BAB II

TINJAUAN PUSTAKA

II.1. Penelitian Terkait

Untuk mendukung keberhasilan penelitian ini, penyusun melakukan pendekatan teoritis melalui beberapa literatur yang berhubungan dengan penelitian yang dilakukan. Beberapa uraian penelitian terdahulu yang menjadi acuan dalam penelitian ini yaitu :

Menurut penelitian Nur Alamsyah (2017) dengan judul "**PERBANDINGAN ALGORITMA WINNOWING DENGAN ALGORITMA RABIN KARP UNTUK MENDETEKSI PLAGIARISME PADA KEMIRIPAN TEKS JUDUL SKRIPSI**" Pendekatan Algoritma Winnowing lebih baik daripada pendekatan algoritma *Rabin Karp* karena menghasilkan tingkat presentase yang lebih kecil dan waktu proses yang lebih cepat, Berdasarkan hasil Pengujian terhadap perbandingan pendekatan algoritma winnowing dengan algoritma *Rabin Karp* dapat dilihat kemungkinan kemiripan teks judul skripsi yang terkecil adalah dengan penggunakan pendekatan algoritma winnowing yaitu pada ujicoba ke 8 dengan nilai n-gram = 9 dan window = 3 , proses waktu 0.0257 dengan tingkat kemiripan terkecil yaitu 32.6 %."

Langkah-langkah deteksi kemiripan judul skripsi dapat digambarkan sebagai berikut:

1. Masukan judul skripsi pada masing- masing algoritma yaitu algoritma winnowing dan algoritma *Rabin Karp* nantinya akan dilihat tingkat presentase dengan judul skripsi yang sudah ada sebelumnya.
2. Memasukan nilai *n-gram*, untuk memebentuk rangkaian gram pada judul yang dimasukan dan judul yang dibandingkan.

3. Masukan nilai Window hanya pada algoritma Winnowing, untuk menentukan pembentukan window dari nilai Hash.

4. Deteksi kemiripan judul skripsi akan diproses dari masing-masing algoritma, dari proses deteksi kemiripan judul tersebut akan menampilkan tingkat presentase dengan judul-judul yang sudah ada sebelumnya.

Fatkhul Amin (2018) **"KOMPARASI KINERJA ALGORITMA SIMILARITAS INNER PRODUCT FAMILY PADA RULE BASE STEMMER STUDI KASUS DOKUMEN TEKS BAHASA JAWA"** Komparasi Algoritma Similaritas *Inner Product Family* dilakukan untuk mengetahui efektifitas algoritma dalam menemukan dokumen teks pada studi kasus dokumen teks bahasa jawa. Dokumen bahasa jawa yang menjadi obyek sejumlah 48.753 kata yang didapatkan dari majalah bahasa jawa penjebar semangad, Joko lodang dan Jaya Baya.
Kinerja similaritas didapatkan dari proses Pengolahan dokumen yang ditempatkan di korpus. Dokumen yang telah ditempatkan dikorpus dilakukan proses Preprosessing yang didalamnya terdapat 3 (tiga) proses yaitu; Tokenizing, Filtering dan Stemming. Setelah melalui proses stemmer menggunakan bahasa jawa metode rule based, selanjutnya tiap-tiap metode diuji cobakan dengan menggunakan beberapa keyword yang sama. Adapun keyword yang diujikan adalah keyword dengan 1 term, 2 term, 3 term, 4 term dan 5 term. Melalui ke-5 keyword tersebut, selanjutnya setiap hasil didata dan dilakukan analisa komparasi setelah sebelumnya dilakukan analisis persepsi untuk setiap keyword yang diujikan. Tahap akhir dari proses melihat kinerja adalah dengan dilakukan uji recall dan uji presisi.

