BAB 2

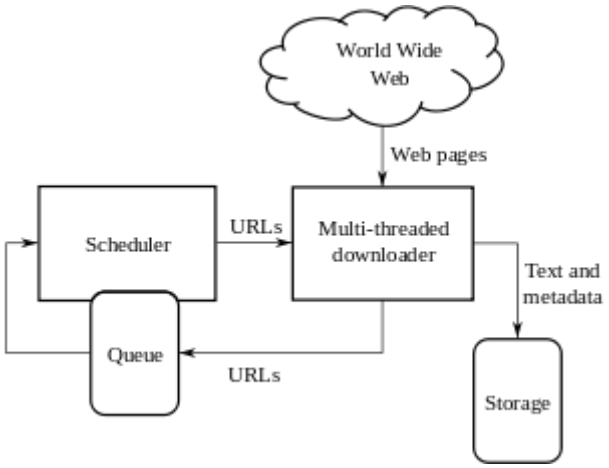
LANDASAN TEORI

2.1. *Web Crawler*

Web Crawler adalah meng-*crawl* (merayapi) seluruh informasi suatu *website* yang biasanya digunakan untuk meng-*index* suatu *website*, pemeliharaan *website*, atau digunakan untuk memperoleh data khusus contohnya *email*. Dan hal ini juga dapat digunakan untuk memvalidasi *hyperlink* dan kode HTML.

Web Crawler dimulai dengan me-*list* daftar URL yang akan dikunjungi, yang disebut dengan *seed*. *Web crawler* akan mengunjungi URL yang ada di daftar dan mengidentifikasi semua *hyperlink* di halaman tersebut serta menambahkannya kedalam daftar URL yang akan dikunjungi yang disebut *crawl frontier*. URL yang telah ada dikunjungi dan diambil informasi yang ada sesuai yang dibutuhkan.

Dengan banyaknya jumlah URL yang mungkin di-*crawl* oleh *crawler server* yang membuatnya sulit untuk menghindari pengambilan konten yang sama. Misalkan protokol HTTP GET membuat kombinasi URL yang sangat banyakdan sedikit dari URL tersebut menghasilkan konten yang berbeda dan selebihnya menghasilkan konten yang sama untuk URL yang berbeda, inilah yang menimbulkan masalah bagi *crawler* agar bisa mengambil konten yang berbeda dari URL-URL tersebut. (Wikipedia, 2016)



Gambar 2.1 Arsitektur *Web Crawler* (Wikipedia, 2016)

Salah satu jenis dari *web crawler* adalah *focused crawler* yang merupakan *web crawler* yang mengambil data dengan spesifikasi tertentu, misalkan dengan topik 'kesehatan', maka crawler hanya akan mengambil halaman web yang hanya berhubungan dengan topik kesehatan. Algoritma ini mencari kesamaan dari halaman yang sedang di-*crawl* dengan *query* yang diberikan. Pada pendekatan ini, *web crawler* akan men-*download* halaman *web* yang mirip dengan halaman yang lainnya, yang dapat dibantu dengan menggunakan Naive Bayes Classifier untuk mengklasifikasikan apakah halaman yang sedang dikunjungi mempunyai kemiripan dengan *query* yang diberikan. (Janbandhu, et al., 2014)

2.2. *Multithreading*

Multithreading adalah kemampuan CPU *single-core* ataupun *multi-core* dalam menjalankan beberapa proses secara bersamaan. Berbeda dengan *multiprocessing*, dimana *thread* membagi sumber daya seperti unit komputer, *CPU caches*, maupun *Translation Lookaside Buffer* (TLB).

Jika *thread* melewatkan *cache*. *Thread* yang lain dapat memakai sumber daya yang tidak terpakai, yang dapat mempercepat eksekusi secara keseluruhan. Jika sebuah *thread* tidak dapat memakai semua sumber daya yang ada, dengan menjalankan *thread* yang lain dapat mencegah sumber daya lain menjadi *'idle'*. Jika beberapa *thread* bekerja pada data yang sama, mereka dapat membagi tugas mereka masing-masing yang meningkatkan penggunaan *cache* (Wikipedia, 2010).

