling* website adalah sebagai berikut.

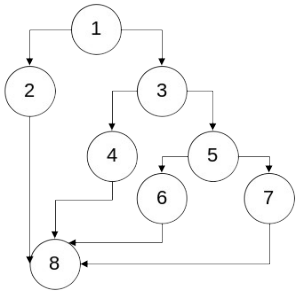
Tabel 4.1 Hasil Pengujian *White Box* Prosedur *Crawling* Website

Jalur	**Kompleksitas Siklomatik (*Cyclomatic Complexity*)**			**Keterangan**
	Region	***edge – node* + 2**	***predict node* + 1**	
1-27	12	37 - 27 + 2 = 12	11+ 1 = 12	Prosedur *Crawling* dokumen melalui 12 region. Alur proses pada prosedur *crawling* dokumen tersebut diuraikan pada jalur.
1-2-3-5-6-22-23-25-26-1-27				
1-2-3-4-5-6-22-23-25-26-1-27				
1-2-3-5-6-22-23-25-26-1-27				
1-2-3-5-6-22-23-24-25-26-1-27				
1-2-3-5-6-22-23-25-5-6-22-23-25-26-1-27				
1-2-3-5-6-7-8-9-10-11-13-14-16-21-22-23-25-26-1-27				
1-2-3-5-6-7-8-10-11-13-14-16-21-22-23-25-26-1-27				
1-2-3-5-6-7-8-10-12-13-14-15-16-21-22-23-25-26-1-27				
1-2-3-5-6-7-8-10-12-13-17-21-22-23-25-26-1-27				
1-2-3-5-6-7-8-10-12-13-17-18-19-21-22-23-25-26-1-27				
1-2-3-5-6-7-8-10-12-13-17-18-20-21-22-23-25-26-1-27				

2. Prosedur *Extractor* Berita

```
String[] news = new String[4];
if (alamat.contains("detik.com")){                                    1
   try {                                                              2
      news = detikextractor(html);
      } catch (StackOverflowError e) {System.out.println(e.toString());}
} else if (alamat.contains("antaranews.com")) {                       3
   news = antaranewsextractor(html);                                  4
} else if (alamat.contains("pontianak.tribunnews.com")) {             5
   news = pontianaktribunnewsextractor(html);                         6
} else {                                                              7
return null;
}                                                                     8
return news;
```

Berikut grafik aliran dari prosedur *extractor* berita



Gambar 4.31 Grafik Aliran dari Prosedur *Extractor* Berita

Berdasarkan Gambar 4.31, kumpulan jalur independen pada grafik aliran dapat ditentukan sebagai berikut.

Jalur 1 : 1-2-8

Jalur 2 : 1-3-4-8

Jalur 3 : 1-3-5-6-8

Jalur 4 : 1-3-5-7-8

Dengan demikian, *cyclomatic complexity* dapat dihitung dengan tiga cara sebagai berikut.

1. Jumlah *region* grafik aliran sesuai dengan *cyclomatic complexity*.
   Grafik aliran mempunyai 4 *region*.
2. CC = jumlah *edge* – jumlah *node* + 2
   Dari Gambar 4.22 diketahui jumlah *edge* = 10 sedangkan jumlah *node* = 8, sehingga jumlah *cyclomatic complexity* dapat diperoleh sebagai berikut.
   CC = 10 *edge* – 8 *node* + 2 = 4
3. CC = jumlah *predicate node* + 1
   Dari Gambar 4.31 diketahui jumlah *predicate node* = 3, sehingga jumlah *cyclomatic complexity* dapat diperoleh sebagai berikut.
   CC = 3 *predicate node* + 1 = 4.

Jadi berdasarkan tiga cara perhitungan di atas, *cyclomatic complexity* untuk grafik aliran Gambar 4.31 adalah 4 Dengan demikian, hasil pengujian *white box* pada prosedur extractor berita adalah sebagai berikut.

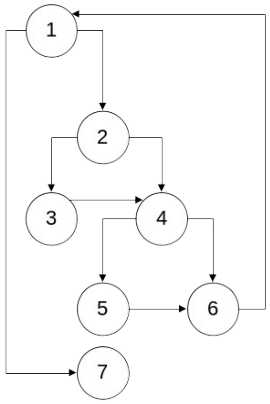
Tabel 4.2 Hasil Pengujian *White Box* Prosedur *Extractor* Berita

Jalur	**Kompleksitas Siklomatik (*Cyclomatic Complexity*)**			**Keterangan**
	Region	***edge – node* + 2**	***predict node* + 1**	
1-2-8	4	10 – 8 + 2 = 4	3 + 1 = 4	Prosedur *extractor* berita melalui 4 *region*. Alur proses pada prosedur *extractor* berita tersebut diuraikan pada jalur.
1-3-4-8				
1-3-5-6-8				
1-3-5-7-8				