Penelitian yang dilakukan oleh Andry Hery Purba (2017) "**Analisis Perbandingan Algoritma Rabin-Karp Dan Levenshtein Distance Dalam Menghitung Kemiripan Teks**" Proses yang harus dilakukan dalam sistem adalah proses preprocessing pada masing-masing algoritma dan kemudian menghitung tingkat kemiripan teks menggunakan algoritma *Rabin Karp* dan *Levenshtein Distance*. Sehingga mendapatkan hasil dari kedua algoritma yang ada pada sistem untuk dianalisis perbandingan hasilnya berupa grafik. Pada pengujian peneliti menggunakan dokumen teks dengan tipe .pdf dan berbahasa Indonesia.

Berikut ini adalah langkah-langkah algoritma *Levenshtein Distance* dalam mendapatkan nilai distance

Langkah 1: Inisialisasi

a) Hitung panjang S dan T, misalkan m dan n

b) Buat matriks berukuran 0…m baris dan 0…n kolom

c) Inisialisasi baris pertama dengan 0…n

d) Inisialisasi kolom pertama dengan 0…m

Langkah 2: Proses

a) Periksa S[i] untuk $1 < i < n$

b) Periksa T[j] untuk $1 < j < m$

c) Jika S[i] = T[j], maka entrinya adalah nilai yang terletak pada tepat didiagonal atas sebelah kiri, yaitu d[i,j] = d[i-1, j-1].

d) Jika $S[i] \neq T[j]$, maka entrinya adalah d[i,j] minimum dari:

¬ Nilai yang terletak tepat diatasnya, ditambah satu, yaitu d[i,j-1]+1

¬ Nilai yang terletak tepat dikirinya, ditambah satu, yaitu d[i-1,j]+1

¬ Terletak pada tepat didiagonal atas sebelah kirinya, ditambah satu, yaitu d[i-1,j-1]+1

Langkah3:

Hasil entri matriks pada baris ke-i dan kolom ke j, yaitu d[i,j]

Langkah 4:

Diulang hingga entri d[m,n] ditemukan.

Menurut penelitian Satia Suhada (2017) "**IMPLEMENTASI ALGORITMA RABIN KARP DAN STEMMING NAJIEF ANDRIANI UNTUK DETEKSI PLAGIARISME DOKUMEN**" Stemming yang akan digunakan dalam sistem pendeteksi plagiarisme ialah steming *najif andriani* yang akan digunakan untuk preprossing teks sebelum kemudian di cari kemiripan kata algoritma *najif andriani* memiliki kelebihan dari segi prosentasi keakuratan (presisi) lebih besar dibanding dengan algoritma stemming porter

Tahapan pada algoritma najief dan andriani :

a. Cari kata yang akan distem dalam kamus. Jika ditemukan maka diasumsikan bahwa kata tesebut adalah root word. Maka algoritma berhenti.
b. *Inflection Suffixes* ("-lah", "-kah", "-ku", "-mu", atau "-nya") dibuang. Jika berupa particles ("-lah", "-kah", "-tah" atau "- pun") maka langkah ini diulangi lagi ISSN : 2355-990X E-ISSN : 2549-5178 86 SWABUMI Vol. 5, Maret 2017: 84 – 89 untuk menghapus *Possesive Pronouns* ("-ku", "-mu", atau "-nya"), jika ada.

Berdasarkan penelitian Rizal Sengkey (2020) dengan judul "**Penerapan Clustering pada Aplikasi Pendeteksi Kemiripan Dokumen Teks Bahasa Indonesia**" Metode ini mempartisi data ke dalam *cluster* sehingga data yang memiliki karakteristik yang sama dikelompokan ke dalam satu *cluster* yang sama dan data yang mempunyai karakteristik yang berbeda dikelompokan ke dalam kelompok yang lain.

Manfaat *clustering* adalah sebagai Identifikasi Object (Recognition) misalnya dalam bidang *Image Processing*, *Computer Vision* atau *Robot vision*. Selain itu adalah sebagai Sistem Pendukung Keputusan dan Data Mining seperti segmentasi pasar, pemetaan wilayah, manajemen marketing, dll.

