BAB 2
LANDASAN TEORI

2.1 Ekstraksi Informasi

Ekstraksi Informasi merupakan salah satu cara untuk mendapatkan informasi tertentu dari suatu teks pada dokumen. Ekstraksi Informasi adalah pencarian otomatis pada informasi yang terstruktur seperti entitas, hubungan antar entitas, dan atribut yang menggambarkan entitas dari sumber yang tidak terstruktur [2]. Contoh Ekstraksi Informasi adalah mengekstraksi info web, jurnal, makalah ilmiah, surat masuk dan lain-lain.

Sistem Ekstraksi Informasi dibuat untuk membantu pengguna dalam menemukan informasi yang dibutuhkan. Informasi yang dihasilkan dapat berupa data seseorang, data perusahaan, nama instansi, riwayat seseorang atau lainnya tergantung oleh kebutuhan pengguna. Sistem yang dibangun menggunakan aturan-aturan tertentu atau melalui bantuan kecerdasan buatan.

2.2 Kecerdasan Buatan

Kecerdasan buatan merupakan salah satu cabang ilmu komputer yang mempelajari bagaimana membuat sebuah mesin yang mempunyai kemampuan untuk belajar dan beradaptasi terhadap sesuatu [7]. Tujuan dari riset-riset kecerdasan buatan adalah bagaimana membuat sebuah mesin bisa berfikir sama halnya dengan manusia berfikir. Implementasi dari kecerdasan buatan adalah sistem pakar, pendukung keputusan, prediksi, diagnose penyakit dan lain-lain.

2.3 *Machine Learning*

Machine learning merupakan bentuk aplikasi dari kecerdasan buatan yang memiliki kemampuan untuk belajar otomatis dan berkembang dari pengalaman tanpa pemograman secara eksplisit [8]. Machine learning memiliki gagasan bahwa sistem dapat belajar dari data, mengidentifikasi pola dan membuat keputusan dengan sedikit intervensi dari manusia.

2.4 Teorema Bayes

Teorema bayes merupakan teorema yang mengacu pada konsep probabilitas bersyarat [9]. Metode ini merupakan pendekatan statistic untuk melakukan

inferensi induksi pada persoalan klasifikasi. Misalkan A dan B adalah kejadian dalam ruang sampel. Probabilitas bersyarat dalam persamaan (2.1).

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \quad \textbf{(2.1)}$$

Dimana P(A∩B) adalah probabilitas interaksi A dan B dan P(B) adalah probabilitas B. Demikian pula dengan $P(B|A) = \frac{P(A \cap B)}{P(B)}$, sehingga nilai P(A∩B)= $P(B|A)$ P(A). Nilai P(A∩B) kemudian disubtitusikan ke dalam persamaan (2.1), maka diperoleh persamaan (2.2).

$$P(A|B) = \frac{P(B|A)\, P(A)}{P(B)} \quad \textbf{(2.2)}$$

2.5 *Naïve Bayes Classifier*

Naïve Bayes Classifier adalah salah satu algoritma sederhana namun memiliki kemampuan kemampuan dan akurasi yang tinggi dan termasuk dalam metode *machine learning* [10]. Berdasarkan kompleksitasnya *Naïve Bayes* merupakan salah satu algoritma sederhana yang menerapkan aturan Bayes disertai beberapa keunggulan yaitu mudah diimplementasikan, data uji yang sedikit, efisien dan memiliki akurasi yang relative tinggi. Namun *Naïve Bayes* juga memiliki kekurangan seperti akurasi yang berkurang apabila atribut yang menjadi parameter dalam klasifikasi tidak independen [11].

Metode klasifikasi Naïve Bayes menggunakan konsep peluang dalam menentukan kelas dokumen. Metode ini menggunakan asumsi dalam sebuah dokumen kemunculan kata tidak mempengaruhi ketidakmunculan kata yang lain [11]. *Naïve Bayes* merupakan teknik prediksi berbasis probabilistik sederhana dengan asumsi indepedensi kuat pada penerapan teorema Bayes. *Naïve Bayes Classifier* mengasumsikan ada tidaknya suatu fitur tertentu pada kelas tidak mempengaruhi keberadaan fitur lainnya.