3. Prosedur Pembobotan Kata

```
IndexSearcher searcher = new IndexSearcher(reader);
TermFreqVector freqVector = reader.getTermFreqVector(doc, "indexstring1");
int freqs[] = freqVector.getTermFrequencies();
int numdocs = reader.numDocs();
String[] term = freqVector.getTerms();
String[] termquery = query.toString().replaceAll("indexstring1", "").replaceAll(":", "").split(" ");
float total = 0;
for (int i=0;i<termquery.length;i++) {                                    1
   Term terma = new Term("indexstring1", termquery[i]);
   int docfreq = searcher.docFreq(terma);
   int index =-1;
   if (Arrays.asList(term).contains(termquery[i])){                       2
         index = Arrays.asList(term).indexOf(termquery[i]);               3
       }
   double idf = (Math.log((double)numdocs/(docfreq)));
   if (index != -1) {                                                     4
      System.out.println(termquery[i]+" "+freqs[index]+" "+idf);          5
      total += (freqs[index]*idf);
   }
}                                                                         6
return total;                                                             7
```

Berikut grafik aliran dari prosedur pembobotan kata



Gambar 4.32 Grafik Aliran dari Prosedur Pembobotan Kata

Berdasarkan Gambar 4.32, kumpulan jalur independen pada grafik aliran dapat ditentukan sebagai berikut.

Jalur 1 : 1-7

Jalur 2 : 1-2-3-4-6-1-7

Jalur 3 : 1-2-4-6-1-7

Jalur 4 : 1-2-4-5-6-1-7

Dengan demikian, *cyclomatic complexity* dapat dihitung dengan tiga cara sebagai berikut.

1. Jumlah *region* grafik aliran sesuai dengan *cyclomatic complexity*.
   Grafik aliran mempunyai 4 *region*. CC = jumlah *edge* – jumlah *node* + 2
   Dari Gambar 4.22 diketahui jumlah *edge* = 9 sedangkan jumlah *node* = 7, sehingga jumlah *cyclomatic complexity* dapat diperoleh sebagai berikut.
   CC = 9 *edge* – 7 *node* + 2 = 4
2. CC = jumlah *predicate node* + 1
   Dari Gambar 4.32 diketahui jumlah *predicate node* = 3, sehingga jumlah *cyclomatic complexity* dapat diperoleh sebagai berikut.
   CC = 3 *predicate node* + 1 = 4.

Jadi berdasarkan tiga cara perhitungan di atas, *cyclomatic complexity* untuk grafik aliran Gambar 4.32 adalah 4 Dengan demikian, hasil pengujian *white box* pada prosedur pembobotan kata adalah sebagai berikut.

Tabel 4.3 Hasil Pengujian *White Box* Prosedur Pembobotan Kata

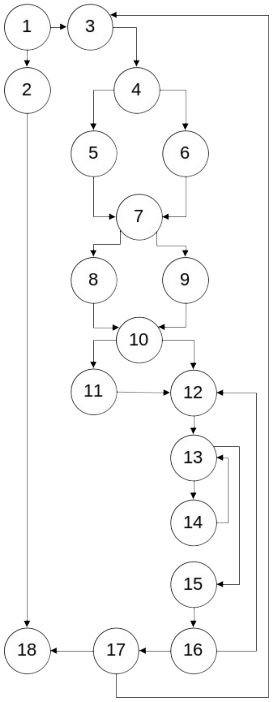
Jalur	**Kompleksitas Siklomatik (*Cyclomatic Complexity*)**			**Keterangan**
	Region	***edge – node* + 2**	***predict node* + 1**	
1-7	4	9 – 7 + 2 = 4	3 + 1 = 4	Prosedur pembobotan kata melalui 4 *region*. Alur proses pada prosedur pembobotan kata tersebut diuraikan pada jalur.
1-2-3-4-6-1-7				
1-2-4-6-1-7				
1-2-4-5-6-1-7				