II.2. Landasan Teoritis

II.2.1. Algoritma Winnowing

Menurut (Muhammad Nur Khidfi, 2018) Implementasi dari algoritma *Winnowing* membutuhkan masukan berupa file teks dan menghasilkan keluaran berupa nilai hash yang disebut finger print. Setiap kata yang terkandung dalam file teks diubah terlebih dahulu menjadi sebuah kumpulan nilai hash dengan teknik rolling hash. Nilai hash merupakan nilai numerik dari perhitungan ASCII untuk setiap karakter. Lalu kumpulan nilai hash yang disebut finger print tersebut digunakan untuk mendeteksi kemiripan antar dokumen.

1. Rolling Hash

Teknik *Rolling Hash* pada awalnya digunakan pada algoritma *Rabin-Karp*. Setiap karakter di dalam dokumen teks diubah (encode) menjadi nilai array bilangan bulat, sehingga nilai masukan yang awalnya berupa karakter menjadi fungsi hash berupa angka. Untuk membandingkan dua string yang dianggap sama, maka setiap A[i] = B[i] dan membutuhkan waktu sebesar O(n). Panjang waktu yang dibutuhkan tergantung pada panjang iterasi elemen string yang dibandingkan.

Metode dasar untuk mencari perbandingan antara kedua string dokumen A dan B adalah:

1. Asumsikan dokumen A memiliki panjang elemen string p, dan dokumen B

memiliki panjang q.

2. Lakukan hashing pada dokumen A untuk mendapatkan h(A) dengan waktu sebesar O(p).
3. Lakukan iterasi pada dokumen B dengan panjang elemen string p, dan bandingkan h(A) dengan waktu sebesar O(qp).
4. Jika nilai hash substring tidak cocok dengan h(A), bandingkan substring yang ada dengan A. Jika cocok, berhenti, jika tidak, lakukan kembali hingga ditemukan waktu sebesar O(p).

Untuk mengurangi waktu komputasi, dapat dilakukan teknik *rolling hash* dengan mengambil waktu sebesar O(p) sehingga didapatkan banyak kecocokan. Contoh, lakukan hashing 5 substring pada kata "komputer". Hash I: "kompu", hash II: "oputer", dan seterusnya. Dengan teknik *rolling hash*, maka didapatkan bahwa kedua hash yang saling dibandingkan akan menghasilkan substring yang sama, yaitu: "ompu" dan berlaku untuk perbandingan hasil hash berikutnya.

Digunakannya perhitungan operasi modulo agar tidak mempersulit system menghitung dalam jumlah banyak, selama nilai modulo yang digunakan tidak terlalu besar pula

Dimana:

b : Nilai bilangan basis (10)

k : Nilai ASCII karakter

h(k) : Nilai hash

m : Nilai bilangan prima (10007) L : Banyaknya karakter yang di-hashing

S(i) : Nilai hash awal

S(i+1) : Nilai hash berikutnya

2) Tahapan Penerapan Algoritma *Winnowing* Beberapa tahapan dalam penerapan

algoritma *Winnowing* adalah sebagai berikut:

1. Tahap Pertama : Membuang karakter yang tidak relevan seperti tanda baca, spasi, dan simbol-simbol lainnya.
2. Tahap Kedua : Membentuk rangkaian gram.
3. Tahap Ketiga : Melakukan proses rolling hash untuk mencari nilai hash dari setiap *gram*.
4. Tahap Keempat : Membentuk window yang terdiri dari nilai hash yang dihasilkan.
5. Tahap Kelima : Membentuk nilai finger print yang unik, dengan memilih nilai terendah dari setiap baris di dalam *window*.

