BAB 2
TINJAUAN PUSTAKA

2.1 Plagiarisme

Menurut Peraturan Menteri Pendidikan RI Nomor 17 Tahun 2010 plagiat adalah perbuatan 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.

Dalam Kamus Besar Bahasa Indonesia (2008) disebutkan plagiat adalah pengambilan karangan (pendapat dan sebagainya) orang lain dan menjadikannya seolah-olah karangan (pendapat) sendiri.

Berdasarkan beberapa definisi plagiarisme di atas, berikut ini diuraikan ruang lingkup plagiarisme [10]:

1. Mengutip kata-kata atau kalimat orang lain tanpa menggunakan tanda kutip dan tanpa menyebutkan identitas sumbernya.
2. Menggunakan gagasan, pandangan atau teori orang lain tanpa menyebutkan identitas sumbernya.
3. Menggunakan fakta (data, informasi) milik orang lain tanpa menyebutkan identitas sumbernya.
4. Mengakui tulisan orang lain sebagai tulisan sendiri.
5. Melakukan parafrase (mengubah kalimat orang lain ke dalam susunan kalimat sendiri tanpa mengubah idenya) tanpa menyebutkan identitas sumbernya.
6. Menyerahkan suatu karya ilmiah yang dihasilkan dan/atau telah dipublikasikan oleh pihak lain seolah-olah sebagai karya sendiri.

Pada penelitian ini sistem yang akan dibuat bukan untuk mendeteksi plagiarisme melainkan salah satu indikatornya yaitu kesamaan kalimat berupa parafrase (mengubah kalimat orang lain ke dalam susunan kalimat sendiri tanpa mengubah idenya) dalam dokumen.

2.2 *Preprocessing*

Preprocessing merupakan tahapan awal dalam mengolah data input sebelum memasuki proses tahapan utama. Dalam penelitian ini tahap preprocessing terdiri dari: tokenisasi kalimat, *case folding,* tokenisasi kata ,filter kata, *stemming*, filter kalimat, dan TF-IDF [6].

2.2.1 Tokenisasi Kalimat

Tokenisasi kalimat merupakan proses pemisahan teks pada dokumen menjadi kumpulan kalimat. Teknik yang digunakan dalam pemisahan kalimat adalah memisahkan kalimat dengan tanda titik (.) sebagai *delimeter*. Sebagai contoh lihat tabel 2.1.

Tabel 2.1 Tokenisasi Kalimat

Masukkan	Token
Saya kuliah di Unikom. Saat ini saya sedang mengerjakan skripsi.	Saya kuliah di Unikom
	Saat ini saya sedang mengerjakan skripsi

2.2.2 *Case Folding*

Case Folding merupakan proses dimana semua huruf pada teks masukan diubah menjadi huruf non-kapital atau kapital. *Case folding* diperlukan supaya tidak ada permasalahan pada sistem saat memproses kata yang sama dengan tipe huruf yang berbeda. Contohnya kata “Saya” dan “saya”, dua kata tersebut sama, tetapi karena “Saya” diawali dengan huruf kapital, sistem akan menganggap kata tersebut berbeda dengan kata “saya”.

2.2.3 Tokenisasi Kata

Tokenisasi kata adalah pemotongan *string input* berdasarkan tiap kata yang menyusunnya. Pada penelitian ini kata dideksi menggunakan *word detector*(nltk) dan setiap karakter bukan kata tetap akan masuk token. Sebagai contoh lihat tabel 2.3.

Tabel 2.2 Contoh Tokenisasi Kata

Masukkan	Token
Saya kuliah di Unikom. Saat ini saya sedang mengerjakan	Saya
	kuliah

skripsi.	di
	unikom
	.
	Saat
	ini
	sedang
	mengerjakan
	skripsi
	.

