BAB 2
LANDASAN TEORI

2.1 Plagiarisme

Pengertian plagiarisme menurut Kamus Besar Bahasa Indonesia (KBBI) **“**Plagiarisme adalah penjiplakan yang melanggar hak cipta” (dikutip dari [KBBI], 2014).

Pengertian plagiarisme dari kata dasar dan sumber lain**.** Plagiarisme berasal dari kata plagiat yang berarti pengambilan karangan (pendapat dan sebagainya) orang lain dan menjadikannya seolah-olah karangan (pendapat dan sebagainya) sendiri, misal menerbitkan karya tulis orang lain atas nama dirinya sendiri (KBBI, 2014). “Plagiarisme adalah bentuk penyalahgunaan hak kekayaan intelektual milik orang lain, yang mana karya tersebut dipresentasikan dan diakui secara tidak sah sebagai hasil karya pribadi” (Sulianta, 2007).

Pengertian Plagiarisme menurut para ahli**.** Neville mengemukakan bahwa Plagiarisme sebagai tindakan mengambil tulisan orang lain tanpa menyebutkan rujukan dan diklaim sebagai miliknya sendiri.

Pengertian Plagiarisme menurut hukum menurut pasal 1 Peraturan Mentri, plagiat adalah perbuatan secara sengaja atau tidak sengaja dalam memperoleh atau mencoba memperoleh kredit atau nilai untuk suatu karya ilmiah, dengan mengutip sebagian atau seluruh karya dan/ atau karya ilmiah pihak lain yang diakui sebagai karya ilmiahnya, tanpa menyatakan sumber secara tepat dan memadai (Mentri Pendidikan Nasional, 2010).

Sejalan dengan definisi yang bermacam-macam. Maka banyak orang membuat “klasifikasi” atau jenis-jenis plagiarisme dengan dasar yang berbeda. Menurut Sudigdo Sastroasmoro (2007, h.5), klasifikasi proporsi atau persentasi kata kalimat, paragraf yang dibajak dibagi menjadi beberapa klasifikasi, yaitu :

a) Plagiarisme ringan : <30%

b) Plagiarisme sedang : 30-70%

c) Plagiarisme berat atau total : >70%

Plagiarisme mempunyai beberapa jenis, yaitu: (a) plagiarisme ide : mengambil ide yang sudah ada tanpa menyebut sumber dengan jelas, (b) plagiarisme isi (data penelitian) : mengambil data penelitian orang lain, (c) plagiarisme kata, kalimat, paragraf, (d) plagiarisme total" (Herqutanto, 2013). Ketiga jenis tindakan plagiarisme ini sering terjadi di kalangan mahasiswa. Penulis mengambil penelitian terhadap jenis plagiarisme pada kata, kalimat, paragraf.

2.2 Preprocessing

Preprocessing merupakan tahapan awal dalam mengolah data input sebelum memasuki proses tahapan utama. *Preprocessing* terdiri dari beberapa tahapan. Adapun tahapan *preprocessing* berdasarkan (Triawati, 2009) , yaitu: case folding, tokenizing / parsing, filtering, stemming. Berikut penjelasan empat tahapan dalam proses *preprocessing* adalah sebagai berikut.

2.2.1 Tokenizing

Tokenizing merupakan proses pemisahan suatu rangkaian karakter berdasarkan karakter spasi, dan mungkin pada waktu yang bersamaan dilakukan juga proses penghapusan karakter tertentu, seperti tanda baca. Sebagai contoh, kata-kata "*computer*", "*computing*", dan "*compute*" semua berasal dari term yang sama yaitu "*comput*" , tanpa pengetahuan sebelumnya dari morfologi bahasa Inggris. Token seringkali disebut sebagai istilah (*term*) atau kata, sebagai contoh sebuah token merupakan suatu urutan karakter dari dokumen tertentu yang dikelompokkan sebagai unit semantik yang berguna untuk diproses (Salton, 1989).