Penilaian persentase similaritas antar dua dokumen yang dibandingkan adalah sebagai berikut:

1. Kategori Nihil (0%) Kedua dokumen tidak terindikasi plagiat karena benar-benar berbeda baik dari segi isi dan kalimat secara keseluruhan.
2. Kategori Sedikit Kesamaan (50%) Hasil uji menunjukkan lebih dari 50%, dapat dikatakan bahwa dokumen yang diuji mendekati tingkat plagiarisme
3. Kategori Plagiat Sedang (15-50%) Kedua dokumen terindikasi plagiat tingkat sedang.
4. Kategori Mendekati Plagiarisme (>50%) Hasil uji menunjukkan lebih dari 50%, dapat dikatakan bahwa dokumen yang diuji mendekati tingkat plagiarisme.
5. Kategori Plagiarisme (100%) Dokumen uji dapat dipastikan murni plagiat karena dari awal dan sampai akhir isi dokumen adalah sama.

Menurut penelitian Nur Alamsyah (2017) Algoritma *winnowing* melakukan

penghitungan nilai-nilai *hash* dari setiap *k-gram*, untuk mencari nilai *hash* selanjutnya digunakan fungsi *rolling hash*. Kemudian dibentuk *window* dari nilai-nilai *hash* tersebut. Dalam setiap window dipilih nilai hash minimum. Jika ada lebih dari satu *hash* dengan nilai minimum, dipilih nilai *hash* yang paling kanan. Kemudian semua nilai hash terpilih disimpan untuk dijadikan fingerprint dari suatu dokumen. Input dari proses document *fingerprinting* adalah file teks. Kemudian outputnya akan berupa sekumpulan nilai hash yang disebut *fingerprint*. *fingerprint* ini yang akan dijadikan dasar pembanding kesamaan antara teks yang telah dimasukkan.

Syarat dari algoritma deteksi plagiarisme:

a. *whitespace insensitivity*,

yaitu pencocokan teks file seharusnya tidak terpengaruh oleh spasi, jenis huruf kapital, tanda baca dan sebagainya.

b. noise surpression,

yaitu menghindari pencocokan teks file dengan panjang kata yang terlalu kecil atau kurang relevan dan bukan merupakan kata yang umum digunakan.

c. *position independence*,

yaitu pencocokan teks file seharusnya tidak bergantung pada posisi kata-kata sehingga kata dengan urutan posisi berbeda masih dapat dikenali jika terjadi kesamaan.

1. Langkah pertama

   Pembuangan Karakter yang Tidak Relevan.

   Yaitu penghapusan tanda baca, spasi dan symbol-simbol seperti =, #, %, &, (,), -, _, $, @, !, /,", seperti contoh dibawah ini:

   Aplikasi Deteksi Source Code C++

Akan dirubah menjadi

Aplikasideteksisourcecodec

2. Langkah kedua

   Pembentukan Rangkaian *n-gram*.

   Yaitu dengan cara membentuk rangkaian karakter sepanjang n dari hasil pembuangan karakter yang tidak relevan, dari teks diatas yang telah dibersihakan dengan ukuran k, ukuran k = 7 (gram terbaik pada penelitian sebelumnya) aplikas plikasi likasid ikaside kasidet asidete sidetek ideteks deteksi eteksis teksiso eksisou ksisour sisourc isource sourcec ourceco urcecod rcecode cecodec

3. Langkah ketiga Perhitungan Fungsi *Hash* untuk tiap *n-gram*. Yaitu melakukan perhitungan- perhitungan nilai-nilai *hash* dari setiap *gram*, fungsi yang digunakan untuk menghasilkan nilai *hash* dari rangkaian gram dalam algoritma *Winnowing* adalah *rolling hash*. *Rolling Hash* adalah suatu cara untuk mentransformasi sebuah string menjadi suatu nilai yang unik dengan panjang tertentu (*fixed-length*) yang berfungsi sebagai penanda string tersebut. Fungsi untuk menghasilkan nilai ini disebut fungsi *hash*, sedangkan nilai yang dihasilkan disebut nilai *hash*. Fungsi *hash* H(c1..ck) didefinisikan

   sebagai berikut:

   H(ck) =c1 * b(k-1) + c2 * b (k-2) +..... + ck * b (k-k)

   Keterangan :

   c = nilai ascii karakter

b = basis (bilangan prima)

k = banyak karakter

hasil rolling hash dari kalimat diatas adalah

26194 27766 27060 26674 26700 26210 27802 26394 26118 26866 28098

26734 27840 28486 27734 28786 28354 28492 27234 25482

Penelitian yang di lakukan oleh , Anton Yudhana (2018) Algoritma *winnowing* merupakan salah satu algoritma yang berfungsi sebagai document *fingerprint* atau algoritma yang digunakan untuk mendeteksi tindakan plagiarisme dengan menggunakan teknik *hashing*. Input dari algoritma *winnowing* berupa dokumen teks, dan akan menghasilkan keluaran berupa kumpulan nilai *hash* yang terbentuk dari peritungan ASCII pada setiap karakter.

Rolling hash adalah metode hashing yang digunakan untuk mencari nilai *hash* dari rangkaian grams yang telah terbentuk dan memberikan kemampuan untuk menghitung nilai tanpa mengulangi seluruh string. Nilai *hash* merupakan nilai numerik yang dibentuk dari kode ASCII[15].

Berikut rumus *rolling hash*:

H (C1..Cl)= C1 .b (l-1) + C2 . b (l-2)+ + C(l-1) . b + Cl (2)

H(c2….cl+1) =(H(c1….cl)– c1 * b(l-1)) * b + c(l-1) (3)

H (C1..Cl) = nilai hash

Cl = nilai ASCII karakter ke -1 pada string

l = panjang string

b = nilai basis hash

Perhitungan awal pada rangkaian *n-gram* paling awal dihitung menggunakan rumus nomer 2, dan rangkaian ngram berikutnya sampai rangkaian gram terakhir di hitung menggunakan rumus nomer 3, sehingga proses

akan jauh lebih cepat karna tidak menghitung lagi dari awal. Algoritma *winnowing* merupakan eksitensi dari algoritma *Rabin-karp*, proses yang dilakukan hampir sama hanya pada algoritma winnowing ditambahkan dengan konsep *window*.

Menurut Hargyo Tri Nugroho (2017) adalah algoritma untuk menghasilkan suatu deret bilangan unik (*fingerprint*) yang mewakili suatu dokumen. Dengan *fingerprint* tersebut kita bisa mengetahui tingkat kemiripan satu dokumen dengan dokumen yang lain.

Secara garis besar, algoritma *Winnowing* bekerja sebagai berikut:

1. Penghapusan karakter-karakter yang tidak relevan (*whitespace insensitivity*).
2. Pembentukan rangkaian gram dengan ukuran k.
3. Penghitungan nilai *hash*.
4. Membagi ke dalam *window* tertentu.
5. Pemilihan beberapa nilai *hash* menjadi *fingerprint* dokumen

Penelitian terkait Agus Setiawan (2017) Algoritma *winnowing* adalah salah satu algoritma pencocokan string. Pada pendeteksiannya, algoritma *winnowing* harus memenuhi kebutuhan mendasar.

Perhitungan Fungsi Hash untuk Setiap n-gram

Algoritma *winnowing* menggunakan *rolling hash* untuk menghitung nilai *hash* masing-masing rangkaian *gram*. Fungsi hash dengan *rolling hash* didefinisikan pada persamaan.

Pembentukan Window dari Nilai Hash

Algoritma *winnowing* tidak menggunakan semua nilai *hash* dari setiap rangkaian gram yang dibentuk. Nilai *hash* yang dibentuk pada tahap sebelumnya akan dibagi ke dalam *window* berukuran w. *Window* pertama berisi nilai *hash*

pertama sampai nilai hash ke-w. *Window* kedua dibentuk dari nilai *hash* kedua sampai nilai *hash* kew+1 dan seterusnya sampai terbentuk *window* dari seluruh nilai *hash*.

