BAB II

TINJAUAN PUSTAKA

II.1 Data Mining

Data mining merupakan teknologi yang menggabungkan metoda analisis tradisional dengan algoritma yang canggih untuk memproses data dengan jumlah yang besar. Beberapa definisi *data mining* yang diantaranya adalah sebagai berikut :

1. *Data mining* adalah serangkaian proses untuk menggali nilai tambah dari suatu kumpulan data berupa pengetahuan yang selama ini tidak diketahui secara manual.
2. *Data mining* adalah analisa otomatis dari data yang berjumlah besar atau kompleks dengan tujuan untuk menemukan pola atau kecenderungan yang penting yang biasanya tidak didasari keberadaannya.
3. *Data mining* atau *Knowledge Discovery in Database* (KDD) adalah pengambilan informasi yang tersembunyi, dimana informasi tersebut sebelumnya tidak dikenal dan berpotensi bermanfaat. Proses ini meliputi sejumlah pendekatan teknis yang berbeda, seperti *clustering, data summarization, learning classification rules*.

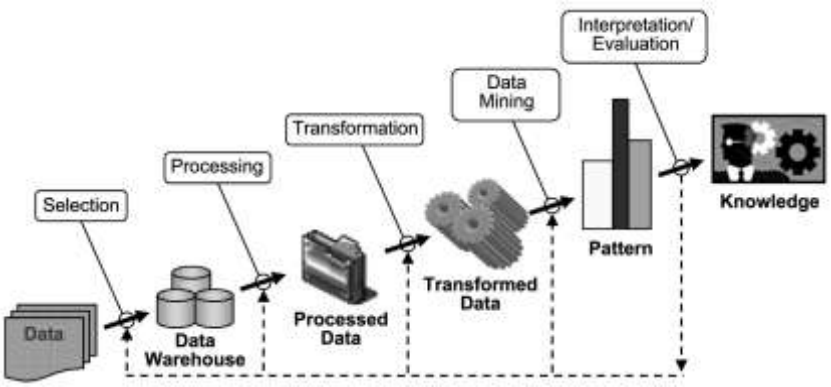
Secara umum, data mining dapat melakukan dua hal yaitu memberikan kesempatan untuk menemukan informasi menarik yang tidak terduga dan juga bisa menangani data berskala besar. Dalam menemukan informasi yang menarik ini, ciri khas data mining adalah kemampuan pencarian secara hampir otomatis,

karena dalam banyak teknik data mining ada beberapa parameter yang masih harus ditentukan secara manual atau semi manual. Data mining juga dapat memanfaatkan pengalaman atau bahkan kesalahan di masa lalu untuk meningkatkan kualitas dari model maupun hasil analisanya, salah satunya dengan kemampuan pembelajaran yang dimiliki beberapa teknik *data mining* seperti klasifikasi.

II.1.1 Tahapan *Data Mining*

Salah satu tuntutan dari *data mining* ketika diterapkan pada data berskala besar adalah diperlukan metodologi sistematis tidak hanya ketika melakukan analisa saja tetapi juga ketika mempersiapkan data dan juga melakukan interpretasi dari hasilnya sehingga dapat menjadi aksi ataupun keputusan yang bermanfaat.

Data mining seharusnya dipahami sebagai suatu proses, yang memiliki tahapan-tahapan tertentu dan juga ada umpan balik dari setiap tahapan ke tahapan sebelumnya. Pada umumnya proses *data mining* berjalan interaktif karena tidak jarang hasil *data mining* pada awalnya tidak sesuai dengan harapan analisnya sehingga perlu dilakukan desain ulang prosesnya.



Gambar II-1 Tahapan *data mining*

Sebagai suatu rangkaian proses, *data mining* dapat dibagi menjadi beberapa tahap yang diilustrasikan di Gambar II-1. Tahap-tahap tersebut bersifat interaktif di mana pemakai terlibat langsung atau dengan perantaraan *knowledge base*. Adapun untuk tahap-tahap *data mining* tersebut adalah sebagai berikut :

1. Pembersihan data

Pembersihan dilakukan untuk membuang data yang tidak konsisten dan berupa *noise*.

2. Integrasi Data

Data yang diperlukan untuk *data mining* tidak hanya berasal dari satu *database* tetapi juga berasal dari beberapa *database* atau file teks. Hasil integrasi data sering diwujudkan dalam sebuah *data warehouse* karena *dengan data warehouse*, data dikonsolidasikan dengan struktur khusus yang efisien. Selain itu *data warehouse* juga memungkinkan tipe analisa seperti OLAP.

3. Transformasi data

Transformasi dan pemilihan data ini untuk menentukan kualitas dari hasil *data mining*, sehingga data diubah menjadi bentuk sesuai untuk di-*mining*.

4. Aplikasi Teknik *Data Mining*

Aplikasi teknik *data mining* sendiri hanya merupakan salah satu bagian dari proses *data mining*.

5. Evaluasi pola yang ditemukan

Dalam tahap ini hasil dari teknik *data mining* berupa pola-pola yang khas maupun model prediksi dievaluasi untuk menilai apakah hipotesa yang ada memang tercapai.

6. Presentasi Pengetahuan

Presentasi pola yang ditemukan untuk menghasilkan aksi tahap terakhir dari proses *data mining* adalah bagaimana memformulasikan keputusan atau aksi dari hasil analisa yang didapat.

