BAB 2
LANDASAN TEORI

2.1 Information Retrieval

Information Retrieval merupakan suatu sistem yang digunakan untuk mempertemukan deskriptor yang mewakili permintaan dengan deskriptor yang mewakili dokumen [7]. *Information Retrieval* menangani proses yang berkaitan dengan representasi dokumen penyimpanan, penelusuran dan penemuan informasi yan relevan terhadap suatu sistem yang dapat menjembatani informasi dengan pengguna yang membutuhkan informasi [7]. Saat melakukan penelusuran pengguna menginput permintaan ke sistem. Permintaan ini sama halnya dengan proses pengindeksan atau analisis isi informasi dari suatu dokumen yang sebelum data direkam pangkal data dalam proses pengindeksan, dokumen dianalisis isinya untuk mengetahui topik atau pokok bahasan. Dokumen untuk diindeks setelah itu topik dan pokok bahasan diungkapkan dalam bentuk kata kunci atau istilah yang mewakili konsep - konsep penting dalam dokumen.

2.2 Preprocessing

Tahap yang mencakup semua rutinitas, dan proses untuk mempersiapkan data yang akan digunakan pada operasi *knowledge discovery sistem text mining* [2]. Tujuan *Preprocessing* tersebut adalah untuk mempermudah dalam proses *Minning.* Pada penelitian ini tahap *Preprocessing* yang dilakukan adalah *Case Folding*, *Cleaning* Data, *Tokenizing*, *Filtering* dan *Stemming*, *Stemming* menggunakan Algoritma Nazief dan Andriani, yaitu algoritma yang dikembangkan untuk stemming dokumen Bahasa Indonesia. Yaitu suatu proses pemotongan imbuhan-imbuhan suatu kata menjadi kata dasarnya yang merupakan suatu tahap penting dalam *Information Retrieval* [1] dan *Filtering merupakan* tahap pengambilan kata-kata yang penting dari hasil *Token*izing [8].

2.3 Pembobotan

Pembobotan dilakukan untuk mendapatkan nilai dari kata/term yang berhasil diekstrak. Pada skripsi ini, menggunakan metode TF-IDF sebagai proses pembobotan. Pada tahap ini, setiap dokumen diwujudkan sebagai sebuah vektor dengan elemen sebanyak term yang berhasil dikenali dari tahap ekstraksi dokumen diatas. Vektor tersebut beranggotakan bobot dari setiap term yang dihitung berdasarkan metode TF-IDF.

Metode TF-IDF merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada *information retrieval* [3]. Perhitungan bobot dengan *Term Frequency – Inverse Document Frequency* (TF-IDF) menggunakan kombinasi dari dua nilai, yaitu frekuensi kata dan inverse frekuensi dokumen yang didapat dari membagi jumlah dokumen secara keseluruan dengan jumlah dokumen dimana kata tersebut muncul [3]. Rumus TF-IDF yang digunakan adalah persamaan 2.1 untuk mendapatkan Wdt dan untuk mendapatkan Idf .

$$IDF = \log(\frac{n}{df}) \qquad (2.1)$$

$$W_{dt} = tf_{df} * IDF_t \qquad (2.2)$$

Deskripsi :