4. Prosedur Pencarian

```
    String[] answer = new String[token.length+sinonim.length];
    System.arraycopy(token, 0, answer, 0, token.length);
    System.arraycopy(sinonim, 0, answer, token.length, sinonim.length);
    answer = new EliminasiStopWord().eliminasi(answer);
    String tagHasilCari = "";
if (answer.length==0)                                                    1
    {
       tagHasilCari = "";                                                2
    }

else {                                                                   3
        Koneksi kon =new Koneksi();
        String indexdir = kon.getWaktuLastIndex();
    IndexerDream idxdream = new IndexerDream();
        WhitespaceAnalyzer analyzer = new WhitespaceAnalyzer(Version.LUCENE_36);
        SimpleFSDirectory index = new SimpleFSDirectory(new File(indexdir));
        String querystr = convertToSpaceDelimited(answer);


Query q = new QueryParser(Version.LUCENE_36, "indexstring1", analyzer).parse(querystr);
IndexReader reader = IndexReader.open(index);
IndexSearcher searcher = new IndexSearcher(reader);
searcher.setSimilarity(new DreamSimilarity());
        CustomScoreQuery customQuery = new DreamScorer(q);
```

```
   int startidx;
   if (page-1!=0) {                                                                    4
      startidx = (page-1)*hitsPerPage;                                                 5
   } else {                                                                            6
      startidx = 0;
   }
         TopDocs topdoc = null;                                                        7
         topdoc = searcher.search(customQuery.createWeight(searcher), null, startidx+hitsPerPage);
   tagHasilCari += "<table><tr><td colspan=2>";
   tagHasilCari += "Found " + topdoc.totalHits + " hits.";
   tagHasilCari += "</td>";
   tagHasilCari += "</tr>";
   tagHasilCari += "<tr><td>";
   totalhits = topdoc.totalHits;
 int endidx;
 if ((startidx + hitsPerPage) < topdoc.scoreDocs.length) {                           8
 endidx = startidx + hitsPerPage;                                                    9
 } else {                                                                            10
 endidx = topdoc.scoreDocs.length;
 }
int pageCount = topdoc.totalHits/ hitsPerPage;
 if (topdoc.totalHits%hitsPerPage != 0) {                                            11
 pageCount += 1;                                                                     12
 }
   for(int i=startidx;i<endidx;++i) {                                                13
     int docId = topdoc.scoreDocs[i].doc;
     Document d = searcher.doc(docId);
     contentcutter cc = new contentcutter();
     String tmprContent;
     try {
     tmprContent = cc.cut(d.get("content"),token,sinonim);
       }
     catch(Exception e) {tmprContent = d.get("content"); }
     StemmerDream stem = new StemmerDream();
     for (int j = 0; j < answer.length; j++) {                                       14
        String s = stem.getKtDasar(answer[j]);                                       15
        tmprContent = tmprContent.replaceAll(s, "<span style='background-color: #99FF66'>"+s+"</span>");
     }

     tagHasilCari += "<a href=\""+d.get("url")+"\""+" target=\"_blank\""+">";        16
     tagHasilCari += "<table width=100% class='trlink'>";
     tagHasilCari += "<tr><td colspan=2><h4>"+d.get("judul") +"</h4></td></tr>";
     tagHasilCari += "<tr><td width=30%>"+ d.get("date") + "</td><td>"+ d.get("reporter") +"</td></tr>";
     tagHasilCari += "<tr ><td colspan=2>"+tmprContent+"</td></tr>";
     tagHasilCari += "</table>";
     tagHasilCari += "</a>";
     tagHasilCari += "<br><br>";
   }

   tagHasilCari += "</td>";
   tagHasilCari += "</tr>";
   tagHasilCari += "</table>";
   searcher.close();
 }                                                                                   17
      return tagHasilCari;                                                           18
```

Berikut grafik aliran dari prosedur pencarian



Gambar 4.33 Grafik Aliran dari Prosedur Pencarian

Berdasarkan Gambar 4.33, kumpulan jalur independen pada grafik aliran dapat ditentukan sebagai berikut.

Jalur 1 : 1-2-18

Jalur 2 : 1-3-4-5-7-8-10-11-12-13-15-16-17-18

Jalur 3 : 1-3-4-6-7-8-10-11-12-13-15-16-17-18

Jalur 4 : 1-3-4-6-7-9-10-11-12-13-15-16-17-18

Jalur 5 : 1-3-4-6-7-9-10-12-13-15-16-17-18

Jalur 6 : 1-3-4-6-7-9-10-12-13-15-14-13-15-16-17-18

Jalur 7 : 1-3-4-6-7-9-10-11-12-13-15-16-12-13-15-16-17-18

Jalur 8 : 1-3-4-6-7-9-10-11-12-13-15-16-17-3-4-6-7-9-10-12-13-15-16-17-18

Dengan demikian, *cyclomatic complexity* dapat dihitung dengan tiga cara sebagai berikut.

1. Jumlah *region* grafik aliran sesuai dengan *cyclomatic complexity*.
   Grafik aliran mempunyai 8 *region*.
   CC = jumlah *edge* – jumlah *node* + 2
2. Dari Gambar 4.33 diketahui jumlah *edge* = 24 sedangkan jumlah *node* = 18, sehingga jumlah *cyclomatic complexity* dapat diperoleh sebagai berikut.

CC = 24 *edge* – 18 *node* + 2 = 8

3. CC = jumlah *predicate node* + 1

   Dari Gambar 4.33 diketahui jumlah *predicate node* = 7, sehingga jumlah *cyclomatic complexity* dapat diperoleh sebagai berikut.

   CC = 7 *predicate node* + 1 = 8.

Jadi berdasarkan tiga cara perhitungan di atas, *cyclomatic complexity* untuk grafik aliran Gambar 4.33 adalah 8 Dengan demikian, hasil pengujian *white box* pada prosedur pencarian adalah sebagai berikut.