Pada dasarnya, model probabilitas untuk sebuah *classifier* adalah model peluang bersyarat.

$$P(C|F_1, \dots, F_n) \quad \textbf{(2.3)}$$

Dengan menggunakan Teorema Bayes maka:

$$P(C|F_1, \dots, F_n) = \frac{P(C)P(F_1,F_2,\dots,F_n|C)}{P(F_1,F_2,\dots,F_n)} \quad \textbf{(2.4)}$$

Dimana Variabel C merepresentasikan kelas sementara variable F_1 … Fn merepresentasikan karakteristik petunjuk yang dibutuhkan untuk melakukan klasifikasi. Maka rumus tersebut menjelaskan bahwa peluang masuknya sampel karakteristik tertentu dalam kelas C (posterior) adalah munculnya kelas C (sebelum masuknya sampel tersebut, atau disebut prior), dikali dengan peluang kemunculan karakterikstik sampel pada kelas C (*likelihood*), dibagi dengan peluang kemunculan karakteristik-karakteristik sampel secara global (*evidence*). Sederhananya rumus tersebut dapat ditulis sebagai berikut:

$$Posterior = \frac{PriorxLikelihood}{Evidence} \quad \textbf{(2.5)}$$

Nilai *Evidence* selalu tetap untuk setiap kelas pada satu sampel. Nilai dari posterior tersebut nantinya akan dibandingkan dengan nilai-nilai posterior kelas lainnya untuk menentukan ke kelas apa suatu sampel akan diklasifikasikan.Dalam penelitian ini untuk mengetahui peluang setiap kelas adalah dengan menggunakan rumus sebagai berikut.

$$P(K) = \frac{n(K)}{n(S)} \quad \textbf{(2.6)}$$

Dengan $n(K)$ adalah jumlah sebuah kelas dan $n(S)$ adalah jumlah sampel. Jadi apabila menghitung peluang kelas “nama” akan menjadi seperti berikut.

$$\text{P(nama)} = \frac{jumlah\ kelas\ nama}{jumlah\ kalimat} \quad \textbf{(2.7)}$$

Untuk menghitung peluang tiap fitur pada suatu kelas adalah menggunakan rumus sebagai berikut.

$$P(w_k|c_i) = \frac{nk+1}{n+|vokabulari|} \quad \textbf{(2.8)}$$

Dengan keterangan sebagai berikut:

nk adalah nilai kemunculan fitur pada kategori ci

n jumlah keseluruhan fitur pada kategori ci

|vokabulari| jumlah seluruh fitur

w_k adalah fitur (f1-f7)

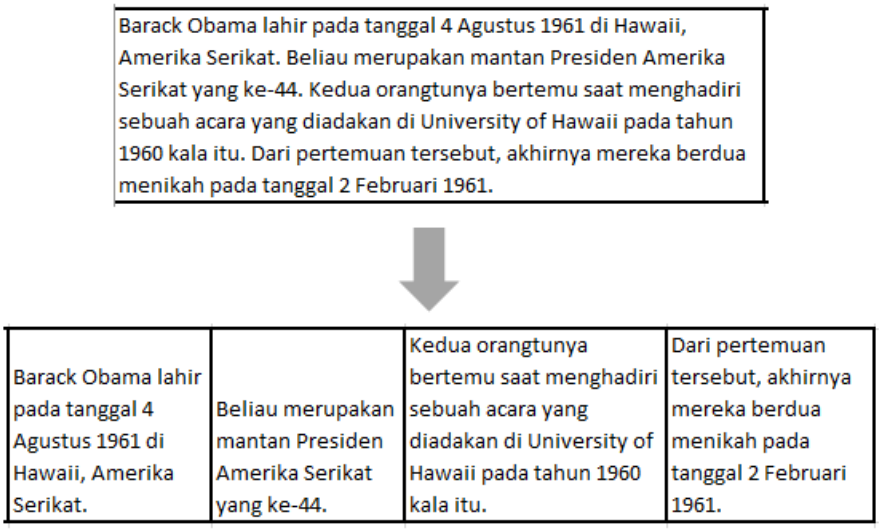
c_i adalah kategori kelas: nama, ttl, pendidikan, other

2.6 *Preprocessing text*

Preprocessing text adalah tahapan dimana dilakukannya penyeleksian data yang akan diproses. Tahap ini berfungsi menyiapkan data ke dalam bentuk yang sesuai dengan kebutuhan untuk proses mining lebih lanjut [12]. Dalam penelitian ini tahap *preprocessing* berupa *tokenizing*, *stopword removal*, dan *feature extraction*.