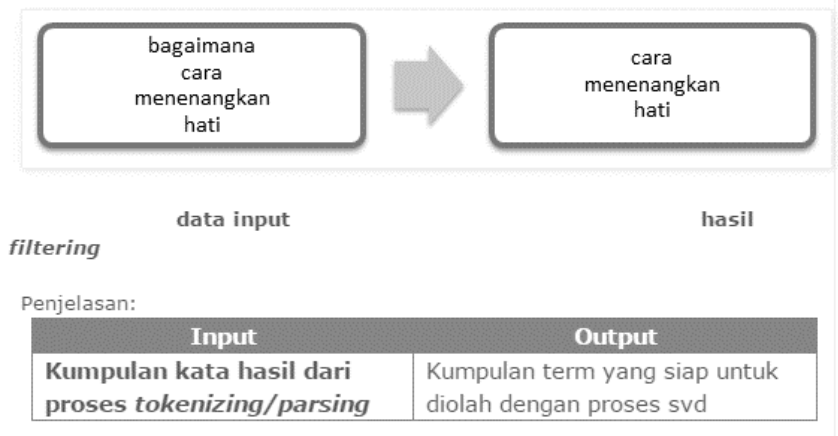
Penjelasan:

Input	Output
Kalimat/kata input hasil dari proses *case folding*	Kumpulan kata

Gambar 2.1 *Tokenizing*

2.2.2 Filtering

Tahap *filtering* adalah tahap mengambil kata - kata penting dari hasil *tokenizing*. Proses filtering dapat menggunakan algoritma *stoplist*(membuang kata yang kurang penting) atau *wordlist* (menyimpan kata penting). *Stoplist / stopword* adalah kata-kata yang tidak deskriptif yang dapat dibuang dalam pendekatan *bag-of-words*. Contoh *stopword* adalah "yang", "dan", "di", "dari" dan lain – lain.(Triawati, 2009).



Penjelasan:

Input	Output
Kumpulan kata hasil dari proses *tokenizing/parsing*	Kumpulan term yang siap untuk diolah dengan proses svd

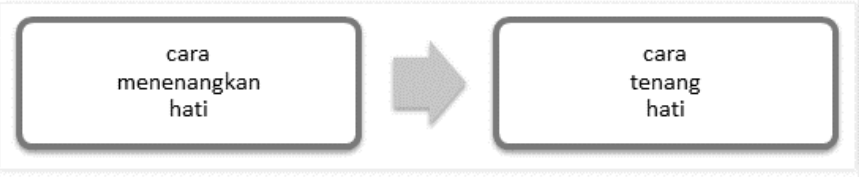
Gambar 2.2 *Filtering*

2.2.3 Stemming

Stemming merupakan suatu proses yang terdapat dalam sistem IR yang mentransformasi kata-kata yang terdapat dalam suatu dokumen ke kata-kata akarnya (*root word*) dengan menggunakan aturan-aturan tertentu (Agusta, 2009). Proses *Stemming* digunakan untuk mengubah *term* yang masih melekat dalam term tersebut awalan, sisipan, dan akhiran. Proses stemming dilakukan dengan cara

menghilangkan semua imbuhan (*affixes*) baik yang terdiri dari awalan *(prefixes*), sisipan *(infixes)*, akhiran *(suffixes)* dan *confixes* (kombinasi dari awalan dan akhiran) pada kata turunan. *Stemming* digunakan untuk mengganti bentuk dari suatu kata menjadi kata dasar dari kata tersebut yang sesuai dengan struktur morfologi bahasa Indonesia yang benar (Tala, 2003). Algoritma *stemming* untuk teks berbahasa Indonesia, diantaranya: Algoritma Porter, Algoritma Nazief & Adriani. Berdasarkan hasil penelitian yang dilakukan(Agusta, 2009).

Pada proses *stemming* menggunakan Algoritma Nazief & Adriani yang membutuhkan waktu lebih lama, tetapi memiliki presentase keakuratan (presisi) lebih besar dibandingkan Algoritma Porter. Kamus yang digunakan juga sangat mempengaruhi hasil *stemming*. Semakin lengkap kamus yang digunakan maka semakin akurat pula hasil *stemming* dan perhitungan presisi.