Tabel 4.4 Hasil Pengujian *White Box* Prosedur Pencarian

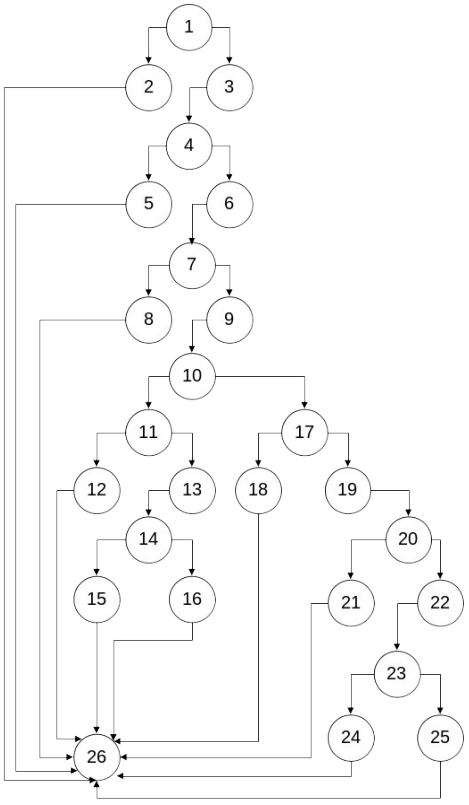
Jalur	**Kompleksitas Siklomatik (*Cyclomatic Complexity*)**			**Keterangan**
	Region	***edge – node* + 2**	***predict node* + 1**	
1-2-18	8	24 – 18 + 2 = 8	7 + 1 = 8	Prosedur pencarian melalui 8 *region*. Alur proses pada prosedur pencarian tersebut diuraikan pada jalur.
1-3-4-5-7-8-10-11-12-13-15-16-17-18				
1-3-4-6-7-8-10-11-12-13-15-16-17-18				
1-3-4-6-7-9-10-11-12-13-15-16-17-18				
1-3-4-6-7-9-10-12-13-15-16-17-18				
1-3-4-6-7-9-10-12-13-15-14-13-15-16-17-18				
1-3-4-6-7-9-10-11-12-13-15-16-12-13-15-16-17-18				
1-3-4-6-7-9-10-11-12-13-15-16-17-3-4-6-7-9-10-12-13-15-16-17-18				

5. Prosedur *Stemming*

```
String tmpStr = KataAwal.trim().toLowerCase();
if(cekKataDasar(tmpStr)){                                   1
return tmpStr;                                              2
}
else {                                                      3
   tmpStr = getkataTANPAPartikel(tmpStr);
   if(cekKataDasar(tmpStr)){                                4
      return tmpStr;                                        5
   }
   else {                                                   6
   tmpStr = getkataTANPAPossesivePronoun(tmpStr);
   if(cekKataDasar(tmpStr)){                                7
      return tmpStr;                                        8
   }
   else {                                                   9
   String tmpStrP1 = getkataTANPAAwalanPertama(tmpStr);
   if (tmpStr.equalsIgnoreCase(tmpStrP1)){                  10
```

```
    tmpStr = getkataTANPAAwalanKedua(tmpStr);
    if(cekKataDasar(tmpStr)){                                    11
    return tmpStr;                                               12
    } else {                                                     13
       tmpStr = getkataTANPAAkhiran(tmpStr, "");
       if(cekKataDasar(tmpStr)){                                 14
       return tmpStr;                                            15
       } else {return KataAwal;}                                 16
    }
  } else if(cekKataDasar(tmpStrP1)){                             17
    return tmpStrP1;                                             18
    } else {                                                     19
       String prefik = tmpStr.substring(0, tmpStr.length()-tmpStrP1.length());
       String tmpStrP2 = getkataTANPAAkhiran(tmpStrP1, prefik);
       if (tmpStrP2.equals(tmpStrP1)) {                          20
          return tmpStrP1;                                       21
       } else {                                                  22
          tmpStrP2 = getkataTANPAAwalanKedua(tmpStrP2);
          if(cekKataDasar(tmpStrP2)){                            23
             return tmpStrP2;                                    24
          } else {return KataAwal;}                              25
          }
       }
    }
    }
  }                                                              26
```

Berikut grafik aliran dari prosedur *Stemming*



Gambar 4.34 Grafik Aliran dari Prosedur *Stemming*

Berdasarkan Gambar 4.34, kumpulan jalur independen pada grafik aliran dapat ditentukan sebagai berikut.

Jalur 1 : 1-2-26

Jalur 2 : 1-3-4-5-26

Jalur 3 : 1-3-4-6-7-8-26

Jalur 4 : 1-3-4-6-7-9-10-11-12-26

Jalur 5 : 1-3-4-6-7-9-10-11-13-14-15-26

Jalur 6 : 1-3-4-6-7-9-10-11-13-14-16-26

Jalur 7 : 1-3-4-6-7-9-10-11-17-18-26

Jalur 8 : 1-3-4-6-7-9-10-11-17-19-20-21-26

Jalur 9 : 1-3-4-6-7-9-10-11-17-19-20-22-23-24-26

Jalur 10 : 1-3-4-6-7-9-10-11-17-19-20-22-23-25-26

Dengan demikian, *cyclomatic complexity* dapat dihitung dengan tiga cara sebagai berikut.

1. Jumlah *region* grafik aliran sesuai dengan *cyclomatic complexity*.
   Grafik aliran mempunyai 10 *region*.
   CC = jumlah *edge* – jumlah *node* + 2
2. Dari Gambar 4.34 diketahui jumlah *edge* = 34 sedangkan jumlah *node* = 26, sehingga jumlah *cyclomatic complexity* dapat diperoleh sebagai berikut.
   CC = 34 *edge* – 26 *node* + 2 = 10
3. CC = jumlah *predicate node* + 1
   Dari Gambar 4.34 diketahui jumlah *predicate node* = 9, sehingga jumlah *cyclomatic complexity* dapat diperoleh sebagai berikut.
   CC = 9 *predicate node* + 1 = 10.