2.6.1 *Splitting*

Splitting merupakan proses pemecahan setiap paragraf teks menjadi kalimat-kalimat [13]. Proses tersebut dilakukan mengandalkan *delimiter* berupa tanda titik. Potongan kalimat paragraf digunakan untuk proses selanjutnya. Contoh proses tokenisasi kalimat dapat dilihat pada Gambar 2.1.



Gambar 2.1 *Splitting* kalimat

2.6.2 *Stopword Removal*

Stopword removal adalah proses penghapusan kata-kata yang tidak memiliki makna penting dalam tahap penelitian [13]. Kata yang dihapus tidak akan merubah makna kalimat.

2.6.3 *Tokenization*

Tokenization atau tokenisasi adalah proses pemecahan tiap kalimat menjadi pecahan-pecahan kata atau token kata [11]. Token kata tersebut kemudian digunakan untuk proses lebih lanjut dalam sistem.

2.7 *Unified Modelling Language (UML)*

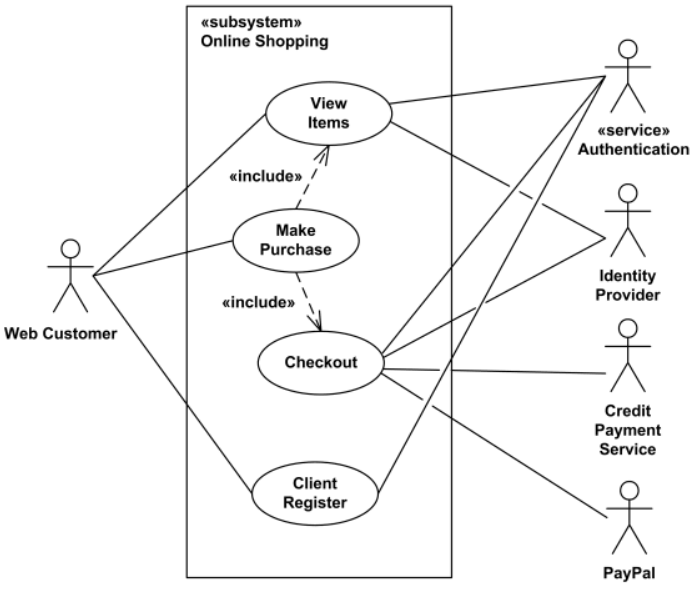
UML adalah ‘bahasa’ pemodelan untuk sistem atau perangkat lunak yang berparadigma ‘berorientasi objek’ [14]. Pemodelan digunakan untuk menyederhanakan masalah yang kompleks dengan tujuan agar lebih mudah dipelajari dan dipahami.

Pada UML terdapat beberapa diagram yang digunakan pada penelitian ini yaitu:

1. *Use case diagram*

Diagram pada UML yang menggambarkan interaksi antara sistem dan actor atau disebut juga sebagai pengguna, use case diagram juga dapat men-deskripsikan tipe interaksi antara si pemakai sistem dengan sistemnya.

Berikut adalah contoh dari *use case diagram* dapat dilihat pada Gambar 2.2.