Gambar 2.3 *Stemming*

2.3 Pembobotan TF-IDF

Term Frequency-Inversed Document Frequency (TF/IDF) merupakan suatu cara untuk memberikan bobot hubungan suatu kata (*term*) terhadap dokumen. Hasil dengan menggunakan metode *tf-idf* adalah nilai atau bobot terhadap kata kunci yang dimasukan. *TF-IDF* menggambarkan dua konsep untuk perhitungan bobot yaitu, frekunsi kemunculan sebuah kata didalam dokumen tertentu dan *inverse* frekuensi dokumen yang mengandung kata tersebut (Robertson, 2004).

Untuk mengetahui nilai dari *tf*, dapat dicari dari jumlah kata pada dokumen yang mengandung kata kunci dari masukan. Dokumen yang mengandung kata kunci akan diambil kemudian akan melalui pre-processing yang akan menghasilkan kata dasar dan setiap kemunculannya akan dihitung sebagai nilai *tf.*

Proses perhitungan *term frequency (tf)* mengunakan persamaan 2.1

$$tf_{dt} = (t, d)$$.....Persamaan 2.1

Dimana:

t = kata

d = dokumen

Untuk mencari nilai dari *df*, dapat dicari banyaknya dokumen yang mengandung kata kunci yang dimasukan. Setiap kata akan dicari keberadaannya pada dokumen, kemudian akan dihitung jumlah dokumen yang mengandung kata tersebut sebagai nilai dari *df.*

Perhitungan *inverse document frequency (idf)* menggunakan persamaan 2.2

$$IDF_t = log\ (D/df_t)$$.....Persamaan 2.2

Dimana:

` *t* = kata ke-t dari kata kunci

df = Jumlah dokumen yang mengandung kata ke-t dari kata kunci

D = Jumlah semua dokumen yang ada di dalam *database*

Idf = Rasio frekuensi dokumen pada kata ke-t dari kata kunci

Perhitungan *term frequency-inverse document frequency* (tf-idf) menggunakan persamaan 2.3

$$W_{d,t} = t\ f_{d,t} * IDF_t$$.....Persamaan 2.3

Dimana:

d = dokumen ke-d

t = kata ke-t dari kata kunci

tf = frekuensi banyaknya kata ke-t dari kata kunci pada dpkumen ke-d

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

IDF = rasio frekunsi dokumen pada kata ke-t dari kata kunci

2.4 Vector Space Model

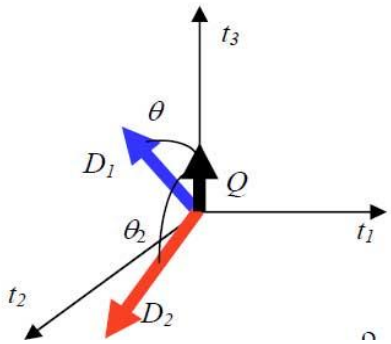
Vector Space Model (VSM) adalah metode untuk melihat tingkat kedekatan atau kesamaan *(similarity)* term dengan cara pembobotan *term.* Dokumen dipandang sebagai sebuah vektor yang memiliki *magnitude* (jarak) dan *direction* (arah). Model ruang vektor sering digunakan untuk mempresentasikan sebuah dokumen dalam ruang vektor (Baeza, 1999).

Kelebihan Vector Space Model (VSM)

1. Adanya peringkat pengambilan informasi
2. Menampilkan referensi yang sesuai kebutuhan
3. Penyocokan secara partial.

Pada VSM, setiap dokumen dan query dari pengguna direpresentasikan sebagai ruang vektor berdimensi n. Biasanya digunakan nilai bobot istilah (term weigthing) sebagai nilai dari vektor pada dokumen nilai 1 untuk setiap istilah yang muncul pada vektor query.