n : banyak dokumen koleksi termasuk dokumen testing

d : dokumen ke-d

t : kata ke-t dari kata kunci

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

tf : banyaknya kata yang dicari pada sebuah dokumen

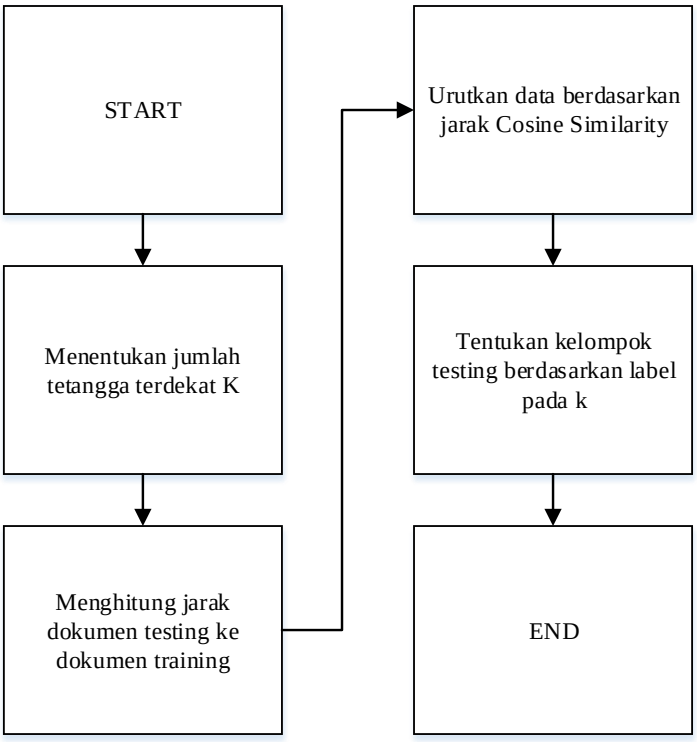
IDF : Inversed Document Frequency Nilai IDF didapatkan dari IDF : log2 (D/df) dimana

D : total dokumen

df : banyak dokumen yang mengandung kata yang dicari

2.4 Algoritma K-*Nearest Neighbor*

K-Nearest Neighbor (KNN) merupakan salah satu metode *machine learning* yang melakukan klasifikasi terhadap objek berdasarkan data pembelajaran yang jaraknya paling dekat dengan objek tersebut. Metode ini sangat sederhana, mudah direpresentasikan, memiliki ketangguhan terhadap training data yang memiliki banyak *noise*, dan cukup efektif untuk proses pengelompokan [10]. Cara kerja dari KNN perlu adanya penetuan inputan berupa data latih, data uji dan nilai k. Kemudian mengurutkan data latih berdasarkan kedekatan jaraknya berdasarkan hitungan dari jarak data yang diuji dengan data latih. Setelah itu diambil dari k data latih teratas untuk menentukan kelas klasifikasi untuk kelas yang dominan dari k data latih yang diambil [10].



Gambar 2. 1 Alur Algoritma K-*Nearest Neighbor*

2.5 *Cosine Similarty*

Cosine similarity adalah ukuran kesamaan yang lebih umum digunakan dalam *information retrieval* dan merupakan ukuran sudut antara vektor dokumen dtest (titik (i,k)) dan dtraining (titik (j,k)) . Tiap vektor tersebut merepresentasikan setiap kata dalam setiap dokumen (teks) . Rumus *Cosine similarity* yang digunakan adalah persamaan 2.3 [10].

$$\cos(\Theta_{ij}) = \frac{\sum_k (d_{ik} d_{jk})}{\sqrt{\sum_k d_{ik}^2} \sqrt{\sum_k d_{jk}^2}} \quad (2.3)$$

2.6 OOP (Obejct Oriented Programming)

Object Oriented Programming (OOP) adalah suatu metode pemrograman yang berbasiskan pada objek, secara singkat pengertian dari OOP adalah koleksi objek yang saling berinteraksi dan saling memberikan informasi satu dengan yang lainnya . Pemogramman berorientasi objek (OOP) merupakan paradigma pemograman yang menganggap semua benda adalah objek, yaitu menganggap segala sesuatu dalam bahasa pemograman adalah objek, bahkan *variable* pun dianggap sebagai suatu objek. Konsep dasar dari pemrograman berorientasi objek menekankan kepada konsep: Objek (*Object*) "Objek merupakan kumpulan variabel dan fungsi yang saling berkaitan" [2].