Jadi berdasarkan tiga cara perhitungan di atas, *cyclomatic complexity* untuk grafik aliran Gambar 4.34 adalah 10 Dengan demikian, hasil pengujian *white box* pada prosedur *stemming* adalah sebagai berikut.

Tabel 4.5 Hasil Pengujian *White Box* Prosedur *Stemming*

Jalur	**Kompleksitas Siklomatik (*Cyclomatic Complexity*)**			**Keterangan**
	Region	***edge – node* + 2**	***predict node* + 1**	
1-2-26				Prosedur s*temming*

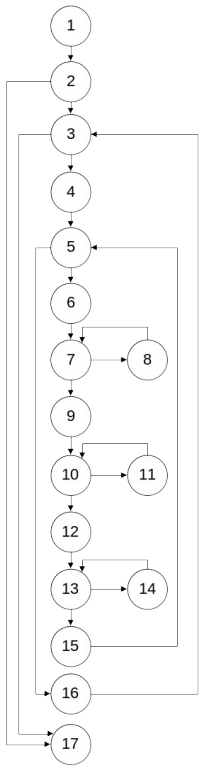
1-3-4-5-26	10	34 – 26 + 2 = 10	9 + 1 = 10	melalui 10 *region*. Alur proses pada prosedur *stemming* tersebut diuraikan pada jalur.
1-3-4-6-7-8-26				
1-3-4-6-7-9-10-11-12-26				
1-3-4-6-7-9-10-11-13-14-15-26				
1-3-4-6-7-9-10-11-13-14-16-26				
1-3-4-6-7-9-10-11-17-18-26				
1-3-4-6-7-9-10-11-17-19-20-21-26				
1-3-4-6-7-9-10-11-17-19-20-22-23-24-26				
1-3-4-6-7-9-10-11-17-19-20-22-23-25-26				

6. Prosedur *Indexing*

```
String message = "";
    try {                                                                                    1
       Koneksi kon = new Koneksi();
       String indexdir = this.setindexdir(stemnonstem);
    WhitespaceAnalyzer analyzer = new WhitespaceAnalyzer(Version.LUCENE_36);
   SimpleFSDirectory index = new SimpleFSDirectory(new File(indexdir));
   IndexWriterConfig config = new IndexWriterConfig(Version.LUCENE_36, analyzer);

   IndexWriter w = new IndexWriter(index, config);
   Koneksi kon = new Koneksi();
    try{
    Class.forName(kon.getdriver()).newInstance();
        java.sql.Connection conn = null;
        conn =
DriverManager.getConnection(kon.getdreamURL(),kon.getdreamUSER(),kon.getdreamPASS());

        Statement stmt = conn.createStatement();
        String query = "select count(id) as nb from dokumen where judul<>\"\" and date<>\"\" and
reporter<>\"\" and isi<>\"\" ";
        ResultSet rst = stmt.executeQuery(query);
      if(rst.next()) {                                                                       2
        int nb = rst.getInt("nb");
        int limit = 100;
        for(int j=0; j<nb; j+=limit) {                                                       3
   String sql = "select id,url,judul,date,reporter,isi from dokumen where judul<>\"\" and date<>\"\" and   4
reporter<>\"\" and isi<>\"\" limit "+j+","+limit;
   ResultSet rs = stmt.executeQuery(sql);
   String StemKatakunci = stemnonstem;
   while (rs.next()) {                                                                       5
      StringBuilder sb = new StringBuilder();                                               6

      String str = rs.getString("judul");
      String[] tokenstem = idxdream.prosesindexstring(str, StemKatakunci);
      int len = tokenstem.length;
      for(int i=0; i<len; i++) {                                                            7
      sb = sb.append(tokenstem[i]).append(" ");                                             8
      }
      str = rs.getString("reporter");                                                       9
      String[] token = tkn.getToken(str);
      len = token.length;
      for(int i=0; i<len; i++) {                                                            10
      sb = sb.append(token[i]).append(" ");                                                 11
      }
      str = rs.getString("isi");                                                            12
      tokenstem = idxdream.prosesindexstring(str, StemKatakunci);
      len = tokenstem.length;
      for(int i=0; i<len; i++) {                                                            13
      sb = sb.append(tokenstem[i]).append(" ");                                             14
      }
```

```
   String indexstring = sb.toString().toLowerCase();                                    15
   Document doc = new Document();
   doc.add(new Field("id", rs.getString("id"), Field.Store.YES, Field.Index.NOT_ANALYZED));
   doc.add(new Field("judul", rs.getString("judul"), Field.Store.YES, Field.Index.NOT_ANALYZED));
   doc.add(new Field("date", rs.getString("date"), Field.Store.YES, Field.Index.NOT_ANALYZED));
   doc.add(new Field("reporter", rs.getString("reporter"), Field.Store.YES, Field.Index.NOT_ANALYZED));
   doc.add(new Field("content", rs.getString("isi"), Field.Store.YES, Field.Index.NOT_ANALYZED));
   doc.add(new Field("url", rs.getString("url"), Field.Store.YES, Field.Index.NOT_ANALYZED));
   doc.add(new Field("indexstring1", indexstring, Field.Store.YES,
Field.Index.ANALYZED_NO_NORMS,Field.TermVector.YES));
   w.addDocument(doc);
}
          rs.close();                                                               16
          }
         }
          rst.close();                                                              17
          stmt.close();
          conn.close();
  }
  catch(Exception e){
             System.out.println("Exception: "+e);
          }
   w.close();
          String lastindex = kon.lastindex(indexdir);
   message = "selesai";
     }
     catch (IOException e) {message=e.toString();}
     finally {return  message;}
```