2.2.4 Filter Kata

Filter kata pada penelitian ini digunakan untuk menseleksi token pada hasil tokenisasi kata dengan kriteria sebagai berikut [6]:

a. Hapus token kata yang bukan alphanumeric

   Sebagai contoh kata ‘ada’, ‘apa’,’ada12’ akan masuk kedalam token karena merupakan alphanumeric, sedangkan karakter ‘.’,’,’’-‘, dll akan dihapus.

b. Hapus kata jika jumlah karakter pada kata kurang dari 3

   Sebagai contoh kata ‘ad’ akan dihapus karena jumlah karakter pada kata tersebut kurang dari 3 yaitu ‘a’ dan ‘d’.

2.2.5 *Stemming*

Stemming merupakan suatu proses atau cara dalam menemukan kata dasar dari suatu kata [11]. *Stemming* sendiri berfungsi untuk menghilangkan variasi-variasi morfologi yang melekat pada sebuah kata dengan cara menghilangkan imbuhan-imbuhan pada kata tersebut, sehingga nantinya di dapat suatu kata yang benar sesuai struktur morfologi bahasa Indonesia yang benar.

Stemming sendiri merupakan bagian dari proses *Information Retrieval*. Secara umum ada dua proses utama dalam IR, yaitu *Indexing* dan *Searching*. Proses *Indexing* terdiri dari 4 subproses antara lain: *Word Token* (mengubah dokumen menjadi kumpulan *term* dengan cara menghapus semua karakter dalam

tanda baca yang terdapat pada dokumen dan mengubah kumpulan term menjadi *lowercase*), *StopWord Removal* (Proses penghapusan katakata yang sering ditampilkan dalam dokumen seperti:*and, or, not* dan sebagainya), *Stemming* (Proses mengubah suatu kata bentukan menjadi kata dasar) dan *Word Weighting* (Proses pembobotan setiap *term* di dalam dokumen) [11].

Algoritma yang digunakan untuk *stemming* adalah algoritma nazief dan andriani. Algoritma nazief dan andriani memiliki kelebihan dari segi prosentasi keakuratan (presisi) lebih besar dibandingkan algoritma poter. Langkah algoritma nazief dan andriai dijelaskan sebagai berikut [11]:

1. Cari kata yang akan distem dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tesebut adalah *root word*. Maka algoritma berhenti.
2. *Inflection Suffixes* (“-lah”, “-kah”, “-ku”,“-mu”, atau “-nya”) dibuang. Jika berupa particles (“-lah”, “-kah”, “-tah” atau “-pun”) maka langkah ini diulangi lagi untuk menghapus *Possesive Pronouns* (“-ku”, “-mu”, atau “-nya”), jika ada.
3. Hapus *Derivation Suffixes* (“-i”, “-an” atau “-kan”). Jika kata ditemukan dikamus, maka algoritma berhenti. Jika tidak maka ke langkah 3.1.
    3.1. Jika “-an” telah dihapus dan huruf terakhir dari kata tersebut adalah “-k”, maka “-k”juga ikut dihapus. Jika kata tersebut ditemukan dalam kamus maka algoritma berhenti. Jika tidak ditemukan maka lakukan langkah 3.2.
    3.2. Akhiran yang dihapus (“-i”, “-an” atau “-kan”) dikembalikan, lanjut ke langkah 4.
4. Hapus *Derivation Prefix*. Jika pada langkah 3 ada sufiks yang dihapus maka pergi ke langkah 4.1, jika tidak pergi ke langkah 4.2.
    4.1. Periksa table kombinasi awalan-akhiran yang tidak diijinkan. Jika ditemukan maka algoritma berhenti, jika tidak pergi ke langkah 4.2.
    4.2. *For* i = 1 *to* 3, tentukan tipe awalan kemudian hapus awalan. Jika *root word* belum juga ditemukan lakukan langkah 5, jika sudah maka

algoritma berhenti. Catatan: jika awalan kedua sama dengan awalan pertama algoritma berhenti.

5. Melakukan Recoding.

   Jika semua langkah telah selesai tetapi tidak juga berhasil maka kata awal diasumsikan sebagai *root word*. Proses selesai.