1. Kelas (*Class*) "*Class* merupakan suatu rancangan dari objek itu sendiri. Jadi satu *class* dapat berisi banyak objek."
2. Pembungkusan (*Encapsulation*) "*Encapsulation* merupakan cara menyembunyikan detail informasi dan fungsionalitas yang ada pada suatu kelas."
3. Pewarisan (*Inheritance*) "*Inheritance* merupakan kemampuan untuk meng-*extends class* yang akan mewarisi semua karakter dari *partner class*-nya."

4. Polimorfisme (*Pholymorphism*) "*Pholymorphism* merupakan kemampuan objek-objek yang berbeda kelas, tetapi terkait dalam perwarisan untuk merespons secara berbeda terhadap suatu pesan yang sama.

2.7 Database

Database adalah koleksi dari infromasi (data) dalam bentuk machine readable yang dibuat dengan menggunakan computer [2]. Database merupakan aspek yang sangat penting dalam system informasi dimana basis data merupakan gudang penyimpanan data yang akan diolah lebih lanjut. Basis data menjadi penting karena dapat menghindari duplikasi data, hubungan antar data yang tidak jelas, organisasi data, dan juga update yang rumit [2]. Database disimpan secara sistematis didalam komputer dan dimanipulasi menggunakan perangkat lunak (program aplikasi) untuk menghasilkan informasi. Dalam aplikasi yang buat, maka menggunakan database yang disimpan dalam bentuk *flat file*.

2.8 Unified Modelling Language (UML)

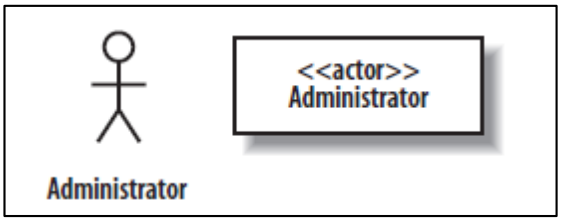
UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem informasi [3] . UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson. Namun demikian UML dapat digunakan 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. Proses pemodelan data menggambarkan keseluruhan proses bisnis yang akan dilakukan oleh sebuah sistem informasi. Proses model juga menjelaskan data-data yang terlibat dalam proses tersebut. Salah satu pemodelan data adalah dengan UML (Unified Modelling Language).

2.8.1 Usecase Diagram

Usecase adalah kasus atau situasi dimana sistem digunakan untuk memenuhi salah satu atau lebih dari kebutuhan penguna. Usecase menangkap sepotong fungsionalitas yang sistem sediakan. Usecase merupakan jantung dari sebuah model, karena usecase mempengaruhi dan membimbing semua elemen lain dalam sistem [3]. Usecase Diagram merupakan teknik untuk merekam persyaratan fungsional sebuah sistem. Use case mendeskripsikan interaksi tipikal antara para pengguna sistem dengan sistem itu sendiri, dengan memberi sebuah narasi tentang bagaimana sistem tersebut digunakan.

a. Aktor

Aktor pada sebuah UML dinotasikan atau digambarkan dengan karakter stickman dan memiliki label yang memiliki nama contoh Aktor pada gambar 2.2 [3]



Gambar 2. 2 Aktor

b. Usecase

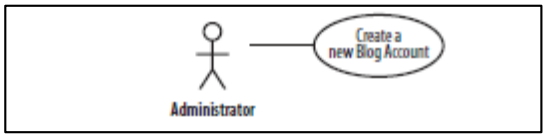
Sebuah u*secase* atau *Action* digambarkan dengan bentuk oval dan ada sebuah aktifitas yang dilakukan dan memiliki ruang lingkup yang digambarkan den persegi pada gambar 2.3 digambarkan melakukan *Action* membuat *Account* baru untuk blog [3] :



Gambar 2. 3 Usecase.