Berikut grafik aliran dari prosedur *Indexing*



Gambar 4.35 Grafik Aliran dari Prosedur *Indexing*

Berdasarkan Gambar 4.35, kumpulan jalur independen pada grafik aliran dapat ditentukan sebagai berikut.

Jalur 1 : 1-2-17

Jalur 2 : 1-2-3-17

Jalur 3 : 1-2-3-4-5-16-3-17

Jalur 4 : 1-2-3-4-5-6-7-9-10-12-13-15-5-16-3-17

Jalur 5 : 1-2-3-4-5-6-7-8-7-9-10-12-13-15-5-16-3-17

Jalur 6 : 1-2-3-4-5-6-7-9-10-11-10-12-13-15-5-16-3-17

Jalur 7 : 1-2-3-4-5-6-7-9-10-12-13-14-13-15-5-16-3-17

Dengan demikian, *cyclomatic complexity* dapat dihitung dengan tiga cara sebagai berikut.

1. Jumlah *region* grafik aliran sesuai dengan *cyclomatic complexity*.
   Grafik aliran mempunyai 7 *region*.
2. CC = jumlah *edge* – jumlah *node* + 2
   Dari Gambar 4.35 diketahui jumlah *edge* = 34 sedangkan jumlah *node* = 17, sehingga jumlah *cyclomatic complexity* dapat diperoleh sebagai berikut.
   CC = 22 *edge* – 17 *node* + 2 = 7
3. CC = jumlah *predicate node* + 1
   Dari Gambar 4.35 diketahui jumlah *predicate node* = 6, sehingga jumlah *cyclomatic complexity* dapat diperoleh sebagai berikut.
   CC = 6 *predicate node* + 1 = 7.

Jadi berdasarkan tiga cara perhitungan di atas, *cyclomatic complexity* untuk grafik aliran Gambar 4.35 adalah 7 Dengan demikian, hasil pengujian *white box* pada prosedur *indexing* adalah sebagai berikut.

Tabel 4.6 Hasil Pengujian *White Box* Prosedur *Indexing*

Jalur	**Kompleksitas Siklomatik (*Cyclomatic Complexity*)**			**Keterangan**
	Region	***edge – node* + 2**	***predict node* + 1**	
1-2-17				Prosedur *indexing*

1-2-3-17	7	22 – 17 + 2 = 7	6 + 1 = 7	melalui 7 *region*. Alur proses pada prosedur *indexing* tersebut diuraikan pada jalur.
1-2-3-4-5-16-3-17				
1-2-3-4-5-6-7-9-10-12-13-15-5-16-3-17				
1-2-3-4-5-6-7-8-7-9-10-12-13-15-5-16-3-17				
1-2-3-4-5-6-7-9-10-11-10-12-13-15-5-16-3-17				
1-2-3-4-5-6-7-9-10-12-13-14-13-15-5-16-3-17				

4.4.2 Hasil Pengujian Proses Pembobotan Kata

Pada pengujian proses ini digunakan data hasil *crawling* sebanyak masing-masing 20 berita dari 3 website. Sehingga jumlah dokumen yang digunakan adalah 30 berita. Pengujian dilakukan terhadap 5 macam kata kunci. Dokumen yang digunakan memiliki variasi jumlah kata dan halaman. Hal ini akan mempengaruhi bobot dari setiap dokumen pada saat pencarian.

Tabel 4.7 Pengujian Pembobotan Kata dengan *Indexing* Mode Stem

Kata Kunci	Jumlah Dokumen Relevan dalam Koleksi	*Indexing* Mode Stem									
		Jumlah Dokumen Relevan ditemukan		Jumlah dokumen ditemukan		Recall		Precision		NIAP	
		Stem	Non Stem	Stem	Non Stem	Stem	Non Stem	Stem	Non Stem	Stem	Non Stem
Penculikan anak	5	5	5	35	34	1.0000	1.0000	0.1316	0.1429	0.2282	0.2184
Pendidikan SMA	5	5	5	18	16	1.0000	1.0000	0.2778	0.3125	0.835	0.9667
Korupsi	1	1	1	1	1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Dahlan Iskan	1	1	1	1	1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Banjir Jakarta	15	15	15	43	43	1.0000	1.0000	0.3488	0.3488	0.7131	0.7323

Tabel 4.8 Pengujian Pembobotan Kata dengan *Indexing* Mode Non Stem

Kata Kunci	Jumlah Dokumen Relevan dalam Koleksi	*Indexing* Mode Non Stem									
		Jumlah Dokumen Relevan ditemukan		Jumlah dokumen ditemukan		Recall		Precision		NIAP	
		Stem	Non Stem	Stem	Non Stem	Stem	Non Stem	Stem	Non Stem	Stem	Non Stem

