BAB 2
TINJAUAN PUSTAKA

2.1 Tinjauan Tempat Penelitian

Tinjauan umum perusahaan merupakan pembahasan mengenai profil, struktur organisasi, serta deskripsi jabatan pada perusahaan yang akan menjadi tempat penelitian.

2.1.1 Visi dan Misi

Berikut adalah Visi Misi dari PT. Kinest Kreatif Ideata:

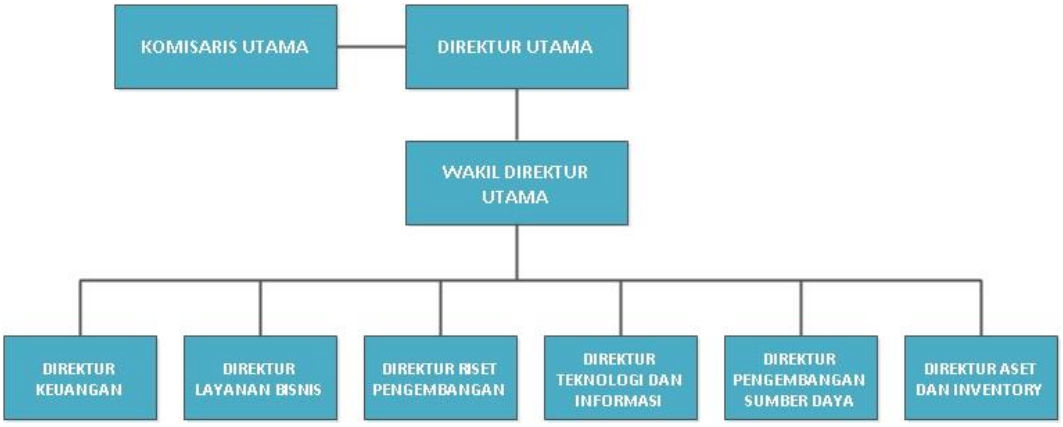
1. Visi

   To be an Indonesian company leader at digital industry.

2. Misi
   a. *Build a strong and coherence teamwork.*
   b. *Generate multilevel partnership network market.*
   c. *Best management quality system.*
   d. *Smooth and efficient management execution.*
   e. *Make strong financial management.*

2.1.2 Struktur Organisasi

Struktur organisasi pada PT. Kinest Kreatif Ideata dapat dilihat pada Gambar 2.1.



Gambar 2.1 Struktur organisasi PT. Kinest Kreatif Ideata

2.1.3 Deskripsi Kerja Jabatan

Berikut adalah pemaparan deskripsi kerja untuk tiap-tiap jabatan di PT. Kinest Kreatif Ideata:

1. Direktorat Keuangan

   Fokus pada pengelolaan keuangan Perusahaan serta mengendalikan operasi keuangan secara terpusat.

2. Direktorat Layanan Bisnis

   Fokus pada pengembangan bisnis perusahaan secara terpusat melalui produk Kakatu dan Hyjabs.

3. Direktorat Riset dan Pengembangan

   Fokus pada pengelolaan riset dan teknologi yang menyangkut kedua produk dari sisi teknlogi dan *user experience* produk.

4. Direktorat Teknologi Informasi

   Fokus pada pengelolaan *infrastructure strategy & governance*, IT *Strategy & Governance, and Solution* serta pengelolaan pendayagunaan IT dan s*ervice operation & management*.

5. Direktorat Pengembangan dan Sumber Daya

   Fokus pada manajemen SDM Perusahaan serta penyelenggaraan operasional SDM.

6. Direktorat Aset dan *Inventory*

   Fokus pada pengelolaan *asset* perusahaan dan *inventory* penggunaan.

2.1.4 Kakatu

Kakatu merupakan sebuah aplikasi yang dikembangkan oleh PT. Kinest Kreatif Ideata dan berada di bawah direktur riset dan pengembangan. Aplikasi ini adalah cara yang aman, mudah, dan menyenangkan bagi anak dalam menggunakan *device* Android. Selain itu Kakatu dapat menjadi sebuah media *parental control* untuk mengawasi dan membatasi penggunaan *handphone* anak agar anak tidak sampai mengakses konten aplikasi yang tidak seharusnya.

Aplikasi ini menawarkan fungsionalitas berbeda kepada dua pengguna dengan peran berbeda, yakni orangtua dan anak. Orangtua dapat memanfaatkan

fitur pembatasan aplikasi, pembatasan panggilan telpon masuk, pembatasan waktu penggunaan HP anak, dan meihat riwayat penggunaan HP anak.

2.2 Landasan Teori

Landasan teori merupakan pemaparan konsep dasar dan beberapa teori berkaitan dengan penelitian yang dilakukan. Berikut adalah landasan teori dalam penelitian ini:

2.2.1 Parental Control

Parental control merupakan sebuah fitur yang biasanya digunakan pada layanan televisi digital, komputer, *video game*, ponsel ataupun perangkat lunak. Secara umum *parental control* terbagi ke dalam empat kategori: *content filtering*, berfungsi membatasi akses terhadap konten yang tidak layak digunakan oleh usia tertentu; *usage controls*, berfungsi untuk membatasi penggunaan perangkat dengan menentukan batas waktu penggunaan atau melarang penggunaan fitur tertentu; *computer usage management tool*, yang mewajibkan menggunakan beberapa perangkat lunak tertentu; dan *monitoring*, yang berfungsi untuk melacak lokasi ataupun aktifitas saat menggunakan perangkat [5].