c. *Communication line*

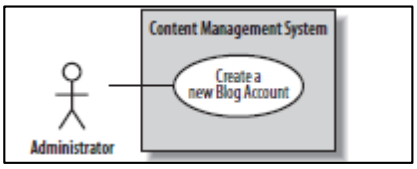
Sebuah *communication line* adalah media penghubung antara Aktor dengan usecase . *communication line* menunjukan aktfitas aktor dalam sebuah *usecase* [3]:



Gambar 2. 4 Communication Line.

d. *System boundaries*

System Boundaries adalah lingkungan sebuah sistem tertentu yang diperuntuhkan untuk user tertentu . Untuk menampilkan system's boundary pada usecase diagram, gambar sebuah kotak disekitar semua usecase tapi aktor tetap berada diluar kotak [3]. Contoh digambar 2.5 sebagai berikut :



Gambar 2. 5 System Boundaries.

2.8.2 *Activity* Diagram

Usecase menggambarkan perilaku sistem. kelas menggambarkan berbagai jenis benda-benda yang dibutuhkan dalam sistem. Diagram aktivitas menunjukkan proses bisnis dari sistem yang sedang berjalan [3].

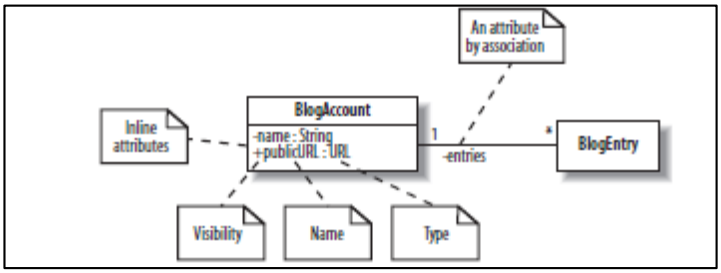
2.8.3 Class Diagram

Class diagram adalah diagram yang menggambarkan struktur sistem dari segi pendefinisian kelas-kelas yang akan dibuat untuk membangun sistem. kelas memiliki 3 bagian utama yaitu attribute, operation, dan name. kelas-kelas yang ada pada struktur sistem harus dapat melakukan fungsi-fungsi sesuai dengan kebutuhan sistem. Dengan diagram ini akan menjelaskan tentang alur program dengan lebih

detail dang lebih bisa dimengerti bagaimana sistem akan bekerja dengan baiknya [3].

a. *Atribut*

Sebuah *class's attributes* adalah informasi yang menggambarkan suatu objek . Atribut ini dapat direpresentasikan pada class diagram dengan menempatkan mereka didalam bagian dari kotak atau oleh asosiasi dengan kelas lain, contoh dapat dilihat pada gambar 2.6 [3].



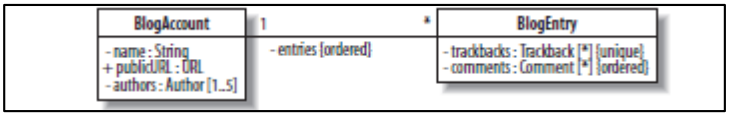
Gambar 2. 6 Atribut.

b. Nama dan Tipe

Dalam atribut nama adalah bagian yang menjadi sebuah atribut tetapi nama tidak diperbolehkan sama . Jenis atribut dapat bervariasi tergantung bagaimana kelas akan diimplementasikan dalam system [3].

c. *Multiplicity*

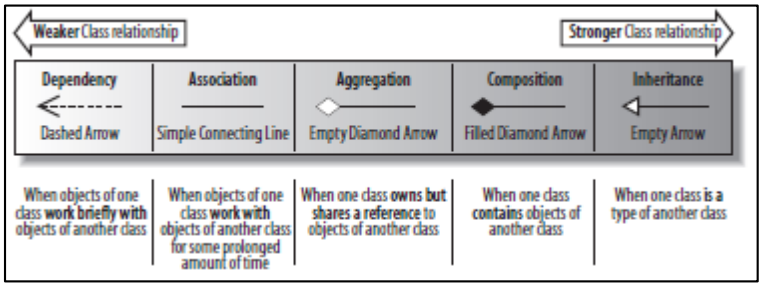
Atribute dapat menggambarkan beberpa objek pengguna atribute untuk menggambarkan beberapa objek contohnya adalah penggunaan untuk Array atau pun list . Contoh dapat dilihat pada gambar 2.7 berikut [3].