Pada penelitian ini digunakan library *stemming* bahasa Indonesia untuk Python dari Sastrawi. Karena *library* ini menggunakan beberapa algoritma yaitu algoritma Nazief dan Adriani, algoritma CS *Stemmer* dan algoritma CS *Stemmer*. Dimana algoritma CS *Stemmer* dan algoritma ECS *Stemm* CS *Stemmer* memodifikasi beberapa aturan yang terdapat di algoritma Nazief dan Adriani.

2.2.6 Filter Kalimat

Filter kalimat yaitu bertujuan untuk menggabungkan kalimat dengan kalimat berikutnya jika kalimat tersebut kurang dari tiga kata. Sebagai contoh terdapat kalimat "Saya Viky. Saya kuliah di unikom.". dapat dilihat pada kalimat pertama terdapat dua kata yaitu "Saya Viky" maka kalimat tersebut digabungkan dengan kalimat berikutnya menjadi "Saya Viky Saya kuliah di unikom." [6].

2.3 TF-IDF

Metode TF-IDF merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada *information retrieval*. Metode ini juga terkenal efisien, mudah dan memiliki hasil yang akurat [12]. Metode ini akan menghitung nilai *Term Frequency* (TF) dan *Inverse Document Frequency* (IDF) pada setiap token kata di setiap dokumen. Pada penelitian ini dokumen menunjuk ke kalimat (*Inverse Sentence Frekuensi*(*ISF*)) bukan dokumen [6]. Metode ini akan menghitung bobot setiap token *t* di kalimat *d* dengan rumus:

$$W_{ts} = tf_{ts} * ISF_t \tag{2.1}$$

Dimana:

s : kalimat ke-s

t : kata ke-t

W_{ts} : bobot kalimat ke-s dari kata ke-t

tf_{ts} : banyak kata ke-t pada sebuah kalimat ke-s

ISF : Inversed Sentence Frequency

Nilai ISF didapatkan dari rumus:

$$ISF = log.\left(\frac{D}{sf}\right) \quad (2.2)$$

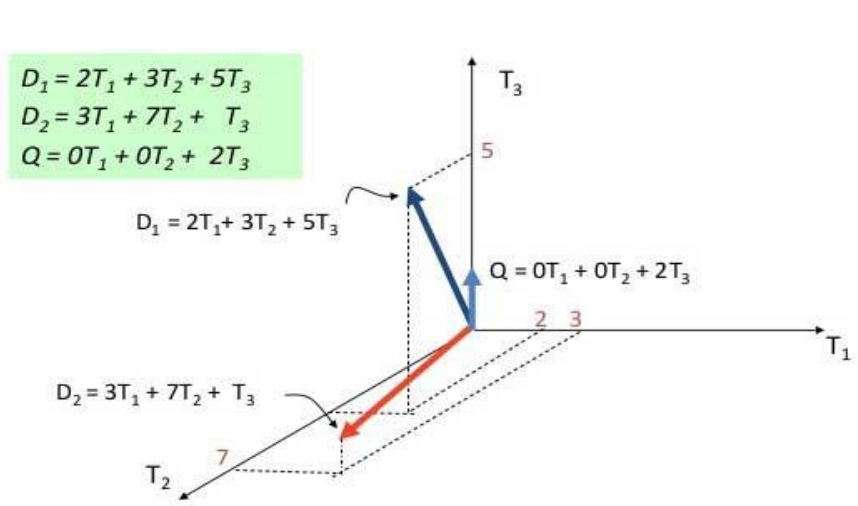
Dimana:

D : total kalimat

sf : banyak kalimat yang mengandung kata yang dicari

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 [12].