Pada model ini, bobot dari query dan dokumen dinyatakan dalam bentuk vektor, seperti: Q = (wq1, wq2, wq3, . . . ,wqt) dan Di= (wi1, wi2, wi3, . . . , wit) Dengan wqj dan wij sebagai bobot istilah Tj dalam query Q dan dokumen Di. Dengan demikian dokumen yang lebih panjang dengan jumlah istilah yang lebih banyak memiliki kemungkinan lebih besar untuk dianggap relevan dengan istilah-istilah query tertentu dibandingkan dokumen-dokumen yang lebih pendek. Sehingga pada kebanyakan lingkungan penemu-kembalian, vektor dokumen ternormalisasi lebih disukai namun proses normalisasi vektor querytidak diperlukan karena ukurannya yang umumnya pendek dan perbedaan panjang antar-query relatif kecil.



Gambar 2.4 *Vector Space Model*

Gambar 2.4 memperlihatkan tiga buah vektor pada ruang dimensi 3, nilai kosinus digunakan untuk mengukur tingkat kesamaan antar dua vektor. Pada Gambar 2.4, Q adalah vektor dari paragraf pembanding, sementara D_1 dan D_2 adalah vektor dari paragraf yang dibandingkan.

Pengukuran *Cosine Similarity* (menghitung nilai kosinus sudut antara dua vector) menggunakan persamaan 2.4

$$Sim\ (q,dj) = \frac{q.dj}{|q| \ x \ |dj|} = \frac{Wiq.Wij}{\sqrt{Wiq^2} \ x \sqrt{Wij^2}}$$ .....Persamaan 2.4

Dimana:

Sim (q, dj) = similaritas antara *query* dan dokumen

q = bobot *query*

dj = bobot dokumen

|*q*| = kuadrat bobot *query*

|*dj*| = kuadrat bobot dokumen

Wij = bobot *term* dalam dokumen

Wiq = bobot *query* dalam dokumen

2.5 Data Flow Diagram (DFD)

Data Flow Diagram (DFD) merupakan diagram yang menggunakan notasi notasi atau symbol simbol untuk mengambarkan sistem jaringan kerja antar fungsi-fungsi yang berhubungan satu sama lain dengan aliran dan penyimpanan data (Adi Nugroho, 2011).

DFD sering digunakan untuk menggambarkan suatu sistem yang telah ada atau sistem baru yang akan dikembangkan secara logika tanpa mempertimba ngkan lingkungan fisik dimana data tersebut mengalir atau dimana data tersebut akan disimpan. Salah satu keuntungan menggunakan diagram aliran data adalah memudahkan pemakai (*user*) yang kurang menguasai bidang komputer untuk mengerti sistem yang akan dikerjakan.

DFD terdiri dari diagram konteks (*context diagram*) dan diagram rinci (level diagram). Diagram konteks adalah diagram yang terdiri dari suatu proses dan menggambarkan ruang lingkup suatu sistem. Diagram konteks merupakan level tertinggi dari DFD yang menggambarkan seluruh input ke sistem atau outpu t dari sistem. Dalam diagram konteks biasanya hanya ada satu proses. Tidak boleh

ada store dalam diagram konteks. Diagram rinci adalah diagram yang menguraikan proses apa yang ada dalam diagram level di atasnya.

2.6 Kamus Data

Kamus data atau data dictionary adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan kamus data analis sistem dapat mendefinisikan data yang mengalir di sistem dengan lengkap. Pada tahap analisis sistem, kamusdata digunakan sebagai alat komunikasi antara analis sitem dengan pemakai sistem tentang datayang mengalir ke sistem, yaitu tentang data yang masuk ke sistem dan tentang informasi yang dibutuhkan oleh pemakai sistem. Pada tahap perancangan sistem, kamus data digunakan untuk merancang input, merancang laporan-laporan dan database (Jogiyanto, 2005)..