II.1.2 Teknik *Data Mining*

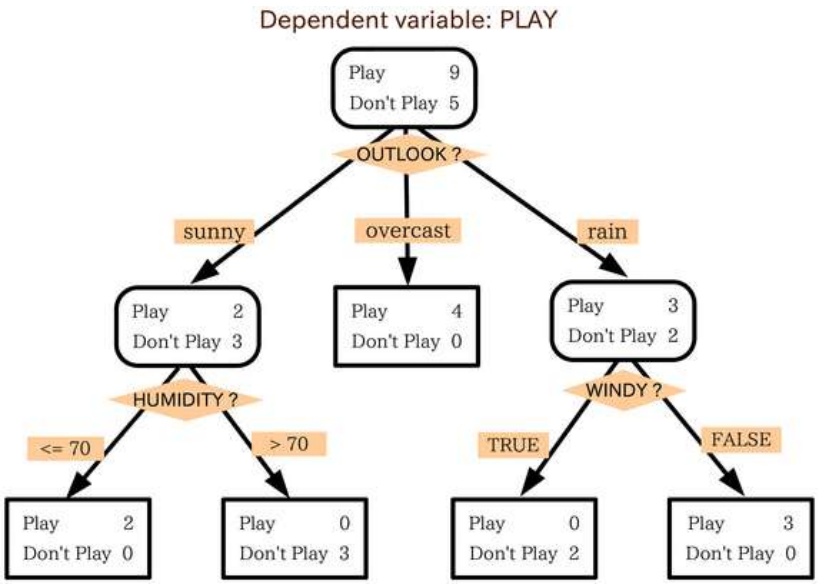
Data mining adalah serangkaian proses untuk menggali nilai tambah dari suatu kumpulan data berupa pengetahuan yang selama ini tidak diketahui secara manual. Perlu diingat bahwa kata *mining* sendiri berarti usaha untuk mendapatkan sedikit data berharga dari sejumlah besar data dasar. Karena itu *data mining* sebenarnya memiliki akar yang panjang dari bidang ilmu seperti kecerdasan buatan (*artificial intelligent*), *machine learning*, statistik dan basis data. Beberapa teknik yang sering disebut-sebut dalam literatur *data mining* antara lain yaitu *association rule mining*, *clustering*, klasifikasi, *neural network*, *genetic algorithm* dan lain-lain.

II.1.2.1 *Classification*

Classification adalah suatu teknik dengan melihat pada kelakuan dan atribut dari kelompok yang telah didefinisikan.Teknik ini dapat memberikan klasifikasi pada data baru dengan memanipulasi data yang ada yang telah diklasifikasi dan dengan menggunakan hasilnya untuk memberikan sejumlah aturan. Aturan-aturan tersebut digunakan pada data-data baru untuk diklasifikasi. Teknik ini menggunkan *supervised induction*, yang memanfaatkan kumpulan pengujian dari record yang terklasifikasi untuk menentukan kelas-kelas tambahan. Salah satu contoh yang mudah dan populer adalah dengan *Decision tree*, yaitu salah satu

metode klasifikasi yang paling populer karena mudah untuk diinterpretasi. Decision tree adalah model prediksi menggunakan struktur pohon atau struktur berhirarki.

Decision tree adalah struktur *flowchart* yang menyerupai *tree* (pohon), dimana setiap simpul internal menandakan suatu tes pada atribut, setiap cabang merepresentasikan hasil tes, dan simpul daun merepresentasikan kelas atau distribusi kelas. Alur pada *decision tree* di telusuri dari simpul akar ke simpul daun yang memegang prediksi kelas untuk contoh tersebut. *Decision tree* mudah untuk dikonversi ke aturan klasifikasi (*classification rules*).



Gambar II-2 Contoh *decision tree*

II.1.2.2 *Association*

Association digunakan untuk mengenali kelakuan dari kejadian-kejadian khusus atau proses dimana link asosiasi muncul pada setiap kejadian. Contoh dari aturan assosiatif dari analisa pembelian di suatu pasar swalayan adalah bisa diketahui berapa besar kemungkinan seorang pelanggan membeli roti bersamaan

dengan susu. Dengan pengetahuan tersebut pemilik pasar swalayan dapat mengatur penempatan barangnya atau merancang kampanye pemasaran dengan memakai kupon diskon untuk kombinasi barang tertentu.

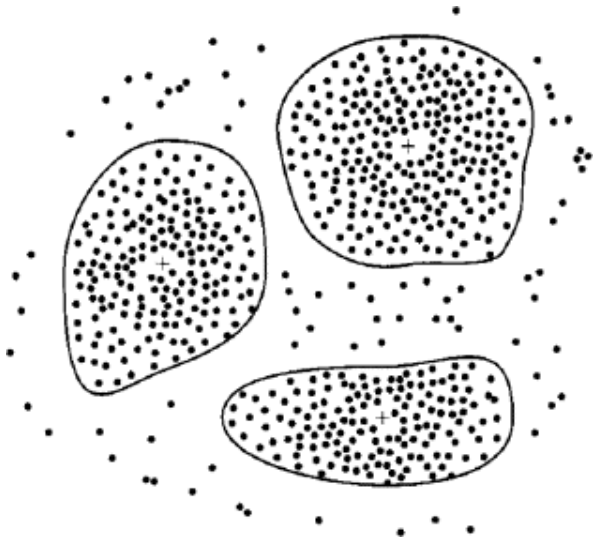
Penting tidaknya suatu aturan assosiatif dapat diketahui dengan dua parameter, *support* yaitu prosentasi kombinasi atribut tersebut dalam basis data dan *confidence* yaitu kuatnya hubungan antar atribut dalam aturan asosiatif. Motivasi awal pencarian *association rule* berasal dari keinginan untuk menganalisa data transaksi supermarket, ditinjau dari perilaku customer dalam membeli produk. *Association rule* ini menjelaskan seberapa sering suatu produk dibeli secara bersamaan. Sebagai contoh, association rule **"*beer* => *diaper* (80%)"** menunjukkan bahwa empat dari lima customer yang membeli *beer* juga membeli *diaper*. Dalam suatu association rule *X => Y, X* disebut dengan *antecedent* dan *Y* disebut dengan *consequent.*