Meningkatnya penggunaan ponsel yang dilengkapi dengan fitur *browser* internet dan banyaknya aplikasi yang dapat diunduh menimbulkan kebutuhan peran *parental control* terhadap penggunaan ponsel [6]. *Parental control* yang berupa perangkat lunak dalam ponsel memungkinkan orangtua untuk membatasi aplikasi apa saja yang dapat digunakan oleh anak, membatasi waktu penggunaan ponsel, melacak lokasi anak mereka, serta mengawasi transaksi lain yang dilakukan anak melalui ponsel seperti SMS, panggilan masuk/keluar, dan MMS.

2.2.2 Sistem Rekomendasi

Sistem rekomendasi merupakan suatu model aplikasi ataupun teknik yang berfungsi memberikan usulan mengenai *item* untuk digunakan oleh pengguna. "*Item*" merupakan istilah standar yang digunakan untuk menggambarkan apa yang direkomendasikan oleh sistem [1]. Normalnya suatu sistem rekomendasi berfokus

kepada *item* tertentu seperti musik, CD, film, dan lain sebagainya. Teknik dalam sistem rekomendasi, desain, maupun *user interface* harus disesuaikan sedemikian rupa dengan *item* tersebut. Sistem rekomendasi merupakan model aplikasi dari hasil observasi terhadap keadaan dan keinginan pengguna. Oleh karena itu sistem rekomendasi memerlukan teknik rekomendasi yang tepat agar yang direkomendasikan sesuai dengan keinginan pengguna serta mempermudah pengguna menentukan produk yang tepat untuk digunakan. Sistem rekomendasi ditujukan kepada pengguna yang kurang memiliki pengalaman maupun pengetahuan dalam mengevaluasi alternatif pilihan *item* yang berpotensial.

2.2.2.1 Teknik Rekomendasi

Terdapat beberapa pedekatan dalam membangun sebuah sistem rekomendasi diataranya *Content-based*, *Collaborative Filtering*, *Demographic*, *Knowledge-based*, *Community-based*, dan *Hybrid Recommender Systems* [1]. Setiap teknik tersebut digunakan sesuai dengan permasalahan yang akan diselesaikan. Pemaparan untuk setiap pendekatan tersebut adalah sebagai berikut:

1. *Content-based*, merupakan pendekatan yang merekomendasikan *item* yang memiliki kesamaan dengan *item* lain yang pernah disukai oleh pengguna pada waktu lampau. Kemiripan dari setiap *item* diukur berdasarkan fitur yang berhubungan antara satu sama lainnya. Sebagai contoh, apabila pengguna menyukai lagu dengan *genre* jazz maka sistem rekomendasi akan mencoba merekomendasikan lagu berbeda dengan *genre* yang sama.
2. *Collaborative Filtering* adalah pendekatan yang merekomendasikan *item* yang pernah disukai oleh pengguna lain dengan selera yang sama kepada pengguna target. Kemiripan selera dihitung berdasarkan kemiripan *rating* terhadap *item* yang pernah dilakukan oleh kedua pengguna tersebut.
3. *Demographic* adalah sistem yang merekomendasikan *item* berdasarkan letak wilayah dari target pengguna atau berdasarkan umur pengguna.
4. *Knowledge-based* merekomendasikan *item* berdasarkan domain pengetahuan yang spesifik tentang bagaimana fitur suatu *item* dapat memenuhi kebutuhan

dan preferensi pengguna. Pendekatan ini akan dibahas lebih mendalam pada subbab 2.2.2.2.

5. *Community-based* merekomendasikan *item* berdasarkan pada preferensi teman target pengguna di media sosial. Pendekatan ini juga biasa disebut dengan *Social Recommender System*.
6. *Hybrid Recommender Systems* menggabungkan dua teknik yang sudah dipaparkan. Sistem *hybrid* menggabungkan teknik A dan teknik B dengan tujuan menutupi kekurangan yang dimiliki oleh teknik A dengan kelebihan yang dimiliki oleh teknik B.

2.2.2.2 Knowledge-based Recommender Systems

Sistem rekomendasi dengan pendekatan yang sering digunakan (*content-based* dan *collaborative filtering*) sangat efektif jika digunakan untuk merekomendasikan *item* berdasarkan selera pengguna seperti pada kasus dengan *item* buku, film, ataupun artikel berita [1]. Ketika sistem rekomendasi bergantung kepada kebutuhan pengguna yang didefinisikan secara eksplisit, kedua pendekatan tersebut bukanlah pilihan terbaik karena bentuk atau perumusan dari kebutuhan pengguna yang didefinisikan secara eksplisit tidak sesuai dengan *framework* dari pendekatan *content-based* maupun *collaborative filtering*.

Knowledge-based adalah pendekatan sistem rekomendasi dengan menggunakan kebutuhan pengguna yang didefinisikan secara eksplisit dalam bentuk *recommendation rules*. Pendekatan ini terbagi kembali ke dalam dua tipe dasar yaitu *case-based* dan *constraint-based* [8]. Kedua pendekatan tersebut memiliki kesamaan pada proses yang dilakukan untuk meghasilkan rekomendasi, yakni pengguna menetapkan kebutuhan, kemudian sistem akan mencoba untuk menentukan solusi. Perbedaan dari kedua tipe pendekatan ini terletak pada bagaimana mereka menggunakan *knowledge* yang tersedia untuk menghasilkan rekomendasi. *Case-based* menghasilkan rekomendasi dengan cara menghitung ukuran kemiripan *item* (menggunakan *threshold* tertentu) dengan kebutuhan yang telah ditentukan oleh pengguna. Di lain pihak, *constraint-based* menghasilkan rekomendasi dengan mencari sekumpulan *item* yang dapat memenuhi