Pemilihan Fingerprint dari Setiap Window

Setelah terbentuk *window* dari seluruh nilai *hash*, tahap selanjutnya adalah menentukan nilai *fingerprint* teks. Nilai *fingerprint* ditentukan dengan memilih nilai *hash* terkecil dari setiap *window*.

Persamaan Jaccard Coeficient

Nilai *fingerprint* yang dibentuk dari algoritma *winnowing* digunakan untuk mengukur prosentase kemiripan teks pada persamaan, persamaan *Jaccard Coeficient*.

$similarity = jumla\ \ fingerpritnt\ sama\ total\ seluru\ \ fingerpritnt\ x$

100%

atau

$similarit(di, dj) = |w(di) \cap w(dj)|\ |w(di) \cup w(dj)|\ x$ 100%

Dengan di nilai-nilai fingerprint pada teks, dj nilai-nilai fingerprint pada teks w(di) ⋂ w(dj) jumlah nilai fingerprint sama antara teks ke-i dan teks ke-j dan w(di) ⋃ w(dj) adalah total nilai fingerprint teks ke-i dan teks ke-j.

Menurut penelitian Iif Alfiatul Mukaromah (2018) Algoritma winnowing merupakan salah satu algoritma yang berfungsi sebagai document fingerprint atau algoritma yang digunakan untuk mendeteksi tindakan plagiarisme dengan menggunakan teknik hashing. Input dari algoritma winnowing berupa dokumen teks, dan akan menghasilkan keluaran berupa kumpulan nilai hash yang terbentuk dari peritungan ASCII pada setiap karakter.

Jaccard Similarity.

Jaccard Similarity atau Jaccard Coefficient merupakan algoritma yang fungsinya untuk membandingkan dua sample yaitu dokumen yang satu dengan yang lainnya berdasarkan kata yang dimilikinya. Jaccard similarity biasanya digunakan untuk membandingkan dokumen dan menghitung nilai kemiripan (similarity) dari dua buah objek atau dokumen[5][10][11]. Jaccard similarity dapat dirumuskan sebagai berikut:

$Similarity\ (X, Y)$ = |X∩Y| |XUY|

Dimana:

X = Dokumen 1

Y = Dokumen 2

Rumus 1 merupakan dari rumus Jaccard Similarity atau Jaccard Coefficient yang digunakan untuk mencari persamaan dan perbedaan pada dua sample.

sebagai contoh diketahui X “Magister Teknik Informatika Yogyakarta”, dan Y “Magister Teknik Sipil Yogyakarta”. Maka akan menghasilkan nilainya:

$Similarity\ (X, Y)$ = |Magister Teknik Yogyakarta| |Magister Teknik Informatika Sipil Yogyakarta|

$Similarity\ (X, Y)$ = 3 5 = 0,6 x100% = 60%

Dari dua contoh sample di atas setelah dihitung kesamaannya menggunakan Jaccard similarity, bahwa kedua sample tersebut memiliki kesamaan atau kemiripan sebesar 60%. Di bawah ini merupakan program dari Jaccard similarity atau Jaccard Coefficient.

Menurut Muhammad Rasyidan (2019) goritma Winnowing. Algoritma Winnowing adalah sebuah cara yang digunakan untuk mendeteksi kesamaan kata/kalimat (common subsequence) dalam dua atau lebih teks yang

dibandingkan. Dua teks diketahui memiliki kesamaan kata/kalimat apabila di dalam dokumen tersebut dijumpai fingerprint, fingerprint inilah yang akan dijadikan dasar pembanding antara teks, algoritma ini akan mencari fingerprint (kesamaan di dalam dua teks) dengan cara mengubah n-gram dari sebuah teks ke dalam bentuk nilai angka yang disebut dengan nilai hash, teknik untuk mencari nilai tersebut adalah Hashing.