II.1.2.3 *Clustering*

Clustering digunakan untuk menganalisis pengelompokkan berbeda terhadap data, mirip dengan klasifikasi, namun pengelompokkan belum didefinisikan sebelum dijalankannya tool *data mining*. Biasanya menggunakan metode *neural network* atau statistik. Clustering membagi item menjadi kelompok-kelompok yang ditemukan tool *data mining*.

Prinsip dari *clustering* adalah memaksimalkan kesamaan antar anggota satu kelas dan meminimumkan kesamaan antar *cluster*. *Clustering* dapat dilakukan pada data yang memiliki beberapa atribut yang dipetakan sebagai ruang multidimensi. Ilustrasi dari *clustering* dapat dilihat di gambar dibawah dimana

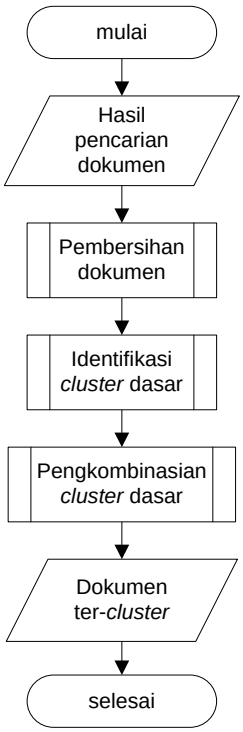
lokasi, dinyatakan dengan bidang dua dimensi, dari pelanggan suatu toko dapat dikelompokkan menjadi beberapa *cluster* dengan pusat *cluster* ditunjukkan oleh tanda positif (+). Banyak algoritma *clustering* memerlukan fungsi jarak untuk mengukur kemiripan antar data, diperlukan juga metoda untuk normalisasi bermacam atribut yang dimiliki data.



Gambar II-3 Contoh clustering

II.2 *Suffix Tree Clustering*

Algoritma STC memiliki tiga proses utama, yaitu d*ocument preprocessing*, identifikasi cluster dasar pembangun menggunakan *suffix tree,* dan pengkombinasian cluster-cluster dasar ke dalam cluster yang lebih umum.



Gambar II-4 Langkah-langkah dalam algoritma STC

Penjelasan untuk langkah-langkah dalam algoritma STC adalah sebagai berikut :

II.2.1 Document *preprocessing*

Pembersihan dokumen merupakan langkah awal dalam algoritma STC. Dalam proses ini dokumen yang didapat berdasarkan pencarian suatu kata kunci dibersihkan dari karakter-karakter yang tidak diperlukan, proses penghapusan *stop-word*, dan proses *stemming*. Penghapusan karakter-karakter yang tidak diperlukan meliputi pembersihan tanda, penghapusan *stop-word* meliputi penghapusan kata-kata umum yang tidak mengacu kepada suatu hal, seperti kata depan, kata gabung, dan sebagainya. Sedangkan proses *stemming* adalah proses pengembalian suatu kata ke dalam bentuk awalnya, misalnya penghilangan imbuhan.

II.2.1.1 Pembersihan *stop word*

Stop word adalah kata-kata yang muncul dalam suatu pencarian, namun bukan merupakan kata-kata yang menjadi inti dari hasil pencarian tersebut. *Stop word* dapat berbeda satu sama lain, misalnya *stop word* untuk menyeleksi kata-kata yang muncul dalam *e-mail* akan berbeda dengan *stop word* yang digunakan untuk menyeleksi dokumen secara umum.

II.2.1.2 Stemming kata

Stemming adalah pengubahan suatu kata ke dalam bentuk dasarnya. Umumnya algoritma ini dibuat untuk kata-kata dalam bahasa Inggris. Namun beberapa algoritma *stemming* saat ini telah dibuat oleh para peneliti untuk mencakup beberapa bahasa lain di dunia selain bahasa Inggris. Contoh algoritma *stemming* untuk selain bahasa Inggris adalah algoritma Nazief dan Andriani, yang dikembangkan khusus untuk bahasa Indonesia.

II.2.2 Identifikasi Cluster Dasar Pembangun

Tahap kedua dari algoritma STC adalah tahap identifikasi pembentukan *cluster* dasar. Pembentukan *cluster* dasar dilakukan dengan cara menemukan kesamaan frasa-frasa yang ditemukan dalam dokumen-dokumen yang diteliti dengan menggunakan struktur data *suffix tree*. Dengan cara ini maka setiap dokumen dapat direpresentasikan menjadi suatu kalimat. Setiap *cluster* dasar yang dibentuk memiliki suatu *score*. Penghitungan *score* tersebut berdasarkan jumlah dokumen yang masuk ke dalam anggota *cluster* dasar dan jumlah kata yang menyusun frasa dari *cluster* dasar. Fungsi untuk menghitung *score cluster* dasar ditunjukkan oleh persamaan di bawah ini.

$$S(B) = |B|.f(|P|) \quad (2.1)$$

Dimana

S(B) = *score cluster*,

|B| = jumlah dokumen yang memiliki frasa yang muncul di dalam *cluster* dasar, dan

f(|P|) = jumlah kata yang menyusun frasa tersebut.