Gambar 2.1 Vector Space Model

Gambar 2.1 diatas menunjukkan pemodelan dokumen teks di ruang dimensi dimana (D) adalah kalimat dokumen sedangkan (T) adalah *term* atau kata. Misalkan terdapat sejumlah n kata yang berbeda sebagai kamus kata (*vocabulary*) atau indeks kata (*terms-index*). Kata-kata ini akan membentuk ruang vektor yang memiliki dimensi sebesar n. Setiap kata i dalam dokumen atau query diberikan bobot sebesar wi. Baik dokumen maupun *query* direpresentasikan sebagai vektor berdimensi n [12].

$$\begin{bmatrix} & T_1 & T_2 & \dots & T_t \\ D_1 & w_{11} & w_{21} & \dots & w_{t1} \\ D_2 & w_{12} & w_{22} & \dots & w_{t2} \\ \dots & \dots & \dots & \dots & \dots \\ D_n & w_{1n} & w_{2n} & \dots & w_{tn} \end{bmatrix}$$

Gambar 2.2 Matrix term dokumen

Untuk mendapatkan nilai jarak atau kemiripan dokumen, dapat menggunakan berbagai macam varian rumus perhitungan jarak diantaranya adalah (1) *Cosine*, (2) *Jaccard*, (3) *Dice*, (3) *Euclidean*, (4) *Manhattan*, (5) *Minkowski*, (6) *Mahalanobis*, (8) *Weighted*. Pada penelitian ini rumus yang digunakan adalah *cosine similarity* dan *dice coefficient* [6].

Cosine similarity adalah ukuran kesamaan yang lebih umum digunakan dan merupakan ukuran sudut antara vektor dokumen. Tiap vektor merepresentasikan setiap kata dalam setiap dokumen (teks) yang dibandingkan [11]. Adapun untuk menghitung *Cosine Similarity* digunakan persamaan berikut.

$$\cos(D,Q) = \frac{D.Q}{|D||Q|} \quad (2.3)$$

Dimana :

$D.Q$ = perkalian dot product vektor kalimat D dan Q

|D| = panjang vektor kalimat *D*

|Q| = panjang vektor kalimat *Q*

Dice Coefficient yang dikenal juga dengan sebutan *Sørensen–Dice index* merupakan metode yang digunakan untuk membandingkan tingkat similaritas dari dua objek. Metode ini dipublikasikan oleh Sørensen dan Lee Raymond Dice pada 1948 dan 1945 secara berturut-turut. Rumus ini memiliki kesamaan dengan rumus *Jaccard*. Namun, perbedaannya terletak pada adanya pencocokan dua kali pada rumus *Dice's coefficient*. Berikut adalah rumus *Dice's coefficient* [13].

$$\text{dice}\ (\ susp_i, src_j\ ) = \frac{2\ |\ \delta(suspi).\delta(srcj)|}{|\delta(suspi)^2|+|\delta(srcj)^2|} \quad (2.4)$$

Dimana :

$|\ \delta(suspi).\delta(srcj)|$ = banyak kesamaan kata di kalimat susp ke i dan src j

$|\delta(suspi)^2|$ = banyak kata di kalimat *suspicious* ke i

$|\delta(srcj)^2|$ = banyak kata di kalimat *source* ke j

2.5 *Seeding*

Proses *Seeding* dilakukan untuk membangun sebuah set *S* yang berupa pasangan kalimat yang sama antara dokumen *suspicious* dan *source* [13]. Adapun tahapan yang dilakukan adalah pertama hitung TF-IDF lalu hitung nilai kesamaan dengan *Cosine Similarity* dan *Dice Coefficient*. Masuk kedalam kategori *S* jika memenuhi persamaan berikut [13].

$$S = \{\ (\ i\ ,\ j\ )\ |\ cos\ (\ susp_i, src_j\ ) > mincos \cap dice\ (\ susp_i, src_j\ ) > mindice\} \quad (2.5)$$

Dimana :

i = index kalimat di *suspicious document* (integer)

j = index kalimat di *source document* (integer)

$susp_i$ = vektor kalimat *suspicious document* ke i

src_j = vektor kalimat *source document* ke j

mincos = nilai parameter inputan (*range* 0,0 – 1,0)

mindice = nilai parameter inputan (*range* 0,0 – 1,0)

cos ($susp_i, src_j$) = nilai *cosine similarity*

dice ($susp_i, src_j$) = nilai *dice coefficient*

2.6 *Extension*

Diberikan nilai dari *S*(i,j) yang berupa fragment kecil dari pasangan kalimat yang sama, tahap *extension* bertujuan untuk membuat fragment besar yang berupa pasangan kalimat yang sama. Pada prosesnya dibagi menjadi dua yaitu *clustering* dan *validation* [13].