2.3. *Boilerpip*e

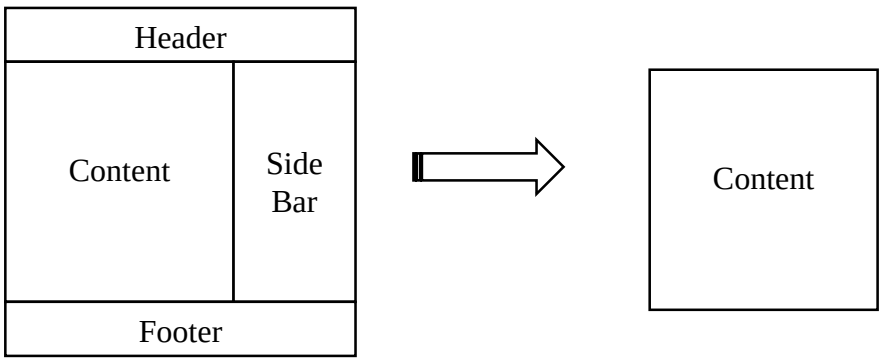
Boilerpipe merupakan salah satu algoritma ekstraksi konten pada halawan HTML yang menyediakan algoritma untuk mendeteksi dan menghapus konten-konten selain *main content* seperti *boilerplate* ataupun *template*. *Boilerplate* merupakan *web template* yang sering digunakan untuk membangun proyek web baru. Dimana hal ini dapat mempermudah *web developer* untuk membangun website. Salah satu cara yang digunakan adalah dengan menghitung jumlah kata pada setiap blok halaman web. Apabila blok tersebut mempunyai jumlah kata terbanyak, maka dapat dikatakan blok tersebut merupakan konten utama. Sedangkan blok yang mempunyai sedikit kata, maka dapat dikatakan bahawa blok tersebut merupakan boilerplate (Kohlschütter, et al., 2010).

Terdapat beberapa teknik *extraction* yang dapat digunakan sesuai dengan jenis halaman yang akan di-extract yaitu:

Tabel 2.1 Jenis *Boilerpipe Extraction*

Teknik *Extraction*	**Deskripsi**
ArticleExtractor	Meng-*extract* konten artikel berita.
DefaultExtractor	Meng-*extract* konten web biasa.
LargestContentExtractor	Mirip dengan *DefaultExtractor*, tetapi tetap mengambil konten yang paling besar. Cocok digunakan untuk web yang bukan artikel.
KeepEverythingExtractor	Mengambil semua dalam web sebagai konten.

Format output yang dapat digunakan adalah html, htmlFragment, text, json dan debug. (Kohlschütter, 2016)



Gambar 2.2 Ilustrasi *Boilerpipe* (Treselle System, 2014)

Pada penelitian ini jenis ekstraksi yang dipakai adalah *ArticleExtractor,* dikarenakan sumber data yang diambil bersumber dari artikel.

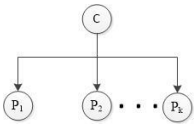
2.4. *Text Prepocessing*

Text Preprocessing merupakan tahapan awal dari text mining yang bertujuan mempersiapkan teks menjadi data yang akan mengalami pengolahan pada tahap selanjutnya. Pada text mining, data mentah yang berisi informasi memiliki struktur yang sembarang, sehingga diperlukan proses pengubahan bentuk menjadi data yang terstruktur sesuai kebutuhan, yaitu biasanya akan mejadi nilai-nilai numerik. Proses ini disebut *Text Preprocessing* (Triawati, 2009).

Pada tahap ini, tindakan yang dilakukan adalah *tokenizing* yaitu proses penguraian deskripsi yang semula berupa kalimat mejadi kata-kata kemudian menghilangkan *delimiter-delimiter* seperti tanda koma (,), tanda titik (.), spasi dan karakter angka yang terdapat pada kata tersebut, *Case Folding* yaitu penyamaan *case* dalam sebuah dokumen dan *filtering* yaitu proses pembuangan kata-kata umum sehingga hanya akan tersisa kata-kata yang khusus (Weiss, et al., 2005).