recommendation rules. Secara umum *constraint-based* dapat dipresentasikan sebagai *constraint satisfaction problem* (CSP) dan dapat diselesaikan dengan menggunakan CSP *solver* ataupun menggunakan *conjunctive query* yang dieksekusi dan diselesaikan oleh *database engine*. *Conjunctive query* memandang permasalah rekomendasi sebagai data *filtering* [8]. *Conjunctive query* adalah *query* database yang berisi sekumpulan kriteria seleksi yang saling terhubung, *query* tersebut didefinisikan sebagai:

$$\sigma_{[criteria]}(P) \quad \textbf{(2-1)}$$

Dimana σ merepresentasikan operator seleksi, *criteria* adalah *recommendation rules* yang berlaku, dan *P* adalah tabel dari kumpulan produk yang tersedia.

2.2.3 Information Retrieval

Information Retrieval 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* terdiri dari tiga komponen utama, yaitu masukan (*Input*), pemroses (*processor*) dan keluaran (*output*). *Input* harus berupa representasi yang tepat dari setiap dokumen dan *query* agar dapat diolah oleh pemroses. Pemroses (*Processor*) bertugas menstrukturkan informasi dalam bentuk yang tepat, misalnya dengan pengindeksan dan klasifikasi serta melakukan proses *information retrieval*, yaitu dengan menjalankan suatu strategi pencarian sebagai respon dari *query*. *Output* adalah keluaran yang diberikan oleh pemroses, biasanya berbentuk informasi tentang suatu dokumen, dokumen itu sendiri dan acuan ke dokumen lain [9].

2.2.4 Klasifikasi Sentimen

Klasifikasi sentimen adalah proses mengkategorisasikan sebuah dokumen yang tidak berlabel ke dalam kategori yang telah ditentukan. Secara umum permasalahan dalam klasifikasi sentimen dapat digambarkan dengan: jika diberikan satu set dokumen D, fungsi yang disebut *classifier* akan mengklasifikasi setiap dokumen d ϵ *D* kepada salah satu dari kelas positif atau negatif. Positif memiliki

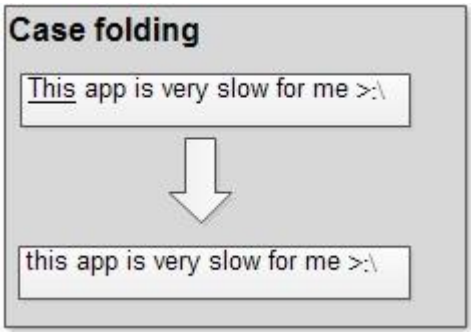
arti bahwa dokumen d mengekspresikan opini positif dan negatif memiliki arti bahwa dokumen d mengkspresikan opini negatif [2].

2.2.4.1 Text Preprocessing

Pada klasifikasi sentimen data yang digunakan berasal dari dokumen atau teks yang tidak terstruktur. Oleh karena itu, dibutuhkan *text preprocessing* yang berfungsi untuk mengubah bentuk data yang semula tidak terstruktur menjadi data yang terstruktur. *Text preprocessing* pada umumnya terdiri dari tahapan *case folding*, *stopword removal*, *covert emoticon*, *convert negation*, *tokenization*, *stemming*, dan *term weighting*. Berikut adalah pemaparan dari setiap tahapan tersebut:

1. *Case folding*

Case folding merupakan proses yang dilakukan untuk menyeragamkan karakter pada data (dokumen/teks). Pada proses ini semua huruf besar (*uppercase*) dijadikan huruf kecil (*lowercase*). Proses *case folding* dapat dilihat pada Gambar 2.2.

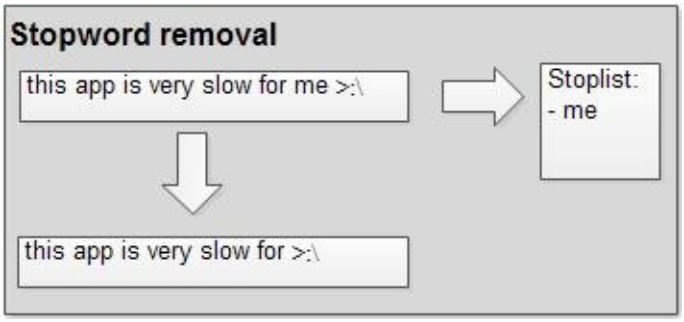


Gambar 2.2 Gambaran proses *case folding*

2. *Stopword removal*

Stopword removal adalah proses yang dilakukan untuk menghilangkan kata yang dianggap tidak memberikan pengaruh dalam menentukan suatu kategori tertentu dalam suatu dokumen. Sebelum dilakukan proses *stopword removal*, terlebih dahulu dibuat kata-kata yang termasuk ke dalam *stopwords* atau *stoplist*. Kata-kata yang termasuk *stoplist* tersebut biasanya berupa kata ganti orang, kata

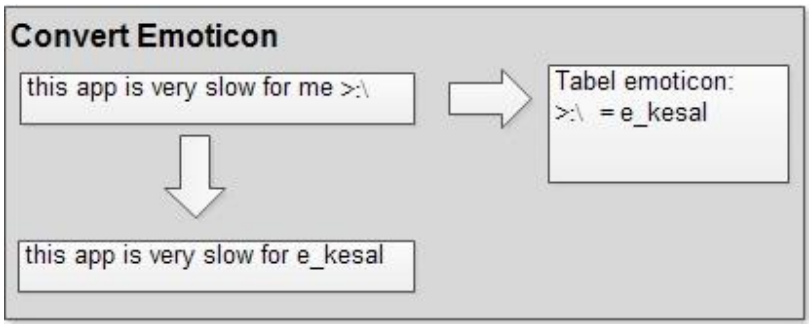
penghubung, pronominal penunjuk, dan lain sebagainya [10]. Proses ini dapat dilihat pada Gambar 2.3.



