BAB 2

LANDASAN TEORI

2.1 Ekstraksi Informasi

Ekstraksi informasi adalah pencarian otomatis pada informasi yang terstruktur seperti entitas, hubungan antar entitas, dan atribut yang menggambarkan entitas dari sumber yang tidak terstruktur [4]. Pencarian dari suatu ekstraksi informasi merupakan sebuah sistem yang berfungsi untuk mencari data spesifik dalam Natural Language Text. Dalam sebuah template formulir atau tabel serta komponen – komponen teks yang ada diambil datanya untuk diekstraksi menjadi data yang memiliki kegunaan [5].

2.2 Optical Character Recognition (OCR)

Optical character recognition (OCR) adalah sebuah sistem komputer yang dapat membaca huruf, baik yang berasal dari sebuah pencetak (printer atau mesin ketik) maupun yang berasal dari tulisan tangan. OCR adalah aplikasi yang menerjemahkan gambar karakter (image character) menjadi bentuk teks dengan cara menyesuaikan pola karakter per baris dengan pola yang telah tersimpan dalam database aplikasi. Hasil dari proses OCR adalah berupa teks sesuai dengan gambar output scanner dimana tingkat keakuratan penerjemahan karakter tergantung dari tingkat kejelasan gambar dan metode yang digunakan [6]. Dalam penelitian ini aplikasi OCR yang akan digunakan adalah FreeOCR.

2.3 Metode Preprocessing

Preprocessing merupakan suatu tahapan dimana data asli akan diolah untuk dipersiapkan sebagai bahan olahan untuk tahap selanjutnya. Data masukan awal yaitu berupa dokumen. Tahap *Preprocessing* yang akan dilakukan dalam penelitian ini adalah tokenisasi kalimat, tokenisasi kata, *filtering*, *case folding* dan *term weighting* atau pembobotan dengan menggunakan TF-ICF.

2.3.1 Tokenisasi

Proses tokenisasi kata adalah tahap pengecekan dari karakter pertama sampai dengan karakter terakhir. Apabila ditemukan karakter pemenggal atau *delimeter* seperti spasi atau tanda baca, maka teks akan dipisahkan dan dimasukkan dalam kumpulan kalimat [7]. Contoh dari proses ini dapat dilihat pada tabel 2.1 :

Tabel 2.1 Contoh Tokenisasi Kata

Teks Input	Hasil Tokenisasi kata
Universitas Komputer Indonesia	Universitas Komputer Indonesia

2.3.2 *Filtering*

Filtering mengacu pada proses memutuskan istilah mana yang harus digunakan untuk merepresentasikan dokumen sehingga dapat digunakan untuk menggambarkan isi dokumen dan membedakan dokumen dengan dokumen lain yang ada pada koleksi [8]. Jadi *Filtering* adalah proses dimana teks selain karakter “a” sampai “z”, simbol “-“ dan spasi akan dihilangkan dan hanya menerima spasi. Tahapan yang dilakukan dalam proses ini adalah :

1. Membaca data masukan berupa teks/dokumen
2. Mengecek apakah ada huruf selain “a” sampai “z”, simbol “-“ dan “ “ (spasi). Apabila tidak ditemukan maka algoritma berhenti, Jika ditemukan;
3. Hapus karakter yang bukan merupakan karakter “a” sampai “z”, simbol “-“ dan “ “ (spasi).
4. Algoritma Selesai.

2.3.3 Case Folding

Case folding adalah tahapan pemrosesan teks dimana semua teks diubah ke dalam *case* yang sama [8]. Dalam penelitian ini semua huruf akan disamakan menjadi huruf kecil. Contoh dari proses dapat dilihat pada tabel 2.2 :

Tabel 2.2 Contoh *Case Folding*

Teks Input	Hasil *Case Folding*
Universitas Komputer Indonesia	universitas komputer indonesia

2.3.4 *Term Wighting* / TF-ICF

Tahap *Term weighting/ TF-ICF* adalah metode yang digunakan dalam menghitung bobot setiap kata. Metode yang sering digunakan yaitu menggunakan metode TF-ICF (*Term Frequency - Inverse Class Frequency*). Pada metode ini, perhitungan bobot *term*(kata) dalam sebuah dokumen dilakukan dengan menghitung nilai *Term Frequency* (TF) dan *Inverse Class Frequency* (ICF) degan cara mengalikannya [9].