Gambar 2. 7 Multiplicity.

d. *Class Relationship*

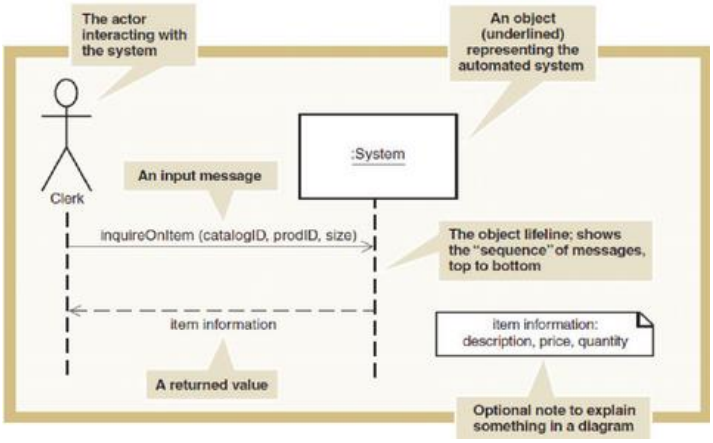
Kelas tidak hidup diruang hampa, mereka bekerja bersama-sama menggunakan berbagai jenis hubungan. Hubungan antar kelas datang dalam kekuatan yang berbeda, seperti ditunjukan pada gambar 2.8 dibawah ini [3].



Gambar 2. 8 *Class Relationship.*

2.8.4 Sequence Diagram

Sequence diagram digunakan untuk menggambarkan scenario dan rangkaian langkah – langkah yang dilakukan sebagai sebuah respon dari kejadian/even untuk menghasilkan output tertentu, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan [3]. Contoh *sequence diagram* bisa dilihat di gambar 2.9 sebagai berikut :



Gambar 2. 9 Sequence Diagram.

2.9 Database

Database atau basis data adalah kumpulan data yang disimpan secara sistematis di dalam komputer dan dapat diolah atau dimanipulasi menggunakan perangkat

lunak (program aplikasi) untuk menghasilkan informasi. Pendefinisian basis data meliputi spesifikasi berupa tipe data, struktur, dan juga batasan-batasan data yang akan disimpan. Basis data merupakan aspek yang sangat penting dalam system informasi dimana basis data merupakan gudang penyimpanan data yang akan diolah lebih lanjut. Basis data menjadi penting karena dapat menghindari duplikasi data, hubungan antar data yang tidak jelas, organisasi data, dan juga update yang rumit. Dalam aplikasi yang penulis buat, maka penulis menggunakan database yang disebut MySQL [2].

2.10 OOP (Obejct Oriented Programming)

Pemogramman berorientasi objek (OOP) merupakan paradigma pemograman yang menganggap semua benda adalah objek, yaitu menganggap segala sesuatu dalam bahasa pemograman adalah objek, bahkan variable pun dianggap sebagai suatu objek. Konsep dasar dari pemrograman berorientasi objek menekankan kepada konsep: Objek (*Object*) "Objek merupakan kumpulan variabel dan fungsi yang saling berkaitan"[2].

1. Kelas (*Class*) "*Class* merupakan suatu rancangan dari objek itu sendiri. Jadi satu *class* dapat berisi banyak objek."
2. Pembungkusan (*Encapsulation*) "*Encapsulation* merupakan cara menyembunyikan detail informasi dan fungsionalitas yang ada pada suatu kelas."
3. Pewarisan (*Inheritance*) "*Inheritance* merupakan kemampuan untuk meng-*extends class* yang akan mewarisi semua karakter dari *partner class*-nya."
4. Polimorfisme (*Pholymorphism*) "*Pholymorphism* merupakan kemampuan objek-objek yang berbeda kelas, tetapi terkait dalam perwarisan untuk merespons secara berbeda terhadap suatu pesan yang sama.