Gambar 2.3 Gambaran proses *stopword removal*

3. *Convert emoticon*

Sebuah *emoticon* merupakan salah satu cara untuk mengekspresikan ungkapan perasaan secara tekstual [11]. Pada penelitian sebelumnya [11] *emoticon* dalam suatu kalimat dianggap penting dan memiliki kontribusi dalam menentukan nilai sentimen suatu kalimat. Proses *convert emoticon* dapat dilihat pada Gambar 2.4.



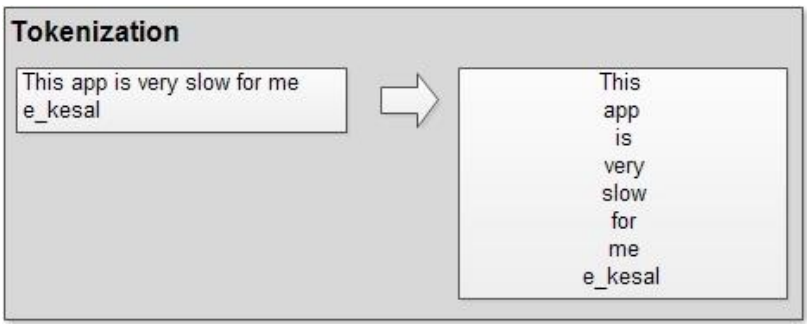
Gambar 2.4 Gambaran proses *convert emoticon*

4. *Convert negation*

Beberapa kata yang bersifat negasi, akan merubah nilai sentimen suatu dokumen. Ketika banyak kata negasi adalah ganjil, maka sentimen dokumen tersebut akan diubah. Contoh beberapa kata yang bersifat negasi dalam bahasa Inggris adalah "*no*" dan "*not*". Sebagai contoh, kata "*good*" adalah kata kunci positif namun jika sebelumnya diikuti kata "*not*" sehingga menjadi "*not good*" maka nilainya menjadi negatif.

5. *Tokenization*

Tokenization adalah proses mengubah teks menjadi ukuran token [11]. Pada penelitian ini token adalah kata atau *word* yang merupakan unit atomik terkecil yang memiliki arti. Proses *tokenization* dilakukan dengan cara memisahkan kata berdasarkan spasi yang terletak diantara dua kata. Proses *tokenization* dapat dilihat pada Gambar 2.5.



Gambar 2.5 Gambaran proses *tokenization*

6. *Stemming*

Pada analisa temu kembali informasi, imbuhan merupakan bagian informasi yang tidak memiliki makna seperti halnya *stopword* [12]. Oleh karena itu, dilakukan proses pengambilan kata dasar dari sebuah kata yang berimbuhan, proses ini disebut *stemming*. Proses *stemming* memiliki beberapa pendekatan dalam hal mendapatkan kata dasar, diantaranya [12]: *corpus-based* yang menggunakan kamus atau kumpulan dokumen, pendekatan ini membutuhkan waktu komputasi yang tinggi; pendekatan berikutnya adalah *inflection* yang mendapatkan kata dasar dengan cara membuang imbuhan sesuai dengan aturan bahasa. Salah satu algoritma *stemming* untuk bahasa inggris yang sering digunakan adalah algoritma Porter yang dibuat oleh Martin Porter. Dalam proses untuk mendapatkan kata dasar dari kata yang berimbuhan, *stemmer* porter menggunakan enam langkah yang diterapkan secara berurutan diantaranya:

a. Menghilangkan plural dan akhiran –ed serta akhiran –ing
b. Mengubah huruf y menjadi i pada akhir kata apabila terdapat huruf vokal lain pada kata tersebut
c. Menghilangkan akhiran –ization dan –ational
d. Mengatasi akhiran –full dan –ness

e. Menghilangkan akhiran –ant dan –ence
f. Menghilangkan akhiran –e

7. *Term Weighting*

Dalam *text preprocessing*, *term weighting* merupakan salah satu tahapan yang sangat penting dimana representasi teks diubah dari dokumen tekstual ke dalam format yang dapat dikenali dan diklasifikasi oleh *classifier*. Tahapan ini dilakukan dengan tujuan untuk memberikan suatu nilai/bobot pada *term* yang terdapat pada suatu dokumen [13]. Bobot yang diberikan terhadap sebuah *term* bergantung kepada metode yang digunakan untuk membobotinya. Salah satu metode yang sering digunakan dalam klasifikasi teks adalah TF.IDF.

Term Frequency (TF) merupakan salah satu metode pembobotan yang paling sederhana. Pada metode ini, setiap *term* diasumsikan memiliki proporsi kepentingan sesuai dengan jumlah munculnya *term* tersebut dalam dokumen [13]. Persamaan TF adalah sebagai berikut:

$$W(d,t) = TF(d,t) \quad \textbf{(2-2)}$$

dimana $TF(d,t)$ adalah frekuensi kemunculan *term* t pada dokumen d.