Penculikan anak	5	5	5	35	34	1.0000	1.0000	0.1429	0.1471	0.2108	0.2308
Pendidikan SMA	5	5	5	15	15	1.0000	1.0000	0.3333	0.3333	1.0000	0.9429
Korupsi	1	1	1	2	1	1.0000	1.0000	0.5000	1.0000	1.0000	1.0000
Dahlan Iskan	1	1	1	1	1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Banjir Jakarta	15	15	15	41	41	1.0000	1.0000	0.3659	0.3659	0.8956	0.7763

Dari perhitungan yang dilakukan pada tabel 4.7 dan table 4.8 didapatkan nilai *Recall, Precision* dan NIAP dari setiap kata kunci. Setiap kata kunci baik dalam pencariannya terhadap data *indexing* mode stem maupun data *indexing* mode non stem menghasilkan nilai *recall* 1 yang berarti semua dokumen relevan ditemukan dalam pencarian. Sebaliknya, Nilai perhitungan *precision* pada setiap kata kunci menghasilkan nilai yang bervariasi. Semakin tinggi nilai *precision* yang dihasilkan berarti semakin banyak dokumen relevan dalam dokumen yang ditemukan. Nilai *precision* memiliki nilai tertinggi 1 yang berarti seluruh dokumen yang ditemukan adalah relevan. Dari pengujian terhadap 5 kata kunci, didapatkan bahwa nilai *precision* akan meningkat ketika digunakan pada kata kunci non stem.

Nilai NIAP tertinggi adalah 1, yang berarti seluruh dokumen relevan berhasil ditemukan dengan seluruh dokumen relevan tersebut ditempatkan pada urutan teratas dalam hasil pencarian. Nilai NIAP yang didapatkan dalam pengujian memiliki nilai bervariasi. Nilai NIAP yang dihasilkan terhadap 5 kata kunci mencapai angka 1 sehingga untuk kata kunci tertentu, sistem dapat mengurutkan semua dokumen relevan ke urutan teratas.

4.4.3 Hasil Pengujian Kecepatan Pencarian

Pada pengujian proses ini digunakan data hasil *crawling* sebanyak masing-masing 20 berita dari 3 website. Sehingga jumlah dokumen yang digunakan adalah 30 berita. Pengujian dilakukan terhadap 5 macam kata kunci. Dokumen yang digunakan memiliki variasi jumlah kata dan halaman. Pengujian dilakukan

pada aplikasi Dream Search Engine dan sebagai pembanding digunakan aplikasi Giggle Search Engine yang juga dibangun dengan java namun menggunakan teknik dasar pembobotan yang dibangun Lucene yaitu TF, IDF dan *normalization*. Hal ini akan mempengaruhi kecepatan eksekusi pada saat pencarian.

Tabel 4.9 Pengujian Kecepatan Pencarian dalam detik

Kata Kunci	Kecepatan Pencarian Dream Search Engine				Kecepatan Pencarian Giggle Search Engine			
	Indexing Mode Stem		Indexing Mode Non Stem		Indexing Mode Stem		Indexing Mode Non Stem	
	Stem	Non Stem	Stem	Non Stem	Stem	Non Stem	Stem	Non Stem
Pendidikan SMA	3.697	3.9	2.918	3.604	1.654	0.011	1.671	0.05
Penculikan Anak	4.227	4.368	4.01	4.383	1.73	1.673	1.7	0.036
Dahlan Iskan	0.437	0.218	0.234	0.25	1.514	0.02	1.495	0.04
Banjir Jakarta	3.806	4.041	3.588	3.884	1.276	0.016	1.342	0.053
Korupsi	0.609	0.265	0.39	0.265	1.69	0.014	1.989	0.039

Ket ▭ menandakan tidak ditemukan apapun pada pencarian

Dari perhitungan yang dilakukan pada tabel 4.9 didapatkan bahwa kecepatan pencarian Dream Search Engine jauh dibawah kecepatan Giggle Search Engine terkecuali pada kata kunci “Dahlan Iskan” dan “Korupsi”. Perbedaan yang signifikan ini terjadi dikarenakan teknik *query expansion* yang diimplementasikan dalam Dream Search Engine sehingga *query* yang dieksekusi untuk pencarian berita menjadi panjang dan lebih banyak bobot yang diperhitungkan, sedangkan pada kata kunci “Dahlan Iskan” tidak ditemukan sinonim kata dalam *query expansion* sehingga kecepatan pencarian mengungguli kecepatan Giggle Search Engine. Kecepatan pencarian juga unggul ketika digunakan kata kunci “Korupsi” yang memiliki sinonim antara lain curang, manipulasi, gelap dan seleweng. Hal ini dikarenakan sinonim yang berjumlah 4 kata tidak secara signifikan mempengaruhi kecepatan pencarian Dream Search Engine sehingga pencarian dapat dilakukan kurang dari 1 detik.