Term Frequency (TF) adalah faktor yang menentukan bobot *term* pada suatu dokumen berdasarkan jumlah kemunculannya dalam dokumen tersebut. Nilai jumlah kemunculan suatu kata (*term frequency*) diperhitungkan dalam pemberian bobot terhadap suatu kata. Semakin besar jumlah kemunculan suatu *term* dalam dokumen, semakin besar pula bobotnya dalam dokumen atau akan memberikan nilai kesesuian yang semakin besar.

Inverse Class Frequency (ICF) adalah pengurangan dominansi *term* yang sering muncul di berbagai kelas. Hal ini diperlukan karena *term* yang banyak muncul di berbagai kelas, dapat dianggap sebagai *term* umum (*common term*) sehingga tidak penting nilainya. Sebaliknya faktor jarang munculnya suatu kata (*term scarcity*) dalam koleksi kelas harus diperhatikan dalam pemberian bobot. Kata yang muncul pada sedikit kelas harus dipandang sebagai kata yang lebih penting (*uncommon term*) daripada kata yang muncul pada banyak kelas. Pembobotan akan memperhitungkan faktor kebalikan frekuensi kelas yang mengandung suatu kata (*inverse class frequency*). Hal ini merupakan usulan dari George Zipf. Zipf mengamati bahwa frekuensi dari sesuatu cenderung kebalikan secara proposional dengan urutannya [9].

Nilai ICF sebuah kata (*term*) dihitung menggunakan persamaan (2.3) berikut :

$$ICF_t = \log (N/cf) \quad (2.3)$$

Untuk menghitung bobot kata (W_i) dalam dokumen dihitung dengan menggunakan persamaan (2.4) berikut:

$$W_{dt} = tf_{dt} * ICF_t \quad (2.4)$$

Dimana :

W = bobot dokumen ke-d terhadap kata ke-t

d = kelas ke-d

t = kata ke-t

tf = banyaknya kata yang dicari pada sebuah dokumen

N = total dokumen

df = banyak dokumen yang mengandung tiap kata

2.4 *K-Nearest Neighbours*

Algoritma *k*-NN (*K-Nearest Neighbors)* adalah algoritma supervised learning yang melakukan klasifikasi terhadap objek berdasarkan yang jaraknya paling dekat dengan objek tersebut. Tujuan dari algoritma *k*-NN adalah mengklasifikasikan objek baru berdasarkan atribut dan training samples. Dimana hasil dari sampel uji yang baru diklasifikasikan berdasarkan mayoritas dari kategori pada *k-NN*. Proses pengklasifikasian *k*-NN didasarkan pada memori. Algoritma *k*-NN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari sample uji yang baru. Jarak digunakan pada *k*-NN adalah jarak Euclidean yang sering digunakan pada data numeric [10].

Pada fase training, algoritma ini hanya melakukan penyimpanan vektor-vektor fitur dan klasifikasi data training sample. Pada fase klasifikasi, fitur – fitur yang sama dihitung untuk testing data (klasifikasinya belum diketahui). Jarak dari vektor yang baru ini terhadap seluruh vektor training sample dihitung, dan sejumlah k buah yang paling dekat diambil. Titik yang baru klasifikasinya diprediksikan termasuk pada klasifikasi terbanyak dari titik – titik tersebut [10].

Nilai k yang terbaik untuk algoritma ini tergantung pada data; secara umumnya, nilai k yang tinggi akan mengurangi efek noise pada klasifikasi, tetapi membuat batasan antara setiap klasifikasi menjadi lebih kabur. Nilai k yang bagus dapat dipilih dengan optimasi parameter, misalnya dengan menggunakan cross-validation. Kasus khusus di mana klasifikasi diprediksikan berdasarkan data pembelajaran yang paling dekat (dengan kata lain, k = 1) disebut algoritma nearest neighbor [10].

Ketepatan algoritma k-NN ini sangat dipengaruhi oleh ada atau tidaknya fitur-fitur yang tidak relevan, atau jika bobot fitur tersebut tidak setara dengan relevansinya terhadap klasifikasi. Riset terhadap algoritma ini sebagian besar membahas bagaimana memilih dan memberi bobot terhadap fitur, agar performa klasifikasi menjadi lebih baik [10].