Tidak semua cluster yang didapat akan diberikan *score*, *cluster* yang diberi *score* dan ditampilkan hanyalah *cluster* yang memiliki jumlah kemunculan kata yang lebih banyak daripada tiga. Nilai tiga dipilih karena jika *cluster* memiliki jumlah *score* yang lebih sedikit maka akan dianggap sebagai *outlier* yang tidak mencerminkan *cluster* dasar secara umum.

II.2.3 Pengkombinasian *Cluster* Dasar

Tahap ini dilakukan unutk menangani *cluster-cluster* yang *overlap*. Pengkombinasian dapat dilakukan dengan menghitung nilai *similarity* antar *cluster* dasar yang didasarkan pada jumlah dokumen yang *overlap*. *Overlapping* dokumen didasarkan karena dokumen dapat memiliki lebih dari satu topik.

Penghitungan nilai *similarity* menggunakan nilai biner. Rumus untuk menghitung nilai *similarity* antar *cluster* dasar dapat dilihat pada persamaan di bawah ini.

$$\frac{|Bm \cap Bn|}{|Bm|} > 0{,}5 \quad (2.2)$$

$$\frac{|Bm \cap Bn|}{|Bn|} > 0{,}5 \quad (2.3)$$

Dimana :

|Bm ∩ Bn| = jumlah dokumen yang mengandung kedua kata <Bm> dan <Bn>.

|Bm| = jumlah dokumen yang mengandung kata <Bm>.

|Bn| = jumlah dokumen yang mengandung kata <Bn>.

Dalam penggunaan di atas, penggunaan nilai ambang 0,5 digunakan karena nilai tersebut merupakan nilai tengah antara 0 sampai 1. Jika nilai dari kedua persamaan di atas bernilai benar (lebih dari 0,5) maka *similarity* antar dua *cluster* dasar akan bernilai satu sehingga kedua *cluster* tersebut akan terhubung. Jika hanya salah satu yang bernilai benar atau keduanya salah maka nilai *similarity* akan dianggap bernilai 0 dan kedua *cluster* dasar tersebut tidak terhubung.

II.3 Jejaring Sosial

Jejaring sosial adalah struktur sosial yang terdiri dari elemen-elemen individual atau organisasi. Jejaring ini menunjukan jalan dimana mereka berhubungan karena kesamaan sosialitas, mulai dari mereka yang dikenal sehari-hari sampai dengan keluarga. Istilah ini diperkenalkan oleh profesor J.A. Barnes di tahun 1954.

Jejaring sosial adalah suatu struktur sosial yang dibentuk dari simpul-simpul (yang umumnya adalah individu atau organisasi) yang diikat dengan satu atau lebih tipe relasi spesifik seperti nilai, visi, ide, teman, keturunan, dan lain-lain.

Sesuai dengan perkembangan teknologi, banyak fasilitas dan media baru penyedia jejaring sosial seperti Digg, Friendster, Facebook, Twitter dan lain-lain yang bertujuan untuk menjalin jaringan pertemanan tanpa dibatasi oleh tempat

dan ruang secara fisik. Dari sekian banyak penyedia jejaring sosial, ada sebagian menghilang begitu saja, adapula yang sukses, tergantung dari pengelolaannya baik secara teknikal maupun manajemen (pemasaran/promosi, keuangan, legalitas, *business intelegent*).

II.4 Mesin Pencari

Mesin pencari adalah program komputer yang dirancang untuk melakukan pencarian atas berkas-berkas yang tersimpan dalam layanan *www*, ftp, publikasi milis, ataupun *news group* dalam sebuah ataupun sejumlah komputer peladen dalam suatu jaringan. Hasil pencarian umumnya ditampilkan dalam bentuk daftar yang seringkali diurutkan menurut tingkat akurasi ataupun rasio pengunjung atas suatu berkas yang disebut sebagai hits. Informasi yang menjadi target pencarian bisa terdapat dalam berbagai macam jenis berkas seperti halaman situs web, gambar, ataupun jenis-jenis berkas lainnya. Beberapa mesin pencari juga diketahui melakukan pengumpulan informasi atas data yang tersimpan dalam suatu basis data ataupun direktori web.

Sebagian besar mesin pencari dijalankan oleh perusahaan swasta yang menggunakan algoritma kepemilikan dan basisdata tertutup, di antaranya yang paling populer adalah Google (MSN Search dan Yahoo!). Telah ada beberapa upaya menciptakan mesin pencari dengan sumber terbuka (*open source*), contohnya adalah Htdig, Nutch, Egothor dan OpenFTS.

II.4.1 Google

Google muncul di akhir tahun 1997, dimana Google memasuki pasar yang telah diisi oleh para pesaing lain dalam penyediaan layanan mesin pencari, seperti Yahoo, Altavista, HotBot, Excite, InfoSeek dan Lycos, dimana perusahaan-perusahaan tersebut mengklaim sebagian perusahaan yang bergerak dalam bidang layanan pencarian di internet, hingga akhirnya Google mampu menjadi sebagai penyedia mesin pencari yang cukup diperhitungkan di dunia.

Tingginya persaingan antar mesin pencari yang ada tidak mampu menghentikan kesuksesan Google. Setelah Yahoo mampu pada posisi puncak di sekitar tahun 2000, Google mampu menerobos liga besar tersebut. sehingga Google dipandang sebagai mesin pencari yang utama seperti yang kita ketahui pada hari ini.