Kamus data dibuat berdasarkan arus data yang ada di Data Flow Diagram (DFD). Arus data di DFD sifatnya adalah global, hanya di tunjukkan nama arus datanya saja. Keterangan lebih lanjut tentang struktur dari suatu arus datadi DFD secara lebih terinci dapat di lihat di kamus data. DD tidak menggunakan notasi grafik sebagaimana halnya DFD. DD berfungsi membantu pelaku sistem untuk mengerti aplikasi secara detil, dan mereorganisasi semua elemen data yang digunakan dalam sistem secara presisi sehingga pemakai dan penganalisa sistem punya dasar pengertian yang sama tentang masukan, keluaran, penyimpanan dan proses. DD mendefinisikan elemen data dengan fungsi sebagai berikut:

1. Menjelaskan arti aliran data dan penyimpanan dalam DFD
2. Mendeskripsikan komposisi paket data yang bergerak melalui aliran,misalnya alamat di uraikan menjadi kota, kodepos, propinsi, dan negara.
3. Mendeskripsikan komposisi penyimpanan data.
4. Menspesifikasikan nilai dan satuan yang relevan bagi penyimpanan dan aliran.
5. Mendeskripsikan hubungan detil antara penyimpanan yang akan menjadi titik perhatian dalam *entity relationship diagram.*

2.7 ERD (Entity Relationship Diagram)

ERD merupakan suatu model untuk menjelaskan hubungan antar data dalam basis data berdasarkan objek-objek dasar data yang mempunyai hubungan antar

relasi**.** ERD untuk memodelkan struktur data dan hubungan antar data, untuk menggambarkannya digunakan beberapa notasi dan simbol (Jogiyanto, 2005). Pada dasarnya ada tiga simbol yang digunakan, yaitu :

a. **Entitas**

Entiti merupakan objek yang mewakili sesuatu yang nyata dan dapat dibedakan dari sesuatu yang lain. Simbol dari entiti ini biasanya digambarkan dengan persegi panjang.

b. **Atribut**

Setiap entitas pasti mempunyai elemen yang disebut *atribut* yang berfungsi untuk mendeskripsikan karakteristik dari entitas tersebut. Isi dari atribut mempunyai sesuatu yang dapat mengidentifikasikan isi elemen satu dengan yang lain. Gambar *atribut* diwakili oleh simbol elips.

c. **Hubungan** / **Relasi**

Hubungan antara sejumlah entitas yang berasal dari himpunan entitas yang berbeda. Relasi dapat digambarkan sebagai berikut :

Relasi yang terjadi diantara dua himpunan entitas (misalnya A dan B) dalam satu basis data yaitu:

1). ***Satu ke satu (One to one)***

Hubungan relasi satu ke satu yaitu setiap entitas pada himpunan entitas A berhubungan paling banyak dengan satu entitas pada himpunan entitas B.

2). ***Satu ke banyak (One to many)***

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B, tetapi setiap entitas pada entitas B dapat berhubungan dengan satu entitas pada himpunan entitas A.

3). ***Banyak ke banyak (Many to many)***

Setiap entitas pada himpunan entitas A dapat berhubungan dengan banyak entitas pada himpunan entitas B.

2.8 PHP (Personal Home Page)

Personal Home Page (PHP) adalah salah satu bahasa *Server-Side* yang didesain khusus untuk aplikasi *web*. PHP dahulunya merupakan proyek pribadi dari

Rasmus Lerdorf (dengan dikeluarkannya PHP versi 1) yang digunakan untuk membuat *home page* pribadinya. Versi pertama ini berupa kumpulan *script* PERL. Untuk versi keduanya, Rasmus menulis ulang *script* - *script* PERL tersebut menggunakan bahasa C, kemudian menambahkan fasilitas untuk *form html* dan koneksi MYSQL.

PHP merupakan bahasa *script* yang digunakan untuk membuat halaman *web* yang dinamis. Dinamis berarti halaman yang ditampilkan dibuat saat halaman itu diminta oleh *client*. Mekanisme ini menyebabkan informasi yang diterima *client* selalu yang terbaru. Semua *script* PHP dieksekusi pada *server* di mana *script* tersebut di jalankan. Oleh karena itu, spesifikasi *server* lebih berpengaruh pada eksekusi dari *script* PHP daripada spesifikasi *client*. Namun tetap diperhatikan bahwa halaman *web* yang dihasilkan tentunya harus dapat dibuka oleh *browser* pada *client*. PHP masuk kedalam kategori *server-side scripting* dimana *browser* pada *client* tidak lagi bertanggung jawab dalam menjalankan kode- kode PHP, melainkan *web server* proses ini diilustrasikan kedalam gambar berikut .