Inverse Document Frequency (IDF) adalah metode pembobotan yang menganggap sebuah term bernilai/berharga jika kemunculannya pada koleksi/himpunan dokumen adalah jarang. Tingkat kepentingan bobot dari suatu term juga diasumsikan berbanding terbalik dengan jumlah dokumen yang mengandung term tersebut. Oleh karena itu jika jumlah kemunculan suatu *term* pada kumpulan dokumen adalah banyak, maka *term* tersebut akan dianggap tidak berharga. Persamaan IDF adalah sebagai berikut:

$$IDF(t) = log\frac{N}{df(t)} \quad \textbf{(2-3)}$$

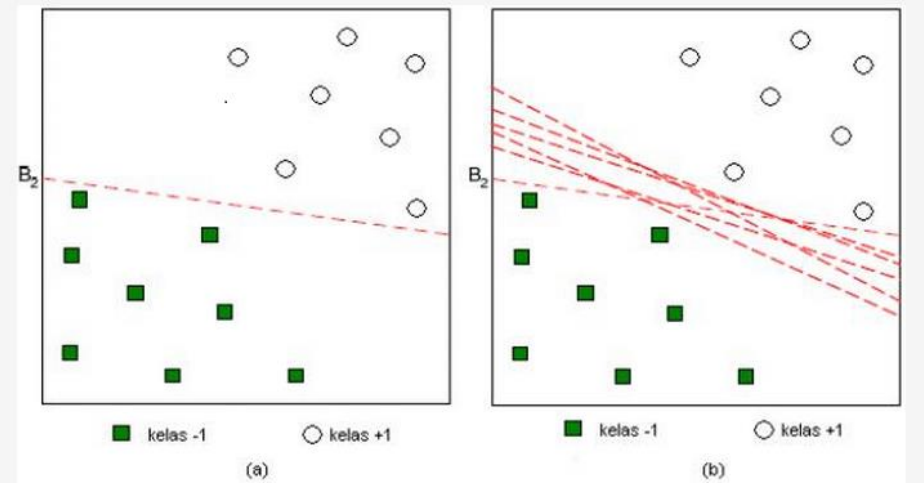
dimana N adalah banyaknya dokumen, dan *df(t)* adalah banyak dokumen yang mengandung *term* t.

Kombinasi metode TF dan IDF (TF.IDF) adalah metode yang diusulkan oleh Salton dengan tujuan untuk mendapatkan performansi yang lebih baik. Metode gabungan ini merupakan hasil perkalian antara metode TF dengan metode IDF. Persamaan TF.IDF adalah sebagai berikut:

$$TF.IDF = TF(d,t) * IDF(d,t) \quad \textbf{(2-4)}$$

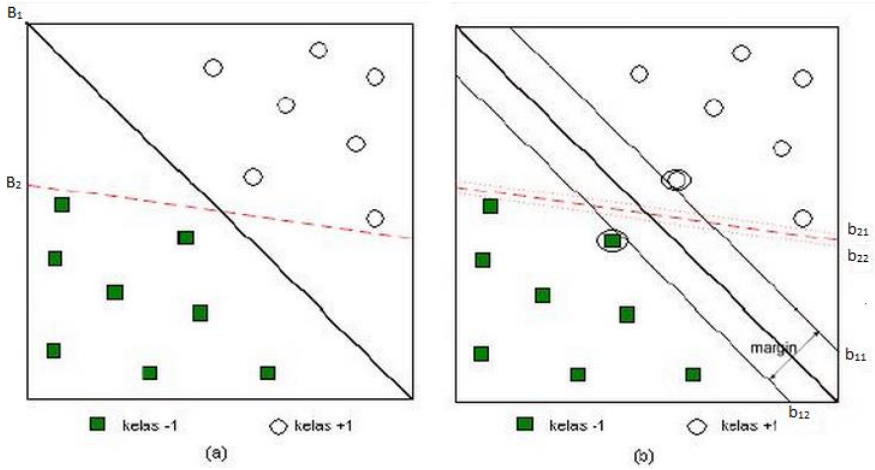
2.2.4.2 Support Vector Machine

Support Vector Machine (SVM) merupakan metode *supervised* (*predictive*) yang dapat digunakan untuk prediksi (*support vector regression*) atau klasifikasi (*support vector classification*) [14]. Pada saat ini SVM telah diaplikasikan dan memiliki performasni yang baik pada beberapa bidang, salah satunya adalah klasifikasi teks. Konsep dasar SVM adalah bagaimana menemukan atau mencari fungsi pemisah (*hyperplane*) terbaik yang memisahkan dua kelas pada *input space* atau ruang input. Cara kerja SVM dapat dilihat pada Gambar 2.6 berikut:



Gambar 2.6 Gambaran cara kerja SVM [14]

Gambar 2.6 memperlihatkan beberapa *record*/*pattern*/titik yang merupakan anggota dari dua buah kelas yaitu kelas -1 dan kelas +1. Titik yang merupakan anggota pada kelas -1 disimbolkan dengan warna hijau dan berbentuk kotak, sedangkan titik yang tergabung pada kelas +1 disimbolkan dengan warna putih dan berbentuk bulat. Masalah klasifikasi pada SVM dapat diterjemahkan dengan usaha untuk menemukan fungsi pemisah (*hyperplane*) yang memisahkan antara kedua kelompok tersebut. Garis B2 pada Gambar 2.6 (a) mengilustrasikan sebuah *hyperplane*, sedangkan beberapa garis merah pada Gambar 2.6 (b) mengilustrasikan berbagai alternatif *hyperplane.*