II.4.2 Manfaat Mesin Pencari

Adapun manfaat dari mesin pencari antara lain adalah sebagai berikut :

1. Mesin pencari merupakan tempat kebanyakan orang mencari sesuatu via internet. Menurut survei hampir 90% pengguna internet memakai mesin pencari untuk mencari lokasi tertentu di internet, dan di antara mesin pencari yang ada, Google merupakan mesin pencari yang paling banyak digunakan.
2. Sebagian besar pengguna mesin pencari tidak pernah melewatkan dua halaman pertama dari mesin pencari.
3. Sebagian besar (hampir 70%) pengguna mesin pencari tidak pernah klik pada hasil pencarian sponsor. Dengan demikian, hasil pencarian yang organik

(secara alami) akan membuat suatu website memperoleh posisi strategis dalam dunia internet.

4. Di negara-negara maju, porsi penjualan yang dilakukan melalui internet sudah hampir mencapai 20% dari keseluruhan transaksi tahunan.
5. Sebuah informasi yang mudah di akses oleh semua orang baik dalam maupun luar negeri.

II.4.3 Cara Kerja Mesin Pencari

Mesin pencari web bekerja dengan cara menyimpan informasi tentang banyak halaman web. Halaman-halaman ini diambil dengan *web crawler* - *browser* yang secara otomatis yang mengikuti setiap pranala yang dilihatnya. Isi setiap halaman lalu dianalisis untuk menentukan indeksnya (misalnya, kata-kata diambil dari judul, subjudul, atau *field* khusus yang disebut meta tag). Data tentang halaman web disimpan dalam sebuah *database* indeks untuk digunakan dalam pencarian selanjutnya. Sebagian mesin pencari menyimpan seluruh atau sebagian halaman sumber (yang disebut cache) maupun informasi tentang halaman web itu sendiri.

Selain halaman web, Mesin pencari juga menyimpan dan memberikan informasi hasil pencarian berupa pranala yang merujuk pada file, seperti file audio, file video, gambar, foto dan sebagainya, serta informasi tentang seseorang, suatu produk, layanan, dan informasi beragam lainnya yang semakin terus berkembang sesuai dengan perkembangan teknologi informasi.

Ketika seseorang mengunjungi mesin pencari dan memasukkan *query*, biasanya dengan memasukkan kata kunci, mesin mencari indeks dan memberikan

daftar halaman web yang paling sesuai dengan kriterianya, biasanya disertai ringkasan singkat mengenai judul dokumen dan kadang-kadang sebagian teksnya.

Ada jenis mesin pencari lain: mesin pencari *real-time*, seperti Orase. Mesin seperti ini tidak menggunakan indeks. Informasi yang diperlukan mesin tersebut hanya dikumpulkan jika ada pencarian baru. Jika dibandingkan dengan sistem berbasis indeks yang digunakan mesin-mesin seperti Google, sistem *real-time* ini unggul dalam beberapa hal: informasi selalu mutakhir, (hampir) tak ada pranala mati, dan lebih sedikit sumber daya sistem yang diperlukan. (Google menggunakan hampir 100.000 komputer, Orase hanya satu.) Tetapi, ada juga kelemahannya, yaitu pencarian lebih lama rampungnya.

Manfaat mesin pencari bergantung pada relevansi hasil-hasil yang diberikannya. Meskipun mungkin ada jutaan halaman web yang mengandung suatu kata atau frasa, sebagian halaman mungkin lebih relevan, populer, atau autoritatif daripada yang lain. Kebanyakan mesin pencari menggunakan berbagai metode untuk menentukan peringkat hasil pencarian agar mampu memberikan hasil "terbaik" lebih dahulu. Cara mesin menentukan halaman mana yang paling sesuai, dan urutan halaman-halaman itu diperlihatkan, sangat bervariasi. Metode-metode yang digunakan pun berubah seiring waktu dengan berubahnya penggunaan internet dan berevolusinya teknik-teknik baru.

Sebagian besar mesin pencari web adalah usaha komersial yang didukung pemasukan iklan dan karenanya sebagian menjalankan praktik kontroversial, yaitu membolehkan pengiklan membayar agar halaman mereka diberi peringkat lebih tinggi dalam hasil pencarian.

II.4.4 Prinsip Umum Mesin Pencari

Sistem kinerja mesin ini ada beberapa hal yang perlu di perhatikan terutama keterkaitannya dengan masalah arsitekrut dan mekanismenya.

II.4.4.1 *Spider*

Spider merupakan program yang men-*download* halaman-halaman yang mereka temukan, mirip dengan *browser*. Perbedannya adalah bahwa *browser* menampilkan secara langsung informasi yang ada (baik tekas, gambar, dll). Untuk kepentingan manusia yang menggunakannya pada saat itu, sedangkan *spider* tidak melakukan untuk menampulkan dalam bentuk yang terlihat seperti itu, karena kepentingannya adalah untuk mesin, bukan untuk manusia, spider pun dijalankan oleh mesin secara otomatis. Kepentingannya adalah untuk mengambil halaman-halaman yang dikunjunginya untuk disimpan kedalam *database* yang dimiliki oleh *search engine*.

II.4.4.2 *Crawler*

Crawler merupakan program yang dimiliki *search engine* untuk melacak dan menemukan link yang terdapat dari setiap halaman yang ditemuinya. Tugasnya adalah untuk menentukan *spider* harus pergi kemana dan mengevaluasi link berdasarkan alamat yang ditentukan dari awal. *Crawler* mengikuti link dan mencoba menemukan dokumen yang belum dikenal oleh *search engine*.