K buah data learning terdekat akan melakukan voting untuk menentukan label mayoritas. Label data query akan ditentukan berdasarkan label mayoritas dan jika ada lebih dari satu label mayoritas maka label data query dapat dipilih secara acak di antara label-label mayoritas yang ada [10].

Pada proses pengklasifikasian, algoritma ini tidak menggunakan model apapun untuk dicocokkan dan hanya berdasarkan pada memori. Algoritma k-NN menggunakan klasifikasi ketetanggaan sebagai nilai prediksi dari sampel uji yang baru. Jarak yang digunakan adalah jarak Euclidean Distance. Jarak Euclidean adalah jarak yang paling umum digunakan pada data numerik. Euclidean distance didefinisikan sebagai berikut :

$$d(x_i, y_i) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (2.5)$$

Keterangan

$d(xi,$yi) = Jarak Euclidean

(x_i) = bobot data latih ke-i

(yi) = bobot data uji ke-i

Adapun langkah-langkah perhitungan *K-Nearest Neighbors* adalah sebagai berikut :

1. Menghitung parameter K (jumlah tetangga yang paling dekat).

2. Menghitung kuadrat jarak Euclid masing-masing objek terhadap data sampel yang diberikan.
3. Lalu mengurutkan objek-objek tersebut ke dalam kelompok jarak *euclidean* yang paling kecil.
4. Mengumpulkan kategori klasifikasi *K-nearest neighbors*.
5. Dengan melakukan proses perhitungan kategori klasifikasi neighbors yang paling banyak maka dapat memprediksi jarak *Euclidean*.

2.5 UML (*Unified Modeling Language*)

Pendekatan pembangunan perangkat lunak yang digunakan dalam penelitian ini adalah pendekatan beorientasi objek menggunakan unified modeling language. Unified Modeling Language (UML) adalah bahasa spesifikasi standar untuk mendokumentasikan, menspesifikasikan, dan membangun sistem perangkat lunak. Unified Modeling Language (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek (OOP) serta aplikasinya. UML adalah metodologi untuk mengembangkan sistem OOP dan sekelompok perangkat tool untuk mendukung pengembangan sistem tersebut. UML merupakan dasar bagi perangkat (tool) desain berorientasi objek dari IBM. UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Penggunaan UML dalam industri terus meningkat. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri peranti lunak dan pengembangan sistem,Dalam Penelitian ini UML yang digunakan sebagai berikut [11]:

2.5.1 *Use Case* Diagram

Use Case Diagram untuk memodelkan proses bisnis. Dalam konteks UML, tahap konseptualisasi dilakukan dengan pembuatan use case diagram yang sesungguhnya merupakan deskripsi peringkat tinggi bagaimana perangkat lunak (aplikasi) akan digunakan oleh penggunanya. Selanjutnya, use case diagram tidak hanya sangat penting pada analisis, tetapi juga sangat penting untuk perancangan

(design), untuk mencari (mencoba menemukan) kelas-kelas yang terlibat dalam aplikasi, dan untuk melakukan pengujian (testing). Membuat use case diagram yang komprehensif merupakan hal yang sangat penting dilakukan pada tahap analisis. Dengan menggunakan use case diagram, akan mendapatkan banyak informasi yang sangat penting yang berkaitan dengan aturan –aturan bisnis. Dalam hal ini, setiap objek yang berinteraksi dengan sistem/perangka lunak (misalnya, orang, suatu perangkat keras, sistem lain, dan sebagainya) merupakan actor untuk sistem/perangkat lunak yang dibangun, sementara use case merupakan deskripsi lengkap tentang bagaimana sistem/perangkat lunak berprilaku untuk para actor-nya. Dengan demikian, use case program merupakan deskripsi lengkap tentang interaksi yang terjadi antara para actor dengan sistem/perangkat lunak yang sedang dibangun. Ketika mengembangkan use case diagram, hal yang pertama dilakukan adalah mengenali actor untuk sistem/aplikasi yang sedang dikembangkan. Dalam hal ini, ada beberapa karakteristik untuk para actor, yaitu actor ada diluar sistem yang sedang dikembangkan dan actor verinteraksi dengan sistem yang sedang dikembangkan.