Pertama-tama *web browser* pada *client* me-*request* sebuah *file* (1). Dalam kasus ini bagaimanapun juga *file* yang di-*request* berekstensi/berakhiran .php (contoh: File.php), tanda bahwa didalam file tersebut terkandung kode-kode PHP yang perlu diproses oleh *server*. *Web server* mengenali *file* ini dan tidak mengirim *file* tersebut langsung ke *browser*, tetapi dikirim ke PHP *scripting engine* (2) (mesin pengolah kode-kode PHP). PHP *engine* merupakan komponen perangkat lunak dari *server* yang mampu mengartikan kode-kode PHP dan memberikan *output* dalam kode HTML. Setiap kode PHP dapat memberikan *output* kode HTML yang berbeda, tergantung pada jenis *request* dari *client* (*browser*). Proses tersebut membangkitkan halaman HTML secara dinamis lalu dikirimkan kembali ke *client* (*browser*) untuk merespon terhadap *request* yang sebelumnya telah dikirimkan. Pemrograman di sisi *server* biasanya digunakan untuk membuat sebuah *website* yang interaktif yang dihubungkan kedalam basis data atau data *store* lain .

2.9 HTML (Hyper Text Markup Language)

HyperText Markup Language, sering disebut sebagai HTML, merupakan standar bahasa markup yang digunakan untuk membuat halaman web. Web browser dapat membaca file HTML dan membuat mereka ke dalam halaman web terlihat atau terdengar. HTML menggambarkan struktur dari situs semantik bersama dengan isyarat untuk presentasi, membuatnya menjadi bahasa markup, daripada bahasa pemrograman.

Elemen HTML membentuk blok bangunan dari semua website. HTML memungkinkan gambar dan objek yang akan tertanam dan dapat digunakan untuk membuat bentuk-bentuk interaktif. Menyediakan sarana untuk membuat dokumen terstruktur oleh struktural menunjukkan semantik untuk teks seperti judul, paragraf, daftar, link, kutipan dan item lainnya.

Bahasa ini ditulis dalam bentuk elemen HTML terdiri dari tag diapit kurung sudut (seperti <html>). Browser tidak menampilkan tag HTML dan script, tetapi menggunakannya untuk menafsirkan isi halaman.

HTML dapat menanamkan script yang ditulis dalam bahasa seperti JavaScript yang mempengaruhi perilaku halaman web HTML. Web browser juga dapat merujuk ke Cascading Style Sheets (CSS) untuk menentukan tampilan dan tata letak teks dan bahan lainnya. The World Wide Web Consortium (W3C), pengelola dari kedua HTML dan CSS standar, mendorong penggunaan CSS lebih eksplisit HTML presentasi sejak tahun 1997.

2.10 MySQL

MySQL adalah *Relational Database Managemen System* (RDBMS) yang didistribusikan secara gratis di bawah lisensi GPL (*General Public Licence*). Dimana setiap orang bebas untuk menggunakannya, tapi tidak boleh dijadikan produk turunan yang bersifat *Closed Source* atau komersial.

MySQL sebenarnya merupakan turunan salah satu konsep utama dalam *database* sejak lama, yaitu SQL (*Structur Query Language*). SQL adalah sebuah konsep pengoperasian *database* terutama untuk pemilihan atau seleksi dan

pemasukan data, yang memungkinkan pengoperasian data dikerjakan dengan mudah secara otomatis.

Keandalan suatu sistem *database* (DBMS) dapat diketahui dari cara kerja optimizer-nya dalam melakukan proses perintah-perintah SQL, yang dibuat oleh *user* maupun program-program aplikasinya sebagai *database server* lainnya dalam *query* data. Hal ini terbukti untuk *query* yang dilakukan oleh *single user*, kecepatan *query* MySQL bisa sepuluh kali lipat lebih cepat.