II.4.4.3 *Indexer*

Komponen ini melakukan aktifitas untuk menguraikan masing-masing halaman dan meneliti berbagai unsur, seperti teks, headers, struktur atau fitur dari gaya penulisan, tag HTML khusus, dan sebagainya.

II.4.4.4 *Database*

Merupakan tempat standar untuk menyimpan data-data dari halaman yang telah dikunjungi, di-*download* dan sudah dianalisis. kadang kala disebut juga dengan *index* dari suatu *search engine*.

II.4.4.5 *Result Engine*

Mesin yang melakukan penggolongan dan penentuan peringkat dari hasil pencarian pada *search engine*. Mesin ini menentukan halaman mana yang menemui kriteria terbaik dari hasil pencarian berdasarkan permintaan penggunanya, dan bagaimana bentuk penampilan yang akan ditampilkan.

Proses ini dilaksanakan berdasarkan algoritma pe-*ranking*-an yang dimiliki oleh *search engine* tersebut, mengikuti kaidah pe-*rangking*-an halaman yang dipergunakan oleh mereka adalah hak mereka, para peneliti mempelajari sifat-sifat yang mereka gunakan, terutama untuk meningkatkan pencarian yang dihasilkan oleh *search engine* tersebut.

II.4.4.6 *Web Server*

Web server merupakan komponen yang melayani permintaan dan memberikan respon balik dari permintaan tersebut. *Web Server* ini biasanya menghasilkan informasi atau dokumen dalam format HTML. Pada halaman tersebut tersedia layanan untuk mengisikan kata kunci pencarian yang diinginkan oleh penggunanya. *Web Server* ini juga bertanggung jawab dalam menyampaikan hasil pencarian yang dikirimkan kepada komputer yang meminta informasi.

II.5 *Application Programming Interface* (API)

API adalah serangkaian instruksi dan standar pemrograman untuk mengakses aplikasi atau layanan berbasis *web*. Sebuah perusahaan *software* atau penyedia layanan berbasis *web* seringkali merilis API mereka kepada publik. Dengannya, pengembang lain dapat mendesain aplikasi yang memanfaatkan layanan mereka. Sebagai contoh, Amazon.com merilis API sehingga para pengembang *web* dapat lebih mudah mengakses informasi produk-produk Amazon dari *website* mereka. Menggunakan API dari Amazon, *website* pihak ketiga dapat mem-*posting link* langsung ke produk-produk Amazon dengan harga aktual dan opsi "buy now".

API adalah *software-to-software interface*, bukan *user interface*. Dengan API, aplikasi-aplikasi saling berkomunikasi tanpa ada intervensi dari pengguna. Misalnya ketika Anda memasukkan nomor kartu kredit untuk berbelanja, toko *online* tersebut akan menggunakan API untuk mengirimkan informasi kartu kredit kepada aplikasi dari perusahaan lain yang memverifikasi kartu kredit. Bila aplikasi verifikasi kartu kredit menyatakan bahwa informasi yang Anda masukkan benar, aplikasi tersebut akan mengirimkan respons kembali ke toko *online* tempat Anda berbelanja, dan menyatakan bahwa proses pembayaran bisa dilakukan. Proses komunikasi antara aplikasi toko *online* dengan aplikasi verifikasi kartu kredit sepenuhnya dilakukan tanpa campur tangan manusia. Anda hanya melihat satu *interface*: toko *online*. Tapi di belakang layar, terdapat beberapa aplikasi yang bekerja sama menggunakan API. Jenis integrasi ini dikenal dengan istilah *seamless*, karena pengguna tidak menyadari ketika informasi yang dimasukkannya dialihkan dari satu aplikasi ke aplikasi lain.

API serupa dengan konsep *Software as a Service* (SaaS), karena pengembang aplikasi tidak perlu memulai segala sesuatu dari kertas kosong (*from scratch*) setiap kali mereka menulis sebuah program. Alih-alih membangun aplikasi inti yang menangani semua pekerjaan (semisal *e-mail*, *billing*, *tracking*, dll) sendiri, pekerjaan-pekerjaan tersebut dapat diserahkan kepada penyedia layanan atau perusahaan *software* lain.

API memungkinkan sebuah aplikasi berkomunikasi dengan aplikasi lain di Internet melalui serangkaian panggilan (*call*). Sebuah API, berdasarkan definisinya, adalah sesuatu yang mendefinisikan cara dua entitas untuk berkomunikasi. Entitas di sini adalah sebuah *software* yang nyata berbeda (dalam layanan) dengan *software* lain. Dengan API, panggilan-panggilan yang bolak-balik antar aplikasi diatur melalui *web service*. *Web service* adalah kumpulan standar teknis dan protokol, termasuk XML (*Extensible Markup Language*), bahasa umum yang digunakan oleh aplikasi-aplikasi tersebut selama berkomunikasi di Internet.

API sendiri merupakan sekumpulan kode software yang ditulis sebagai serangkaian pesan XML. Setiap pesan XML berhubungan dengan fungsi spesifik dari aplikasi yang akan diajak berkomunikasi. Sebagai contoh, pada API Facebook, terdapat pesan XML yang berhubungan dengan fungsi spesifik *wall post*, *wall comment*, *wall like*. Pengembang aplikasi pihak ketiga menggunakan pesan-pesan XML yang berhubungan dengan fungsi-fungsi spesifik dari layanan *web* yang akan diajak berkomunikasi. Pengembang bebas memilih fungsi khusus apa saja yang akan diajak berkomunikasi, dan ditampilkan pada aplikasi

rancangannya. Sebagai contoh, Anda bisa membuat Facebook *client* yang hanya menampilkan *update* status teman-teman Anda.