Gambar 2.7 Gambaran penentuan *hyperplane* terbaik [14]

Fungsi pemisah (*hyperplane*) terbaik antara dua kelas dapat ditemukan dengan mencari margin *hyperplane* terbesar. Tujuan dari usaha ini adalah untuk meningkatkan probabilitas pengelompokan secara benar pada data *testing*. Margin pada konsep ini adalah jarak antara titik terdekat dari masing-masing kelas terhadap *hyperplane*. Titik terdekat dari masing-masing kelas ini disebut *support vector*, diilustrasikan dengan titik yang dilingkari pada Gambar 2.7 (b). Dengan kata lain tugas utama dari SVM adalah menemukan *hyperplane* yang terletak tepat di tengah-tengah antara dua *support vector* dari kelompok kelas yang berbeda, dan besar atau jarak margin terhadap masing-masing *support vector* adalah sama.

Secara matematika, kasus klasifikasi secara linear dengan SVM dapat dituliskan sebagai berikut [15]:

$$g(x) := sgn(f(x)) \quad \textbf{(2-5)}$$

$$\text{dengan } \mathrm{f}(x) = \mathrm{w}\, x + b$$

$$\text{atau } g(x) = \begin{cases} +1, jika\ (wx + b) \geq +1 \\ -1, jika\ (wx + b) \leq -1 \end{cases}$$

dimana x,$w \in \Re^n$ dan $b \in \Re$. Pada SVM, poin-poin yang merupakan vektor dinotasikan sebagai $x_i \in \Re^n$ sedangkan label masing-masing dinotasikan sebagai $y_i \in \{-1, +1\}$ untuk i = 1, 2, …, l dimana l adalah banyaknya data. Sebuah pattern x_i yang termasuk kelas -1 dapat dirumuskan sebagai *pattern* yang memenuhi

pertidaksamaan $w\,x + b \leq -1$ sedangkan *pattern* x_i yang termasuk kelas +1 dapat dirumuskan sebagai *pattern* yang memenuhi pertidaksamaan $w\,x + b \geq +1$.

Mencari *hyperplane* terbaik ekuivalen dengan memaksimalkan margin atau jarak antara dua set objek dari kelas yang berbeda. Margin terbesar dapat ditemukan dengan memaksimalkan nilai jarak antara *hyperplane* dan titik terdekatnya, yaitu $\frac{1}{||w||}$. Hal ini dapat dirumuskan sebagai *Quadratic Programming* (QP) *problem*, yaitu mencari titik minimal persamaan **(2-6)**:

$$min\ \frac{1}{2}||w||^2 \qquad \textbf{(2-6)}$$

dengan memperhatikan *constraint*:

$$yi(xi.w + b) - 1 \geq 0$$

Permasalahan ini dapat dipecahkan dengan berbagai teknik komputasi diantaranya *Lagrangae Multiplier*:

$$\max \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j x_i^T\, x_j \qquad \textbf{(2-7)}$$

Subject to

$$\sum_{i=1}^{l} \alpha_i y_i = 0$$

$$0 \leq \alpha_i, i = 1,2,..l.$$

Dengan *dot product* $x_i x_j$ sering diganti dengan simbol K yang merupakan matriks kernel, dan support vectornya adalah α_i yang bernilai lebih besar dari nol.

Secara umum, kasus-kasus di dunia nyata adalah kasus yang tidak linier. Data ini sulit dipisahkan secara linear, metode kernel adalah salah satu cara untuk mengatasi hal tersebut [15]. Fungsi kernel yang biasanya dipakai dalam literatur SVM adalah [15]:

a. Linear: $x^T x$
b. Polynomial: $(x^T x_i\ + 1)^p$
c. Radial basis function (RBF):$\exp(-\frac{1}{2\sigma^2}\ \|x - x_i\|^2)$
d. Tangent hyperbolic (sigmoid): $\tanh(\beta x^T x_i + \beta_1)$, dimana $\beta,\ \beta_1\ \in\ \Re$

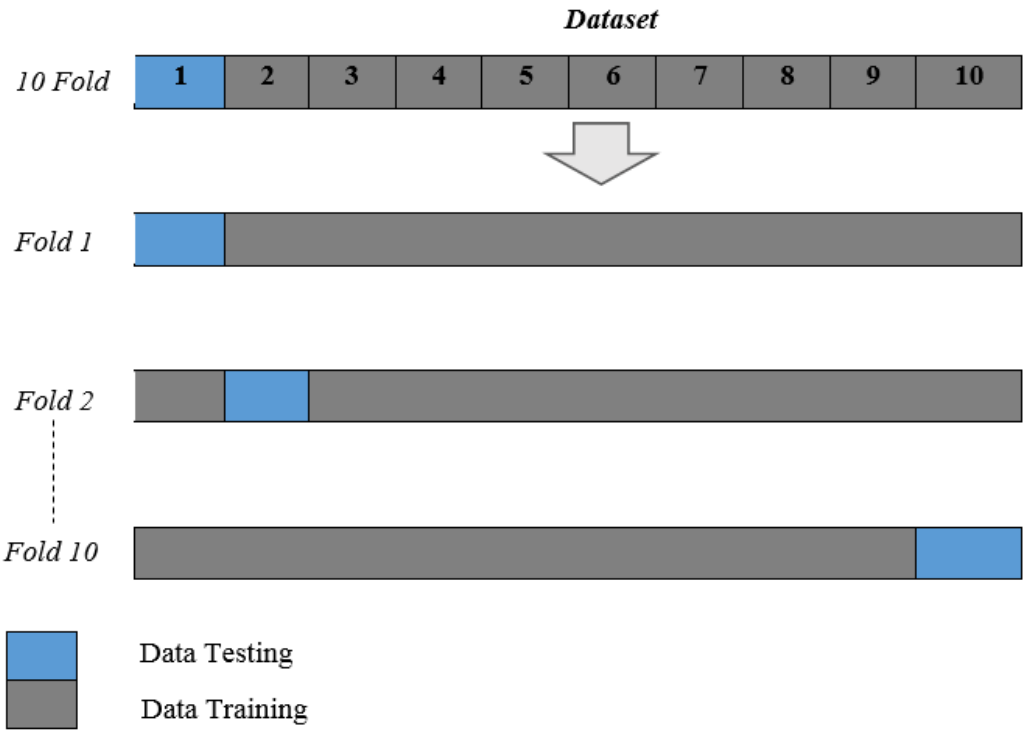
Setelah itu, vector w dapat dicari menggunakan persamaan:

$$w = \sum_{i=1}^{l} \alpha_i x_i \qquad \textbf{(2-8)}$$

2.2.4.3 K-fold Cross Validation

Jumlah data memegang peranan penting di dalam algoritma *machine learning*. Jumlah data yang sedikit (<1000 *instance*) dapat membuat algoritma *machine learning* tidak akurat. Algoritma *machine learning* merekomendasikan jumlah *instance* yang banyak (>1000 *instance*) namun data itu sendiri tidak mudah untuk diperoleh.

Metode yang digunakan untuk mengevaluasi kinerja *classifier* pada penelitian ini adalah *k-fold cross-validation*. *K-fold cross validation* adalah teknik evaluasi yang dapat digunakan apabila memiliki jumlah data yang terbatas (jumlah *instance* tidak banyak) [15]. *K-fold cross validation* merupakan salah satu metode yang digunakan untuk mengetahui rata-rata keberhasilan dari suatu sistem dengan cara melakukan perulangan dengan mengacak atribut masukan sehingga sistem tersebut teruji untuk beberapa atribut input yang acak. *K-fold cross validation* diawali dengan membagi data sejumlah *n-fold* yang diinginkan. Dalam proses *cross validation* data akan dibagi ke dalam *n* buah partisi dengan ukuran yang sama (D_1,D_2,D_3..D_n), selanjutnya proses testing dan training dilakukan sebanyak n kali. Dalam iterasi ke-$_i$, partisi D_i akan menjadi data testing dan sisanya akan menjadi data training. Penggunaan jumlah *fold* terbaik untuk uji validitas, dianjurkan menggunakan 10-*fold cross validation* dalam model [16]. Contoh pembagian dataset dalam proses 10-*fold cross validation* terlihat pada Gambar 2.8.



Gambar 2.8 Iterasi pada 10-*fold cross validation*

Cara kerja *K-fold cross validation* adalah sebagai berikut:

1. Total *instance* dibagi menjadi N bagian.
2. *Fold* ke-1 adalah ketika bagian ke-1 menjadi data uji (testing data) dan sisanya menjadi data latih (training data). Selanjutnya, hitung akurasi berdasarkan porsi data tersebut. Perhitungan akurasi tersebut dengan menggunakan persamaan sebagai berikut[]:

$$\text{Akurasi} = \frac{\sum \text{data uji benar klasifikasi}}{\sum \text{total data uji}} \times 100\% \quad \textbf{(2-8)}$$

3. *Fold* ke-2 adalah ketika bagian ke-2 menjadi data uji (testing data) dan sisanya menjadi data latih (training data). Selanjutnya, hitung akurasi berdasarkan porsi data tersebut.

Demikian seterusnya hingga mencapai *fold* ke-K. Kemudian, hitung rata-rata akurasi dari K buah akurasi di atas. Rata-rata akurasi tersebut menjadi akurasi final.

2.2.5 Web Scraping

Web scraping merupakan sekumpulan teknik untuk mendapatkan informasi dari sebuah website secara otomatis tanpa harus menyalinnya secara manual.

Tujuan dari web scraper adalah untuk mencari informasi tertentu, mengekstraksi informasi tersebut, kemudian menghimpunnya dalam halaman web yang baru. Beberapa web scraper memfokuskan dirinya untuk mengubah data yang tidak terstruktur dan menyimpannya ke dalam data yang terstruktur.

Secara teknis web scraping tidak termasuk dalam bidang data mining karena data mining mengandung upaya untuk memahami pola semantik atau tren dalam kumpulan data yang besar yang telah diperoleh. Aplikasi web scraping (umunya disebut juga dengan agen yang cerdas dan otomatis) berfokus hanya pada mendapatkan data dengan cara pengambilan dan ekstraksi serta dapat melibatkan kumpulan data dengan perbedaan ukuran yang signifikan [17].

2.2.6 Web Services

Web services adalah sebuah sistem perangkat lunak yang didesain guna mendukung kapabilitas sebuah sistem atau produk untuk berinteraksi dan berfungsi dengan sistem lain melalui sebuah jaringan [16]. Interaksi sistem lain dengan *web service* dilakukan dengan cara yang telah diuraikan menggunakan SOAP *messages*, biasanya disampaikan menggunakan HTTP dengan serialisasi XML bersamaan dengan standar web lainnya.