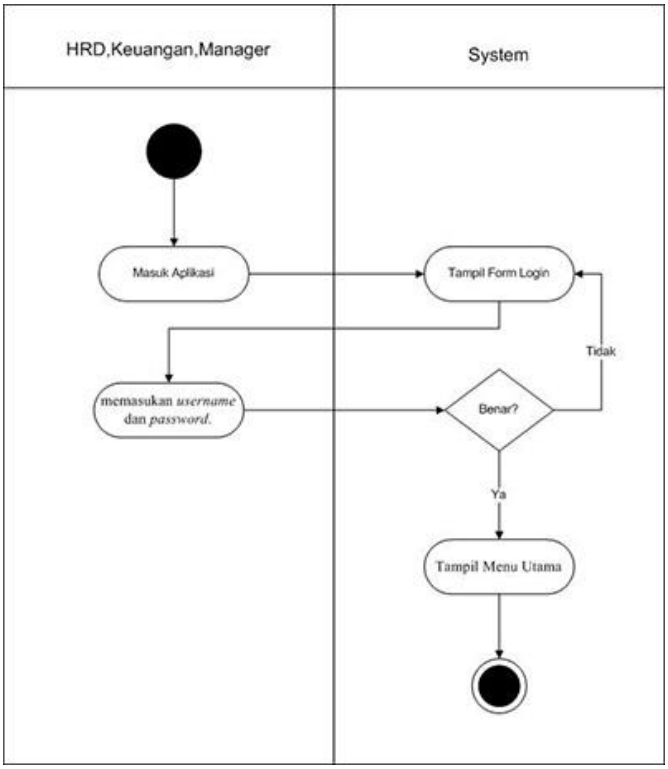


Gambar 2.2 Contoh *use case diagram*

2. *Activity diagram*

Activity diagram berfungsi untuk memodelkan proses-proses apa saja yang terjadi pada sistem. *activity diagram* dibuat berdasarkan jumlah *use case diagram* yang ada.

Berikut adalah contoh activity diagram dapat dilihat pada Gambar 2.3.

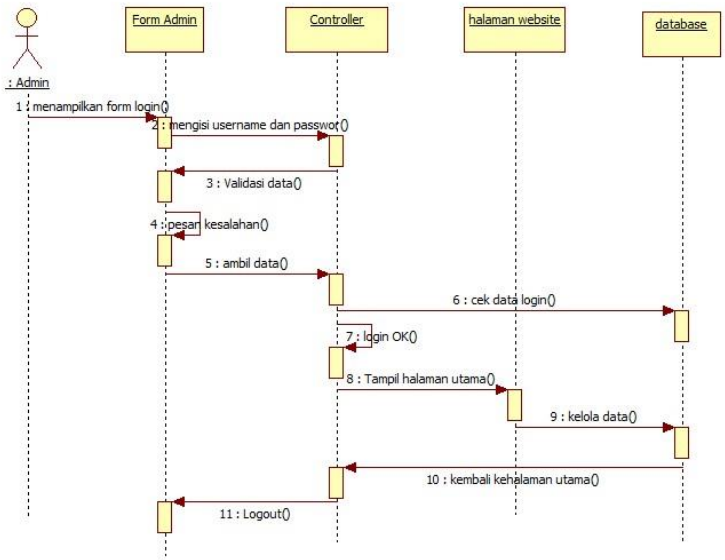


Gambar 2.3 Contoh *activity* diagram

3. *Sequence diagram*

Sequence diagram menjelaskan interaksi objek berdasarkan urutan waktu. Biasa menggambarkan urutan atau tahapan yang harus dilakukan untuk dapat menghasilkan sesuatu seperti pada *use case diagram*.

Berikut adalah contoh activity diagram dapat dilihat pada Gambar 2.4.