2.11 *Tools* Pembangunan Sistem

2.11.1 Java

Java adalah sebuah bahasa pemrograman yang diciptakan oleh James Gosling, seorang developer dari Sun Microsystem pada tahun 1991. Selanjutnya

Java dikembangkan Sun Microsystem dan banyak digunakan untuk menciptakan Executable Content yang dapat didistribusikan melalui network [2]. Java merupakan salah satu bahasa pemograman tingkat tinggi, Java merupakan bahasa pemograman berorientasi objek yang merupakan paradigma pemograman masa depan sebagai bahasa pemograman java dirancang menjadi handal dan aman juga dirancang agar dapat dijalankan disemua platform [2].

2.11.2 NetBeans

Netbeans adalah sebuah aplikasi *Intergrated Development Environment* (IDE) yang berbasiskan Java dari Sun Microsystems yang berjalan di atas swing. Swing merupakan sebuah teknologi Java untuk mengembangkan aplikasi desktop yang dapat berjalan pada berbagai macam *platform* seperti windos, linux, Max OS X dan Solaris. Sebuah IDE merupakan lingkup pemrograman yang di integrasikan ke dalam suatu aplikasi perangkat lunak yang menyediakan Graphic User Interface (GUI), suatu kode editor atau text, suara compiler dan suatu *debugger* [2]. Ada juga yang bilang *Integrated Design Enviropment* dan *Integrated Debugging Environment,* yakni sebuah program/alat bantu yang terdiri atas *Editor, Compiler, Debugger,* dan *Design* yang terintegrasi dalam satu aplikasi [2].

2.12 *Black-Box Test*

menurut haryanto [4], Tester menggunakan *behavioral test* (disebut juga *Black-Box Tests*), sering digunakan untuk menemukan bug dalam *high level operations*, pada tingkatan fitur, profil operasional dan skenario customer. Pengujian black box ini hanya mengevaluasi sistem dari tampilan luarnya saja. Tester melakukan pengujian lebih ditujukan pada desain software sesuai standar dan reaksi apabila terdapat celah-celah bug pada program aplikasi yang dilakukan pengujian

2.13 Confusion Matrix

Confusion matrix adalah suatu metode yang biasanya digunakan untuk melakukan perhitungan akurasi pada konsep data mining. Metode yang menggunakan tabel matriks seperti tabel [6]. Data dianggap sebagai positif dan data yang lain dianggap negative.

Tabel 2. 1 Model Confusion Matrix

klasifikasi yang benar	Diklasifikasikan sebagai	
	+	-
+	True Positives	False negatives
-	False Positives	True Negatives

True positives adalah jumlah *record positif* yang diklasifikasikan sebagai positif, false positives adalah jumlah record negatif yang diklasifikasikan sebagai positif, false negatives adalah jumlah record positif yang diklasifikasikan sebagai negatif, true negatives adalah jumlah record negatif yang diklasifikasikan sebagai *negative*, kemudian masukkan data uji. Setelah data uji dimasukkan ke dalam confusion matrix, hitung nilai-nilai yang telah dimasukkan tersebut untuk dihitung jumlah *sensitivity (recall), specificity, precision dan accuracy*. *Sensitivity* digunakan untuk membandingkan jumlah TP terhadap jumlah record yang positif sedangkan specificity adalah perbandingan jumlah TN terhadap jumlah record yang negative [12]. Untuk menghitung digunakan persamaan (2.4).

$$Accuracy = \frac{True\ Positives + True\ Negatives}{True\ Positives + False\ Positives + True\ Positives + False\ Negatives} \quad (2.4)$$