Pembentukan Window dari Nilai Hash:

Kelompokkan (windowing) untuk masing-masing hasil hash, langkahnya mirip seperti n-gram. Pembentukan window dari hasil perhitungan nilai hash dengan ukuran lebar window

Persamaan Jaccard Coeficient

Perhitungan kesamaan dengan menggunakan persamaan jaccard coefficient yaitu; Similarity (kemiripan) = 65/88 * 100% = 73.86 %

Jumlah Fingerprints pada teks judul 1 = 87

Jumlah Fingerprints teks judul 2 = 66 Union (Gabungan)

Fingerprints 1 dan 2 = 153 Intersection (fingerprints yang sama) = 65 (Union - Intersection) = 88

Prosentase Plagiarisme Koefisien Jaccard = (Intersection / (UnionIntersection) * 100 (65/88) * 100 = 73.86 %

Jadi dapat disimpulkan dari kedua judul diatas memiliki tingkat plagiarisme sebesar 73.86 %

Perhitungan Fungsi Hash untuk Setiap n-gram

Buat Rolling Hash untuk masing-masing N-Gram , Perhitungan nilai hash pada rangkaian n-gram bagian pertama “apl” dengan nilai basis

(b) = 2, panjang rangkaian ngram(n) = 3

H(apl) =asci (a)* 2 3 + asci (p)* 2 2 + asci (l) * 2 1

= 97 *8 + 112*4 + 108 * 2

= 1440

Berdasarkan penelitian terkait , Jayanti Yusma Sari (2018) Implementasi dari algoritma Winnowing membutuhkan masukan berupa file teks dan menghasilkan keluaran berupa nilai hash yang disebut finger print [2]. Setiap kata yang terkandung dalam file teks diubah terlebih dahulu menjadi sebuah kumpulan nilai hash dengan teknik rolling hash. Nilai hash merupakan nilai numerik dari perhitungan ASCII untuk setiap karakter. Lalu kumpulan nilai hash yang disebut finger print tersebut digunakan untuk mendeteksi kemiripan antar dokumen

Pada dasarnya sistem pendeteksian haruslah memiliki 3 unsur utama yang harus dipenuhi, seperti:

1. *Whitespace insensitivity*, sistem pencocokan teks seharusnya tidak terpengaruh pada spasi, adawnya huruf kapital, berbagai tanda baca, dan sebagainya;
2. *Noise surpression*, sistem haruslah menghindari pencocokan kata yang terlalu pendek;
3. *Position independence*, sistem seharusnya tidak bergantung pada posisi kata yang dicari sehingga apabila ditemukan kata yang terindeksi sama dengan posisi berbeda masih dapat dikenali;

Penilaian persentase similaritas antar dua dokumen yang dibandingkan adalah sebagai berikut:

1. Kategori Nihil (0%) Kedua dokumen tidak terindikasi plagiat karena benar-benar berbeda baik dari segi isi dan kalimat secara keseluruhan.
2. Kategori Sedikit Kesamaan (50%) Hasil uji menunjukkan lebih dari 50%,

dapat dikatakan bahwa dokumen yang diuji mendekati tingkat plagiarisme.

3. Kategori Plagiat Sedang (15-50%) Kedua dokumen terindikasi plagiat tingkat sedang.
4. Kategori Mendekati Plagiarisme (>50%) Hasil uji menunjukkan lebih dari 50%, dapat dikatakan bahwa dokumen yang diuji mendekati tingkat plagiarisme.
5. Kategori Plagiarisme (100%) Dokumen uji dapat dipastikan murni plagiat karena dari awal dan sampai akhir isi dokumen adalah sama.

Untuk mengurangi waktu komputasi, dapat dilakukan teknik rolling hash dengan mengambil waktu sebesar O(p) sehingga didapatkan banyak kecocokan. Contoh, lakukan hashing 5 substring pada kata “komputer”. Hash I: “kompu‟, hash II: “omput‟, dan seterusnya. Dengan teknik rolling hash, maka didapatkan bahwa kedua hash yang saling dibandingkan akan menghasilkan substring yang sama, yaitu: “ompu‟ dan berlaku untuk perbandingan hasil hash berikutnya.