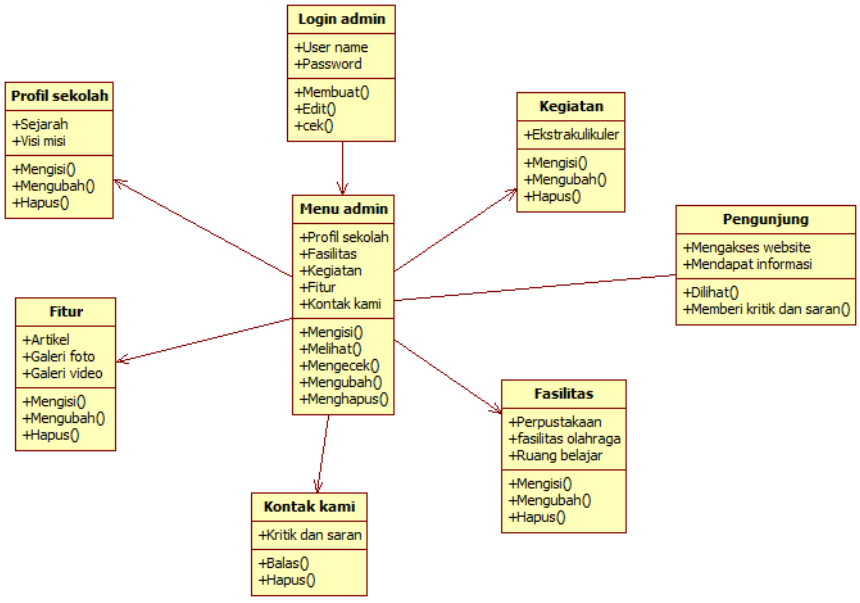


Gambar 2.4 Contoh *sequence* diagram

4. *Class diagram*

Class diagram digunakan untuk menampilkan kelas-kelas maupun paket-paket yang ada pada suatu sistem yang nantinya akan digunakan. Diagram ini dapat memberikan gambaran mengenai sistem maupun relasi yang ada pada suatu sistem.

Berikut adalah contoh class diagram dapat dilihat pada Gambar 2.5.



Gambar 2.5 Contoh *class diagram*

2.8 *Hyper Text Markup Language (*HTML*)*

HTML adalah Bahasa pemformatan teks untuk dokumen-dokumen pada jaringan komputer yang sering disebut sebagai world wide web [15]. HTML merupakan Bahasa pemrograman yang fleksibel dan dapat digabungkan dengan Bahasa pemrograman lain seperti PHP, ASP, JSP, JavaScript. Beberapa tag dalam dokumen-dokumen HTML menentukan bagaimana teks diformat sedangkan tag yang lain memberitahukan komputer bagaimana menanggapi aksi-aksi yang datang dari pengguna.

2.9 *Beautifulsoup4*

Beautifulsoup4 adalah *library python* untuk menarik data dari sebuah file html dan xml. Itu bekerja dengan *parser* untuk menyediakan jalur navigasi pencarian, dan modifikasi pohon parse [16]. Kemampuan yang dimiliki oleh *library* ini macam-macam, salah satunya adalah mengekstrak teks dari sebuah file.

2.10 *Synthetic Minority Oversampling Techniques* (SMOTE)

SMOTE adalah suatu teknik untuk menghadapi ketidakseimbangan kelas pada data set. SMOTE adalah turunan dari *oversampling.* SMOTE diperkenalkan pertama kali oleh Nithes V Chawla. Teknik ini menggunakan replikasi data sintesis dari kelas minoritas [17]. SMOTE bekerja dengan mencari data minor kemudian dibuat data sintesis duplikasi sebanyak data lainnya untuk disama-ratakan.

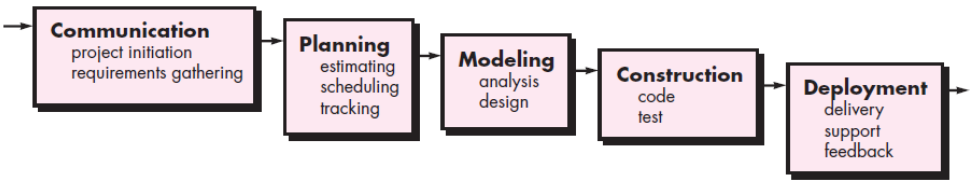
2.11 *Python*

Python dikembangkan oleh Guido van Rossum pada tahun 1990 di CWI, Amsterdam sebagai kelanjutan dari bahasa pemrograman ABC. *Python* adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. *Python* merupakan salah satu bahasa pemrograman yang populer di dunia kerja Indonesia.

Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan dengan sitaks kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Pada umumnya *Python* mendukung multi paradigma pemrograman, seperti pemrograman berorientasi objek, pemrograman imperatif dan pemrograman fungsional [18].

2.12 Model *Waterfall*

Waterfall model adalah model klasik yang digunakan sebagai model dalam membangun perangkat lunak. Model *waterfall* bersifat sistematis dan berurutan dalam membangun perangkat lunak [6]. Model ini terdiri dari 5 tahapan untuk pengembangan. Berikut adalah penjelasan tahap-tahap yang dilakukan dalam model ini.