2.5. *Naïve Bayes Classifier*

Klasifikasi Bayes adalah klasifikasi statistik yang dapat memprediksi kelas suatu anggota probabilitas. Untuk klasifikasi Bayes sederhana yang lebih dikenal sebagai Naïve Bayesian Classifier dapat diasumsikan bahwa efek dari suatu nilai atribut sebuah kelas yang diberikan adalah bebas dari atribut-atribut lain. Asumsi ini disebut *class conditional independence* yang dibuat untuk memudahkan perhitungan-perhitungan pengertian ini dianggap "naive", dalam bahasa lebih sederhana naïve itu mengasumsikan bahwa kemunculan suatu *term* kata dalam suatu kalimat tidak dipengaruhi kemungkinan kata-kata yang lain dalam kalimat padahal dalam kenyataanya bahwa kemungkinan kata dalam kalimat sangat dipengaruhi kemungkinan keberadaan kata-kata yang dalam kalimat. Dalam Naïve Bayes di asumsikan prediksi atribut adalah tidak tergantung pada kelas atau tidak dipengaruhi atribut laten



Gambar 2.3 Klasifikasi *Naïve Bayes* sebagai jaringan bayes dengan atribut prediksi (P_1, P_2, …….P_k) dan kelas (C)

C adalah anggota kelas dan X adalah variabel acak sebuah vektor sebagai atribut nilai yang diamati. c mewakili nilai label kelas dan x mewakili nilai atribut vector yang diamati. Jika diberikan sejumlah x tes untuk klasifikasi maka probablitas tiap kelas untuk atribut prediksi vektor yang diamati adalah

$$p(C = c|X = x) = \frac{p(C = c)p(X = x|C = c)}{p(C = c)}$$

Model multinomial mengambil jumlah kata yang muncul pada sebuah dokumen, dalam model multinomial sebuah dokumen terdiri dari beberapa kejadian kata dan di asumsikan panjang dokumen tidak bergantung pada kelasnya. Dengan menggunakan asumsi Bayes yang sama bahwa kemungkinan tiap kejadian kata dalam sebuah dokumen adalah bebas tidak terpengaruh dengan konteks kata dan posisi kata dalam dokumen. Tiap dokumen d_i di gambarkan sebagai distribusi multinomial kata, N_{it} dihitung dari jumlah kemunculan kata w_t yang terjadi dalam dokumen d_i . Maka kemungkinan sebuah dokumen diberikan sebuah kelas adalah (McCallum & Nigam, 1998)

$$P(d_i|c_j;\theta) = P(|d_i|)\ |d_i|!\prod_{t=1}^{|v|}\frac{P(w_t|c_j;\theta)^{N_{it}}}{N_{it}!}$$

Kemungkinan untuk tiap kata dapat ditulis $\theta_{w_t|c_j} = P(w_t|c_j;\theta)$ dimana $0 \leq \theta_{w_t|c_j} \leq 1$ dan $\sum_t \theta_{w_t|c_j} = 1$

Disini perkiraan untuk kemungkinan untuk kata w_t dalam kelas c_j adalah

$$\theta_{w_t|c_j} = P(w_t|c_j;\theta_j) = \frac{1 + \sum_{1=1}^{D} N_{it}\,P(c_j|d_i)}{|V| + \sum_{s=1}^{|V|}\sum_{i=1}^{D} N_{is}\,P(c_j|d_i)}$$

2.6. Sistem Terdistribusi

Hampir semua sistem berbasis computer yang besar saat ini merupakan sistem terdistribusi (sistem tersebar). Sitem terdistribusi adalah sistem dimana pemrosesan informasi didistribusikan pada beberapa computer dan tidak terbatas hanya pada satu mesin saja. Jelas rekayasa terdistribusi memiliki banyak kesamaan dengan rekayasa perangkat lunak lainnya tetapi ada isu-isu khusus yang harus diperhitungkan ketika merancang tipe sistem ini. Perekayasa perangkat lunak harus menyadari dan memperhitungkan karena sistem terdistribusi ini banyak digunakan. Belum lama ini

kebanyakan sistem besar masih menggunakan sistem sentral yang berjalan pada satu *mainframe* dengan *terminal-terminal* yang terhubung kepadanya. Sistem tersebut bayak kelemahannya dimana terminal-terminal hanya sedikit kemampuan pemrosesannya dan semua tergantung pada komputer sentral.

Sampai saai ini ada tipe sistem yang utama yaitu:

- Sistem *Personal* yang tidak terditribusi dan dirancang untuk satu *workstation* saja.
- Sistem *Embedded* yang bejalan pada satu *processor* atau pada kelompok prosessor yang terintegrasi.
- Sistem Terdistribusi dimana perangkat lunak sistem berjalan pada kelompok *processor* yang bekerja sama dan terintegrasi secara longgar, dengan dihubungkan oleh jaringan. Contohnya sistem ATM bank, sistem *groupware*, dll

Menurut (Coulouris, et al., 2012) mengidentifikasi enam karakteristik yang penting untuk sistem terdistribusi yaitu:

- Pemakain bersama sumber daya
- Keterbukaan. Keterbukaan sistem adalah terbuka untuk banyak sistem operasi dan banyak vendor
- Konkurensi. Sitem terdistribusi memungkinkan beberapa proses dapat beroperasi pada saat yang sama pada berbagai computer di jaringan. Proses ini dapat (tapi tidak perlu) berkomunikasi satu dengan lainnya pada saat operasi normalnya.
- Skalabilitas. Sitem terdistribusi dapat diskala dengan meng-*upgrade* atau menambahkan sumber daya baru untuk memenuhi kebutuhan sistem.
- Toleransi kesalahan. Sitem terdistribusi bersifat toleran terhadap beberapa kegagalan perangkat keras dan lunak dan layanan terdegradasi dapat diberikan ketika terjadi kegagalan.
- Transparansi. Sitem terdistribusi adalah bersifat terbuka bagi pengguna.

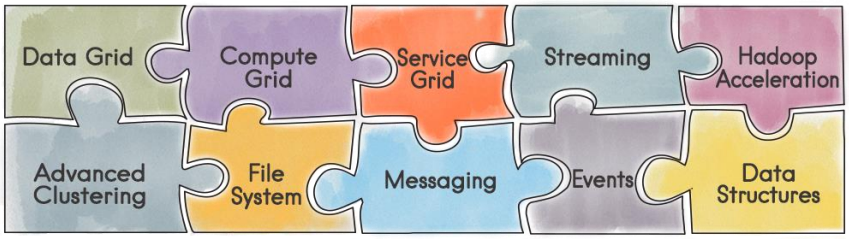
Selain hal-hal tersebut ada juga kelemaham dari Sistem terdistribusi yaitu:

- Kompleksitas. Sistem terdistribusi bersigat lebih kompleks daripada sistem sentral.
- Keamanan. Sistem terdistribusi dapat diakses dari beberapa komputer dan jalur jaringan mudah disadap, sehingga keamanan jarinagan sistem terdistribusi menjadi masalah yang besar.
- Kemampuan untuk dikendalikan. Komputer yang terdapat di sistem terdistribusi bis aterdiri dari berbagai tipe yang berbeda dan mungkin dijalankan pada sistem operasi yang berbeda pula. Kesalahan pada satu mesin bisa berakibat pada yang lainnya. Sehingga harus banyak usaha untuk mengendalikannya.
- Tidak dapat diramalkan. Sistem terdistribusi tidak dpat diramalkan tanggapannya. Tanggapan tergantung beban total sistem, pengorganisasiandan beban jaringannya.

2.7. *Apache Ignite*

Apache Ignite In-Memory Data Fabric merupakan *platform* yang memungkinkan untuk melakukan komputasi dan transaksi dengan dataset yang besar secara real-time, terintegrasi dan terdistribusi yang memungkinkan memiliki kecepatan yang lebih besar dibandingkan dengan sistem penyimpanan tradisional maupun teknologi *flash.*

Apache Ignite In-Memory Data Fabric dirancang untuk melakukan *in-memory computing* untuk mendapatkan performa komputasi yang tinggi, *advance data-grid, service grid,* maupun *streaming*.

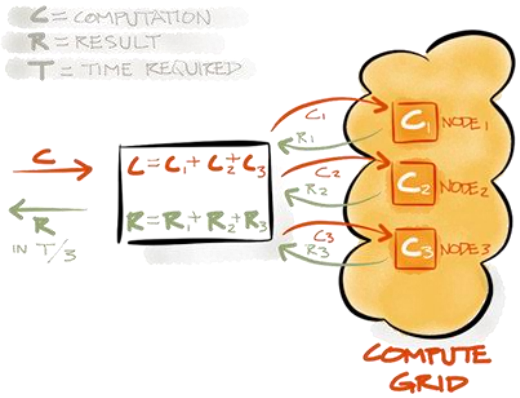


Gambar 2.4 Fitur *Apache Ignite* (Apache Ignite, 2016)

2.8.1. In-Memory Compute Grid

Komputasi terdistribusi yang dilakukan secara paralel untuk meningkatkan performa, *low latency*dan *linear scalability*. Apache Ignite menyedikan API yang memungkinkan untuk mendistribusikan komputasi dan pemrosesan data dalam beberapa komputer didalam *cluster.*