Pada tahap **clustering** nilai dari *seed* dikelompokan berdasarkan nilai *maxgap* antar kalimat. Dalam penyelesaiannya pengelompokan dilakukan pada nilai *seed* yang ada di dokumen *suspicious* terlebih dahulu dimana $|i_n - i_{n+1}| - 1 \leq$ *maxgap*. Kemudian hasil dari pengelompokan dari *seed* dokumen *suspicious* dikelompokkan lagi berdasarkan dokumen *source* dimana $|j_n - j_{n+1}| - 1 \leq$ *maxgap*, lakukan sampai tidak ada lagi kluster yang terbentuk. Hasil kluster yang kurang dari *minsize* akan dihapus [13].

Tahap **validation** berfungsi untuk menilai kualitas dari kluster yang terbentuk. Untuk menilai kualitas digunakan persamaan 2.4 yaitu dengan menghitung nilai *cosine similarity* antar *fragment text*. Jika nilai *cosine* dalam kluster kurang dari *threshold* yang diberikan (*th_validation*), kembali ke tahap kluster awal dengan nilai *maxgap* = *maxgap* - 1. Nilai *maxgap* akan terus berkurang sampai *maxgap* = *maxgap_least* jika threshold (*th_validation*) tidak terpenuhi. Jika nilai *maxgap_least* tercapai dan *th_validation* tidak tercapai maka kluster dihapus [13].

2.7 *Filtering*

Pada proses *filtering* jumlah karakter yang terdapat pada kalimat hasil proses *extension* akan diseleksi berdasarkan nilai parameter inputan *minplaglen* (minimum jumlah karakter). Jika jumlah karakter dari kluster hasil dari proses *extension* kurang dari *minplaglen* maka kluster akan dihapus [13].

2.8 Penghitungan Akurasi

Pengujian akurasi yang akan dilakukan adalah dengan cara melakukan evaluasi hasil kesamaan dengan menggunakan metode evaluasi intrinsik. Salah satunya dengan menggunakan ROUGE [9]. Perhitungan akurasi dengan metode ini dengan cara melihat hasil kesamaan dari sistem dan hasil kesamaan manual

dari manusia. Evaluasi ini mengukur nilai *recall* dan *precission* kalimat dan kombinasi antara keduanya yang menghasilkan *f-measure*. Nilai – nilai tersebut didapatkan melalui rumus 2.6, 2.7, dan 2.8.

$$recall = \frac{hits}{hits + misses} \quad (2.6)$$

$$precission = \frac{hits}{hits + noise} \quad (2.7)$$

$$f\text{-}measure = \frac{2 * precission * recall}{recall + precission} \quad (2.8)$$

Dimana :

$hits$ = Jumlah kalimat sama pada dokumen *suspicious* atau *source* yang berhasil diekstrak sistem yang sesuai dengan hasil ekstrak manusia

$misses$ = Jumlah kalimat sama pada dokumen *suspicious* atau *source* yang diekstrak manusia tetapi tidak terdapat dalam hasil ekstrak sistem

$noice$ = Jumlah kalimat sama pada dokumen *suspicious* atau *source* yang diekstrak sistem tetapi tidak terdapat dalam kalimat yang diekstrak manusia

2.9 Pemodelan

Skema pemodelan adalah metode yang memungkinkan untuk memodelkan atau menggambarkan rancangan sistem pada aplikasi deteksi kesamaan dokumen yang akan dibangun.