Gambar 2.6 Model *Waterfall Pressman*

1. *Communication (Project Initiation & Requirements Gathering)*

Sebelum memulai pekerjaan yang bersifat teknis, sangat diperlukan adanya komunikasi dengan customer demi memahami dan mencapai tujuan yang ingin dicapai. Hasil dari komunikasi tersebut adalah inisialisasi proyek seperti menganalisis permasalahan yang dihadapi dan mengumpulkan data-data yang diperlukan, serta membantu mendefinisikan fitur dan fungsi dari aplikasi. Pengumpulan data-data tambahan bisa juga diambil dari jurnal, artikel, paper dan internet.

2. *Planning (Estimating, Scheduling, Tracking)*

Tahap berikutnya adalah tahapan perencanaan yang menjelaskan tentang estimasi tugas-tugas teknis yang akan dilakukan, resiko-resiko yang dapat terjadi, sumber daya yang diperlukan dalam membuat sistem, produk kerja yang ingin

dihasilkan, penjadwalan kerja yang akan dilaksanakan, dan tracking proses pengerjaan sistem.

3. *Modeling (Analysis & Design)*

Tahapan ini adalah tahap perancangan dan pemodelan arsitektur sistem yang berfokus pada perancangan struktur data, arsitektur software, tampilan interface, dan algoritma program. Tujuannya untuk lebih memahami gambaran besar dari apa yang akan dikerjakan.

4. *Construction (Code & Test)*

Tahapan ini merupakan proses penerjemahan bentuk desain menjadi kode atau bentuk bahasa yang dapat dibaca oleh mesin. Setelah pengkodean selesai, dilakukan pengujian terhadap sistem dan juga kode yang sudah dibuat. Tujuannya untuk menemukan kesalahan yang mungkin terjadi untuk nantinya diperbaiki.

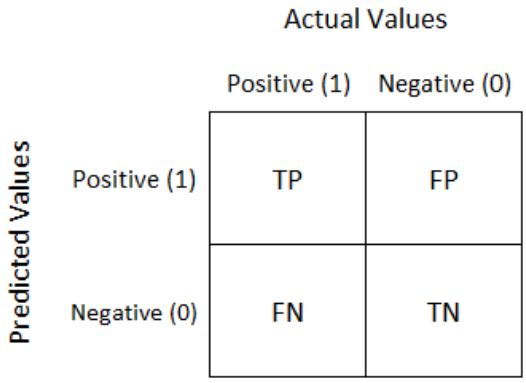
5. *Deployment (Delivery, Support, Feedback)*

Tahapan terakhir ini merupakan tahapan implementasi software ke customer, perbaikan software, evaluasi software, dan pengembangan software berdasarkan umpan balik yang diberikan agar sistem dapat tetap berjalan dan berkembang sesuai dengan fungsinya.

2.13 Pengujian Performansi

Pengujian performansi didasarkan pada penelitian Xhemali [3] yaitu menguji menggunakan standar metrik *accuracy*, *precision*, dan *recall.* Aspek tersebut diuji menggunakan tabel prediktif klasifikasi yang disebut juga tabel *confusion matrix.*

Berikut adalah contoh dari tabel *confusion matrix* dapat dilihat pada Gambar 2.7.



Gambar 2.7 Contoh Tabel *Confusion Matrix*

Dengan menggunakan persamaan sebagai berikut:

$$Recall = \frac{TP}{TP+FN} \quad \textbf{(2.9)}$$

$$Precision = \frac{TP}{TP+FP} \quad \textbf{(2.10)}$$

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN} \quad \textbf{(2.11)}$$

$$\text{F-measure} = \frac{2(recall\ x\ precision)}{recall+precision} \quad \textbf{(2.12)}$$

Dengan keterangan berikut:

TP (*True Positive*) adalah kondisi nilai actual dan predicted sama-sama bernilai positif.

TN (*True Negative*) adalah kondisi nilai actual dan predicted sama-sama bernilai negatif.

FN (*False Negative*) adalah kondisi nilai actual bernilai positif, tetapi nilai predicted bernilai negatif.

FP (*False Positive*) adalah kondisi nilai actual bernilai negatif, tetapi nilai predicted bernilai positif.