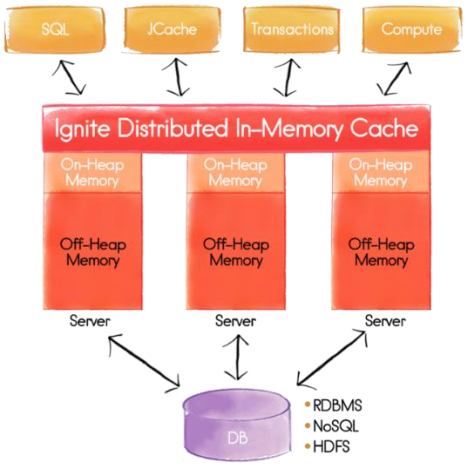
Distributed parallel processing memiliki kemampuan untuk mengeksekusi komputasi didalam cluster dan menngembalikan hasilnya kembali.



Gambar 2.5 *In-Memory Compute Grid* (Apache Ignite, 2016)

2.8.2. In-Memory Data Grid

In-Memory Data Grid merupakan *key-value in-memory store* yang memungkinkan caching data in-memory didalam *cluster* terdistribusi. Ignite data grid merupakan salah satu pengimplementasian transaksional atau *atomic* data dalam *cluster* terdistribusi.



Gambar 2.6 *In-Memory Data Grid* (Apache Ignite, 2016)

2.8. Penelitian Terdahulu

Penggunaan *Focused Crawler* didukung oleh penelitian terdahulu pada tahun 2014, oleh Rashmi Janbandhu, Prashant Dahiwaledan M.M.Raghuwanshi mereka meneliti algoritma apa yang paling efektif digunakan dalam mesin *crawler*. Mereka

membandingkan beberapa algoritma dalam mesin crawler seperti *Breadth First Search Algorithm, Depth First Search Algorithm, Page Rank Algorithm*. Penelitian ini menunjukkan bahwa *Focused Crawling Algorithm* mempunyai kelebihan daripada algoritma yang lainnya, dimana algoritma ini mempunyai *response time* yang paling kecil daripada yang lain.

Penelitian yang dilakukan Ricardo Baeza-Yates, Mauricio Marin, Carlos Castillo, Andrea Rodriguez pada tahun 2005 mereka meneliti algoritma apa yang efektif untuk dipakai dalam *Page Ordering* dalam *web crawler*. Algoritma *Breadth-First* mempunyai performa yang buruk dibandingkan dengan algoritma yang lain. *Larger-Sites-First* mempunyai kelebihan daripada OPIC, yaitu membutuhkan waktu perhitungan yang sedikit dan juga tidak membutuhkan bobot yang terdapat pada *link* pada halaman yang dilakukan oleh OPIC. Dengan begitu *Larger-Sites-First* yang mempunyai performa yang lebih baik daripada algoritma yang lainnya. Algoritma ini mengurutkan website yang akan di-*crawl* berdasarkan halaman yang dimiliki dari terbanyak sampai yang terkecil.

Pada tahun 2013, Harry T Yani Achsan, Wahyu Catur Wibowo mereka meneliti penggunaan *multi thread web crawler* pada satu komputer yang akan didistribusikan ke *public proxy server*. Karena jika tidak memakai *proxy server, web crawler* dapat di anggap sebagai *'cyber-attack'* dan akan di *'ban'* oleh *web server*. Mereka juga meneliti bahwa *multi thread web crawler* harus memakai jumlah *thread* yang optimal untuk memaksimalkan waktu men-*download*, karena jika terlalu banyak *thread* akan mengurangi kenerja komputer, tetapi jika terlalu banyak *thread* akan mengurangi kecepatan dalam mengumpulkan data. Mereka menyimpulkan bahwa menggunakan multi thread *web crawler* yang didistribusikan ke *public proxy server* merupakan cara yang lebih mudah dan murah daripada menggunakan *web crawler* terdistribusi yang menggunakan beberapa komputer.

Pada tahun 2010, Wenxian Wang, Xingshu Chen, Yongbin Zou, Haizhou Wangdan Zongkun Dai mereka meneliti penggunaan TF-IDF *(Term Frequency – Inverse Document Frequency)* dalam meng-*extract* konten yang ada pada halaman web dan menggunakan *Naive Bayes Classifier* untuk menghitung *pagerank* dalam *focused crawler*. Hasilnya adalah *Crawler* yang mereka miliki memiliki perporma yang lebih baik daripada *PageRank Crawler* dan *BFS (Breath First Search) Crawler.*