Kecepatan pencarian bertambah secara signifikan hingga lebih dari 3 detik pada kata kunci “Penculikan Anak”, “Pendidikan SMA” dan “Banjir Jakarta”. Ini

dikarenakan sinonim yang dihasilkan oleh proses *query expansion* untuk kata kunci tersebut berjumlah banyak sehingga *query* menjadi panjang dan bobot yang harus diperhitungkan menjadi banyak. Pada pencarian dengan kata kunci "Penculikan Anak", bobot dari kata yang diperhitungkan meliputi penculikan, anak, cucu, ananda, anggota, arek, awing, bani, bayi, bocah, buah, hati, budak, bujang, bumiputra, buyung, darah, daging, entong, ibnu, kanak, turun, lerai, demam, putra, putri, ujang, yuana, duduk, warga, cabang, cawang, cah, ranting dan tunas yang berjumlah 35 kata. Pada pencarian dengan kata kunci "Pendidikan SMA", bobot dari kata yang diperhitungkan meliputi pendidikan, sma, bimbing, dik, edukasi, kuliah, pelajaran, latih, ajar, adab, bibit, pemeliharaan, cerah, asuh, gembleng, penggodokan, sekolah, tarbiah dan tuntun yang berjumlah 19 kata. Pada pencarian dengan kata kunci "Banjir Jakarta", bobot dari kata yang diperhitungkan meliputi banjir, jakarta, air, naik, pasang, ampuh, bah, empoh, genang, kayau, lembak, luap, sebak, limpah, tumpah dan ruah yang berjumlah 16 kata.

4.5 Analisis Hasil Pengujian Sistem

Dari pengujian sistem temu balik yang telah dilakukan, analisis yang dapat disimpulkan sebagai berikut:

1. Pengujian pada sistem temu balik informasi dengan menggunakan metode *white box* berhasil dilakukan dengan error nol, dimana setiap *statement* pada program telah dieksekusi paling tidak satu kali selama pengujian dan semua kondisi logis telah diuji dan berhasil.
2. Pencarian dokumen dengan kata kunci akan menampilkan semua dokumen yang mengandung kata kunci dan menampilkan dokumen paling relevan ke urutan teratas.
3. Proses pencarian dapat dilakukan oleh sistem dengan menampilkan hasil pencarian yang didalamnya terdapat dokumen relevan
4. Sistem dapat menemukan semua dokumen relevan.
5. Precision akan meningkat ketika pencarian dilakukan pada kata kunci yang non stem, dan jumlah dokumen yang ditemukan akan menurun.

6. *Indexing* non stem lebih baik dalam ketepatan mengurutkan dokumen relevan.
7. Dream Search Engine tidak lebih efisien dari Giggle Search Engine terkecuali ketika memiliki kata kunci yang tidak banyak memiliki sinonim.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan hasil analisis dan pengujian terhadap Sistem Temu Balik Informasi dengan Metode Pembobotan Kombinasi Tf-Idf yang digunakan untuk pencarian dokumen Berbahasa Indonesia, maka dapat ditarik kesimpulan sebagai berikut:

1. Sistem mampu mengumpulkan dokumen berita melalui proses *crawling* website dan memberikan bobot dengan mengimplementasikan metode pembobotan kata dengan metode kombinasi Tf-Idf secara lengkap sehingga lebih banyak data relevan yang dapat diperhitungkan dalam pencarian.
2. Sistem dapat melakukan proses pencarian dan menemukan informasi yang relevan berdasarkan hasil pengujian yang dilakukan pada 5 kata kunci. Data hasil pengujian menghasilkan nilai *recall* 1 yang menunjukkan bahwa semua dokumen yang relevan dapat ditemukan sistem dan nilai *precision* antara 0.1316 dan 1 yang menunjukkan terdapat dokumen lain selain dokumen relevan yang ikut ditemukan oleh sistem. Nilai NIAP yang dihasilkan mencapai nilai 1 yang menunjukkan sistem dapat mengurutkan dokumen relevan ke dalam urutan hasil pencarian teratas.
3. Sistem Dream Search Engine tidak lebih efisien dari Giggle Search Engine walaupun keduanya sama-sama menggunakan proses *indexing* terkecuali ketika memiliki kata kunci yang tidak banyak memiliki sinonim.

5.2 Saran

Adapun beberapa hal yang perlu ditambahkan dalam pengembangan sistem ini adalah sebagai berikut:

1. Perlu tambahan *extractor* berita sehingga berita yang ditemukan tidak terbatas pada website tertentu saja.
2. Perlu penyempurnaan *extractor* berita yang sudah ada sehingga pengenalan isi, tanggal, reporter dan judul pada semua dokumen tepat.

3. Dibuat suatu sistem untuk mengolah pengumpulan dokumen dari website yang mampu menangani dokumen dalam jumlah yang tak terbatas dengan penggunaan perangkat keras yang terdistribusi.
4. Perlu penyempurnaan teknik *query expansion* dengan menambahkan tesaurus yang lebih lengkap, sehingga kata hasil dari proses ini memiliki hubungan yang sangat erat dengan kata kunci dan tidak sekedar sinonim dari kata kunci yang dapat dipertimbangkan berdasarkan makna dari sebuah kata.