API dan *web service* sepenuhnya bekerja di belakang layar. Para peselancar *web* dan pengguna software sama sekali tidak melihatnya. Keduanya bekerja diam-diam, menyediakan jalan bagi beberapa aplikasi untuk bekerja sama untuk memberikan informasi atau fungsionalitas yang dibutuhkan oleh pengguna aplikasi. Dengan demikian, API adalah standar komunikasi yang dibuka oleh perusahaan *software*, agar dapat dimanfaatkan oleh pengembang pihak ketiga untuk mendesain aplikasi yang memanfaatkan layanan mereka dengan mudah [15].

II.6 *Data Flow Diagram* (DFD)

Data Flow Diagram (DFD) adalah alat pembuatan model yang memungkinkan profesional untuk menggambarkan sistem sebagai suatu jaringan proses fungsional yang dihubungkan satu sama lain dengan alur data, baik secara manual maupun komputerisasi. DFD ini sering disebut juga dengan *Bubble Chart*, *Bubble Diagram*, model proses, diagram alur kerja, atau model fungsi.

DFD ini adalah suatu alat pembuatan model yang sering digunakan, khususnya bila fungsi-fungsi sistem merupakan bagian yang lebih penting dan kompleks daripada data yang dimanupalsi oleh sistem. Dengan kata lain, DFD adalah alat pembuatan model yang memberikan penekanan hanya pada fungsi sistem.

DFD ini merupakan alat perancangan sistem yang berorientasi pada alur data dengan konsep dekomposisi dapat digunakan untuk penggambaran analisa maupun rancangan sistem yang mudah dikombinasikan oleh profesional sistem kepada pemakai maupun pembuat program.

II.6.1 Komponen *Data Flow Diagram* (DFD)

Data Flow Diagram memiliki beberapa komponen utama sebagai berikut :

II.6.1.1 Komponen *Terminator* atau Entitas Luar

Terminator mewakili entitas eksternal yang berkomunikasi dengan sistem yang sedang dikembangkan. Biasanya *terminator* dikenal dengan nama entitas luar (*external entity*).

Terdapat dua jenis *terminator* yaitu :

1. *Terminator* Sumber (*source*), merupakan terminator yang menjadi sumber
2. *Terminator* Tujuan (*sink*), merupakan terminator yang menjadi tujuan data atau informasi sistem.

II.6.1.2 Komponen Proses

Komponen proses menggambarkan bagian dari sistem yang mentransformasikan input menjadi output. Proses dilambangkan dengan lingkaran. Nama proses dituliskan dengan satu kata, singkatan atau kalimat sederhana.

II.6.1.3 Komponen *Data Store*

Komponen ini digunakan untuk membuat model sekumpulan paket data dan diberi nama dengan kata benda bersifat jamak. *Data store* dapat berupa

file/*database* yang tersimpan dalam *harddisk* atau bersifat manual seperti buku alamat.

II.6.1.4 Komponen Alur Data

Alur data digunakan untuk menerangkan perpindahan data atau paket data dari satu bagian ke bagian lainnya. Alur data dapat berupa kata, pesan, atau informasi. Ada empat konsep tentang alur data yaitu :

1. *Packets of data*

Apabila ada dua data atau lebih yang mengalir dari satu sumber yang sama menuju pada tujuan yang sama dan mempunyai hubungan, maka digambarkan dengan satu alur data.

2. *Diverging data flow*

Apabila ada sejumlah paket data yang berasal dari sumber yang sama menuju pada tujuan yang berbeda atau paket data yang kompleks dibagi menjadi beberapa elemen data yang dikirim ke tujuan berbeda.

3. *Converging data flow*

Apabila ada beberapa alur data yang berbeda sumber menuju ke tujuan yang sama.

4. Sumber dan tujuan

Semua alur data harus minimal mengandung satu proses.

II.6.1.5 Bentuk *Data Flow Diagram* (DFD)

Terdapat dua bentuk DFD, yaitu Diagram Alur Data Fisik dan Diagram Alur Data Logika. Diagram alur data fisik lebih menekankan pada bagaimana proses

dari sistem diterapkan, sedangkan diagram alur data logika lebih menekankan proses-proses apa yang terdapat di dalam sistem.

1. Diagram Alur Data Fisik (DADF)

Diagram alur data fisik lebih tepat digunakan untuk menggambarkan sistem yang ada (sistem yang lama). Penekanan dari diagram alur data fisik adalah bagaimana proses-proses dari sistem yang diterapkan (dengan cara apa, oleh siapa dan dimana), termasuk proses-proses manual.

2. Diagram Alur Data Logika (DADL)

Diagram alur data logika lebih tepat digunakan untuk menggambarkan sistem yang akan diusulkan (sistem yang baru). untuk sistem komputerisasi, penggambaran diagram alur data logika hanya menunjukkan kebutuhan proses dari sistem yang diusulkan secara logika, biasanya proses-proses yang digambarkan hanya merupakan proses-proses secara komputer saja.

II.6.1.6 Pembuatan DFD

Tidak ada aturan baku untuk membuat DFD, tetapi dari berbagai referensi yang ada adalah sebagai berikut :

1. *Diagram Context*

Diagram ini adalah diagram level tertinggi dari DFD yang menggambarkan keterkaitan aliran-aliran data antar sistem dengan bagian luar (kesatuan luar). Kesatuan luar ini merupakan sumber arus data atau tujuan data yang berhubungan dengan sistem informasi tersebut.