2.9.1 Diagram Konteks

Diagram Konteks adalah sebuah diagram sederhana yang menggambarkan hubungan antara *entity* luar, masukan dan keluaran dari sistem. Diagram konteks direpresentasikan dengan lingkaran tunggal yang mewakili keseluruhan sistem [14]. Diagram konteks dimulai dengan penggambaran terminator, aliran data, aliran kontrol penyimpanan, dan proses tunggal yang menunjukan keseluruhan sistem. Ada beberapa aturan dalam menggambarkan diagram konteks, berikut di bawah ini menunjukan aturan tersebut.

1. Menggunakan hanya satu simbol proses
2. Memberi label simbol proses tersebut untuk menggambar seluruh sistem, biasanya berupa kata kerja di tambah objek

3. Tidak memberi nomor pada simbol proses
4. Menyertakan semua terminator dari sistem
5. Menunjukan semua arus data antara terminator dan sistem

2.9.2 DFD (Data Flow Diagram)

DFD adalah suatu model logika data atau proses yang dibuat untuk menggambarkan darimana asal data dan kemana tujuan data yang keluar dari sistem, dimana data disimpan, proses apa yang menghasilkan data tersebut dan interaksi antara data yang tersimpan dan proses yang dikenakan pada data tersebut [14].

1. Arus Data (*Data Flow*)

   Menunjukan arus dari data yang dapat berupa masukan untuk sistem atau dari proses sistem.
2. Proses

   Proses adalah kegiatan yang dilakukan oleh orang, mesin atau komputer dari hasil arus data yang masuk ke dalam proses untuk dihasilkan arus data yang akan keluar dari proses.
3. Kesatuan Luar (*External Entity*)

   Kesatuan luar merupakan kesatuan di lingkungan luar sistem yang dapat berupa orang, organisasi atau sistem lain yang akan memberikan masukan atau menerima keluaran dari sistem.
4. File

   Kumpulan data yang disimpan dengan cara tertentu. Data yang mengalir disimpan dalam file. Aliran data di-update atau ditambahkan ke dalam file.

2.9.3 Flowchart

Flowchart adalah bagan-bagan yang mempunyai arus yang menggambarkan langkah-langkah penyelesaian suatu masalah. *Flowchart* merupakan cara penyajian suatu algoritma. *Flowchart* adalah penggambaran secara grafik dari langkah-langkah dan urutan-urutan prosedur dari suatu program.

Flowchart biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut [14].

Tujuan membuat *Flowchart* adalah menggambarkan suatu tahapan penyelesaian masalah secara sederhana, terurai, rapi dan jelas menggunakan simbol-simbol standar [14].

2.10 Bahasa Pemrograman

Bahasa pemrograman merupakan kumpulan aturan yang disusun sedemikian rupa sehingga memungkinkan pengguna komputer membuat program yang dapat dijalankan dengan aturan tersebut [15]. Beberapa contoh bahasa pemrograman yang banyak dipakai, yaitu C, C++, C#, Pascal, Java, JavaScript, PHP, SQL, dan Python. Bahasa pemrograman yang digunakan pada pembangunan aplikasi deteksi kesamaan dokumen ini adalah Python.

2.10.1 Python

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. Python diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. Python juga didukung oleh komunitas yang besar [15].

Python mendukung multi paradigma pemrograman, utamanya; namun tidak dibatasi; pada pemrograman berorientasi objek, pemrograman imperatif, dan pemrograman fungsional. Salah satu fitur yang tersedia pada python adalah sebagai bahasa pemrograman dinamis yang dilengkapi dengan manajemen memori otomatis. Seperti halnya pada bahasa pemrograman dinamis lainnya, python umumnya digunakan sebagai bahasa skrip meski pada praktiknya penggunaan bahasa ini lebih luas mencakup konteks pemanfaatan yang umumnya tidak dilakukan dengan menggunakan bahasa skrip. Python dapat digunakan untuk berbagai keperluan pengembangan perangkat lunak dan dapat berjalan di berbagai platform sistem operasi [15].