2.5.2 *Activity* Diagram

Activity diagram menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir. Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi. Activity diagram merupakan state diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (internal processing). Oleh karena itu activity diagram tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum. 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. Use case diagram merupakan gambaran menyeluruh dan pada umumnya

sangatlah tidak terperinci. Oleh karena itu, harus diperinci perilaku sistem untuk masing-masing use case yang ada. Penggunaan activity diagram dapat memberikan gambaran secara meyeluruh terhadap perangkat lunak yang dibangun.

2.5.3 *Class* Diagram

Class diagram merupakan diagram yang selalu ada di permodelan sistem berorientasi objek. Class diagram menunjukkan hubungan antar class dalam sistem yang sedang dibangun dan bagaimana mereka saling berkolaborasi untuk mencapai suatu tujuan. classs diagram menggambarkan interaksi dan relasi antar kelas yang ada di dalam suatu sistem. Kelas memiliki atribut dan metode. Atribut merupakan variabel-variabel yang dimiliki oleh suatu kelas. Metode adalah fungsi-fungsi yang dimiliki oleh suatu kelas. Atribut dan metode dapat memiliki salah satu sifat sebagai berikut :

1. Private, tidak dapat dipanggil dari luar kelas yang bersangkutan.
2. Protected, hanya dapat dipanggil oleh kelas yang bersangkutan dan anak -anak yang mewarisinya.
3. Public, dapat dipanggil oleh siapa saja.

Class diagram menggambarkan relasi atau hubungan antar kelas dari sebuah sistem. Berikut ini beberapa gambaran relasi yang ada dalam class diagram:

1. Assiciation

   Hubungan antar class yang statis. Class yang mempunyai relasi asosiasi menggunakan class lain sebagai atribut pada dirinya.
2. Aggregation

   Relasi yang membuat class yang saling terikat satu sama lain namun tidak terlalu berkegantungan.
3. Composition

   Relasi agregasi dengan mengikat satu sama lain dengan ikatan yang sangat kuat dan saling berkegantungan.
4. Dependency

Hubungan antar class dimana class yang memiliki relasi dependency menggunakan class lain sebagai atribut pada method.

5. Realization

   Hubungan antar class dimana sebuah class memiliki keharusan untuk mengikuti aturan yang ditetapkan class lainnya.

2.5.4 *Sequence* Diagram

Diagram sekuen menggambarkan kelakuan/perilaku objek pada use case dengan mendeskripsikan waktu hidup objek dan message yang dikirimkan dan diterima antar objek. Objek-objek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut. Oleh karena itu untuk menggambar diagram sekuen maka harus diketahui objek-objek yang terlibat dalam sebuah use case beserta metode-metode yang dimiliki kelas yang diinstansiasi menjadi objek itu.

2.6 Confusion Matrix

Confusion matrix merupakan sebuah metode perhitungan yang digunakan untuk mencari keakuratan pada hasil klasifikasi [12]. Berikut merupakan tabel 2.3 *confusion matrix*:

Tabel 2. 3 *Confusion Matrix*

Kelas		Prediksi	
		1	0
Kelas sebenarnya	1	TP	FN
	0	FP	TN

Keterangan:

a. *True Positive* (TP), merupakan jumlah dokumen dari kelas 1 yang benar diklasifikasikan sebagai kelas 1.
b. *False Positive* (FP), merupakan jumlah dokumen dari kelas 0 yang salah diklasifikasikan sebagai kelas 1.

c. *False Negative* (FN), merupakan jumlah dokumen dari kelas 1 yang salah diklasifikasikan sebagai kelas 0.
d. *True Negative* (TN), merupakan jumlah dokumen dari kelas 0 yang benar diklasifikasikan sebagai kelas 0.

Kemudian akan dihitung akurasinya, akurasi merupakan perhitungan antara jumlah prediksi benar dan dibagikan dengan jumlah prediksi, untuk menghitung akurasi digunakan rumus sebagai berikut:

$$akurasi = \frac{jumlah\ prediksi\ benar}{jumlah\ seluruh\ prediksi\ yang\ dilakukan} \times 100\ \%$$