Web services tidak terpengaruh oleh *platform* dan menyediakan *method-method* yang dapat diakses oleh jaringan. *Web services* menggunakan XML untuk pertukaran data khususnya pada dua entitas bisnis yang berbeda. Beberapa karakteristik dari *web services* diantaranya: *message-based, standard-based, programming language independent*, dan *platform-neutral*. Beberapa *key* standar dalam *web services* adalah: JSON, XML, SOAP, WDSL, dan UDDI.

2.2.7 JSON

JSON (*JavaScript Object Notation*) adalah format pertukaran data yang ringan, mudah dibaca dan ditulis oleh manusia, serta mudah diterjemahkan dan dibuat (*generate*) oleh komputer [18]. Format ini dibuat berdasarkan bagian dari bahasa pemrograman JavaScript. JSON merupakan format teks yang tidak bergantung pada bahasa pemrograman apapun karena menggunakan gaya bahasa

yang umum digunakan oleh programmer keluarga C termasuk C, C++, C#, Java, JavaScript, Perl, Python dan lain sebagainya. Oleh karena sifat-sifat tersebut, menjadikan JSON ideal sebagai bahasa pertukaran-data. JSON terbuat dari dua struktur diantaranya [18]:

1. Kumpulan pasangan nama/nilai. Pada beberapa bahasa, hal ini dinyatakan sebagai objek (*object*), rekaman (*record*), struktur (*struct*), kamus (*dictionary*), tabel hash (*hash table*), daftar berkunci (*keyed list*), atau *associative array*.
2. Daftar nilai yang terurut (*an ordered list of values*). Pada kebanyakan bahasa, hal ini dinyatakan sebagai larik (*array*), vektor (*vector*), daftar (*list*), atau urutan (*sequence*).

2.2.8 Object Oriented Analysis and Design

Konsep OOAD mencangkup analisis dan desain sebuah sistem dengan pendekatan objek, yakni analisis berbasis objek (OOA) dan desain berorientasi objek (OOD). OOA adalah metode analisis yang memeriksa *requirement* yang harus dipenuhi oleh sebuah sistem dari sudut pandang kelas-kelas dan objek-objek yang ditemui dalam lingkup sistem. Sedangkan OOD adalah metode untuk mengarahkan arsitektur perangkat lunak yang didasarkan pada manipulasi objek-objek sistem atau subsistem [19].

2.2.9 Unified Modelling Language

Unified Modelling Language (UML) adalah bahasa standar yang digunakan untuk menjelaskan dan memvisualisasikan diagram dari proses analisis dan desain berorientasi objek. UML menyediakan standar pada notasi dan diagram yang bisa digunakan untuk memodelkan suatu sistem [20]. UML memungkinkan developer melakukan pemodelan secara visual, yaitu penekanan pada penggambaran, bukan didominasi oleh narasi. Pemodelan visual membantu untuk menagkap struktur dan kelakuan dari objek, mempermudah penggambaran interaksi antar elemen dalam sistem, dan mempertahankan konsistensi antara desain dan implementasi dalam pemrograman.

UML menyediakan beberapa notasi dan diagram standar yang bisa digunakan sebagai alat komunikasi bagi para pelaku dalam proses analisis dan desain. Diagram dalam UML didefinisikan sebagai informasi dalam berbagai bentuk yang digunakan atau dihasilkan dalam proses pengembangan perangkat lunak. Diagram yang umum dipakai dalam OOAD adalah:

1. *Use Case Diagram*

Use case diagram menjelaskan apa yang dapat dilakukan oleh sistem yang akan dibangun dan siapa yang berinteraksi dengan sistem tersebut. *Use case diagram* menjadi dokumen kesepakatan antara *costumer*, pengguna, dan *developer*. *Use case diagram* pada umumnya tersusun dari elemen *actor*, *use case*, *dependency*, *generalization*, dan *association*. *Use case diagram* memberikan gambaran yang statis dari sistem yang sedang dibangun dan merupakan diagram dari proses analisis [20].

2. *Sequence Diagram*

Sequence diagram menjelaskan secara detil urutan proses yang dilakukan dalam sistem untuk mencapai tujuan dari *use case* seperti: interaksi yang terjadi antar *class*, operasi apa saja yang terlibat, urutan antar operasi, dan informasi yang diperlukan oleh masing-masing operasi. *Sequence diagram* biasanya tersusun dari elemen *object*, *interaction*, dan *message*. *Interaction* menghubungkan dua *object* dengan *message*-nya. Diagram ini menjelaskan aspek dinamis dari sistem yang sedang dibangun [20].

3. *Class Diagram*

Class diagram menunjukan hubungan antar *class* dalam sistem yang sedang dirancang dan bagaimana mereka saling berkolaborasi untuk mencapai satu tujuan. *Class* pada *class diagram* terdiri dari tiga bagian utama yaitu nama kelas, isi properti dari kelas dan metode yang ada pada kelas tersebut. *Class diagram* juga memiliki jenis-jenis hubungan seperti asosiatif, dependensi, agregasi, komposisi, spesifikasi, dan generalisasi. Hubungan ini digunakan untuk menggambarkan bagaimana hubungan dan interaksi yang terjadi antar *class*. Masing-masing komponen penyusun *class* memiliki hak akses seperti *public*, *private*, dan *protected* [19].

4. *Activity Diagram*

Activity Diagram menggambarkan berbagai alur aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing aliran berawal, decision yang mungkin terjadi, dan bagaiana mereka berakhir. *Activity Diagram* juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Sebuah aktivitas dapat direalisasikan oleh satu *use case* atau lebih. Aktivitas menggambarkan proses yang berjalan, sementara *use case* menggambarkan bagaimana aktor menggunakan sistem untuk melakukan aktivitas [20].