2. Diagram Level Nol

Diagram ini adalah dekomposisi dari *diagram context*.

3. Diagram Level Satu

Diagram ini merupakan dekomposisi dari diagram level nol.

II.7 *Entity Relationship Diagram*

Entity Relationship Diagram (ERD) merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data, berdasarkan objek-objek dasar data yang mempunyai hubungan antar relasi.

ERD digunakan untuk memodelkan struktur data dan hubungan antar data, dan untuk menggambarkannya digunakan beberapa notasi dan simbol. Pada dasarnya ada tiga simbol yang digunakan yaitu :

II.7.1 Entitas (*Entity*)

Entitas merupakan objek yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain. Simbol dari entitas ini biasanya digambarkan dengan persegi panjang. Entitas memiliki dua tipe yaitu :

a. Entitas Kuat

Entitas yang mempunyai atribut kunci. Entitas ini bersifat mandiri, keberadaannya tidak bergantung pada entitas lainnya. Kebanyakan entitas dalam suatu organisasi dapat digolongkan sebagai entitas kuat (*strong entity*). Entitas kuat memiliki karakteristik yang unik (dinamakan *identifier*), yaitu sebuah atribut tunggal atau gabungan atribut yang secara unik dapat digunakan untuk membedakannya dari entitas kuat yang lain.

b. Entitas Lemah

Entitas yang tidak mempunyai atribut kunci. Entitas lemah diidentifikasikan dengan menghubungkan entitas tertentu dari tipe entitas yang lain ditambah atribut dari entitas lemah. Tipe entitas lain yang dipakai untuk mengidentifikasikan suatu entitas lemah disebut *identifying owner* dan *identifying relationship*.

c. Atribut

Setiap entitas pasti mempunyai elemen yang disebut atribut yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Isi dari atribut mempunyai sesuatu yang dapat mengidentifikasi isi elemen satu dengan yang lain.

Atribut memiliki beberapa tipe yaitu :

a. Atribut Sederhana (*Simple Attribute*)

Merupakan atribut atomik yang tidak dapat dipilah lagi.

b. Atribut Komposit (*Composite Attribute*)

Merupakan atribut yang masih dapat diuraikan lagi menjadi subsub atribut yang masing-masing memiliki makna

c. Atribut Bernilai Banyak (*Multivalued Attribute*)

Ditunjukkan pada atribut-atribut yang dapat kita isi dengan lebih dari 1 nilai, tetapi jenisnya sama. Contohnya nomor telepon, hobi, dan lain-lain.

d. Atribut Bernilai Tunggal (*Single-Valued Attribute*)

Ditujukkan pada atribut-atribut yang memiliki paling banyak satu nilai untuk setiap baris data.

e. Atribut Turunan

Adalah atribut yang nilai-nilainya diperoleh dari pengolahan atau dapat diturunkan dari atribut atau tabel lain yang berhubungan. Contoh atribut umur yang dapat dikalkulasi dari atribut tanggal lahir.

f. Atribut Harus Bernilai (*Mandatory Attribute*)

Adalah atribut yang harus berisi nilai. Contoh pada tabel mahasiswa, nomor induk mahasiswa dan nama mahasiswa harus diketahui.

g. Atribut Tidak Harus Bernilai (*Non Mandatory Attribute*)

Adalah atribut yang nilainya boleh dikosongkan.

d. Hubungan atau Relasi

Hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda.

Relasi yang terjadi diantara dua himpunan entitas (misalnya A dan B) dalam satu basis data yaitu :

1. Satu ke satu (*One to One*)

Hubungan relasi satu ke satu yaitu setiap entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas pada himpunan entitas B.

2. Satu ke banyak (*One to Many*)

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi setiap entitas pada entitas B hanya dapat berhubungan dengan satu entitas pada himpunan entitas A.

3. Banyak ke satu (*Many to One*)

Setiap entitas A hanya dapat mempunyai satu hubungan dengan himpunan entitas B.

4. Banyak ke banyak (*Many to Many*)

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B.

II.8 Aplikasi Pembangun

Aplikasi pembangun yang digunakan dalam pembangunan perangkat lunak ini adalah sebagai berikut :

II.8.1 PHP

PHP adalah bahasa pemrograman *script* yang paling banyak dipakai saat ini. PHP banyak dipakai untuk memprogram situs *web* dinamis, walaupun tidak tertutup kemungkinan digunakan untuk pemakaian lain.

II.8.1.1 Kelebihan PHP

Kelebihan PHP bila dibandingkan dengan program bahasa lainnya adalah sebagai berikut :

1. Bahasa pemrograman PHP adalah sebuah *script* yang tidak melakukan sebuah kompilasi dalam penggunaannya.
2. *Web server* yang mendukung PHP dapat ditemukan dimana-mana dari mulai *Apache*, IIS, *Lighttpd*, hingga *Xitami* dengan konfigurasi yang relatif mudah.
3. Dalam sisi pengembangan lebih mudah dan *developer* yang siap membantu dalam pengembangan.

4. Dalam sisi pemahaman, PHP adalah bahasa *scripting* yang paling mudah karena memiliki referensi yang banyak.

PHP adalah bahasa *open source* yang dapat digunakan di berbagai mesin (*Linux, Unix, Macintosh, Windows*) dan dapat dijalankan secara *runtime* melalui *console* serta juga dapat menjalankan perintah-perintah sistem.