Digunakannya perhitungan operasi modulo agar tidak mempersulit system menghitung dalam jumlah banyak, selama nilai modulo yang digunakan tidak terlalu besar pula.

Menurut , Pasnur (2017) Algoritma Winnowing adalah algoritma yang digunakan untuk membuat fingerprint dari sebuah dokumen [3]. Algoritma winnowing mengubah dokumen teks menjadi sekumpulan nilai hash yang disebut sebagai fingerprint. Proses untuk menghasilkan fingerprint dari sebuah dokumen adalah sebagai berikut:

1. Membuang karakter-karakter yang tidak relevan seperti spasi, tanda baca maupun artikel atau kata yang tidak penting, misalnya kata sambung. Langkah ini sesuai dengan syarat algoritma pendeteksi penjiplakan yaitu *whitespace insensitivity* dan *noise suppression*.
2. Membentuk rangkaian n-gram dari teks. Semisal ditentukan n = 6, maka akan dihasilkan sekumpulan gram atau string yang berukuran 6.
3. Melakukan fungsi hash untuk setiap gram. Persamaan (1) adalah perhitungan fungsi hash dari algoritma winnowing

   $H\ c1\ldots ck = c1 * b\ k-1 + c2 * b\ k-2 + \cdots + ck-1 * b\ k + ck\ H\ c2\ldots ck+1 =$

   $H\ c1\ldots ck - c1 * b\ k-1 * b + ck+1\ \ldots .$

   H adalah nilai hash, c adalah karakter pada gram, b adalah bilangan basis, dan k adalah jumlah karakter gram.
4. Membuat himpunan-himpunan yang disebut window yang terdiri dari i nilai hash. Jika i = 6, maka di dalam satu window terdapat 6 nilai hash.
5. Memilih fingerprint dari hasil hashing dengan pembagian hasil hash berdasarkan satu nilai window w, dan kemudian dipilih nilai hash terkecil dari setiap window tersebut.

Langkah b – e, berkaitan dengan syarat *Position Independence*. Dengan membentuk fingerprint dari sebuah dokumen teks, maka dapat dilakukan komparasi dengan fingerprint dokumen lainnya tanpa menghiraukan posisi dari

fingerprint tersebut.

Pada dasarnya sistem pendeteksian haruslah memiliki 3 unsur utama yang harus dipenuhi, seperti:

4. *Whitespace insensitivity*, sistem pencocokan teks seharusnya tidak terpengaruh pada spasi, adanya huruf kapital, berbagai tanda baca, dan sebagainya;
5. *Noise surpression*, sistem haruslah menghindari pencocokan kata yang terlalu pendek;
6. *Position independence*, sistem seharusnya tidak bergantung pada posisi kata yang dicari sehingga apabila ditemukan kata yang terindeksi sama dengan posisi berbeda masih dapat dikenali;

II.2.2 NetBeans

Menurut Nofriadi dalam arizona & Kaunen (2017) "*Netbeans* adalah IDE (*Integred Development Enviroment*) yang baerbasis Java dari *Sun Microsystems* yang berjalan diatas *swing* dan banyak digunakan sebagai editor untuk berbagai pemrograman".

II.2.3 Program

Menurut Sutarman dalam (Maarif et al., 2017) menyatakan "program adalah barisan perintah atau instruksi yang di susun sehingga dapat di pahami oleh komputer dan kemudian di jalankan sebagai barisan perhitungan numerik, dimana barisan perintah tersebut berhingga, berakhir, dan menghasilkan output".

II.2.4 UML

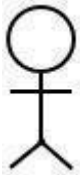
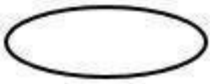
(Menurut Suhendri, 2018) *Unified Modeling Language* (UML) merupakan bahasa pemodelan perangkat lunak atau sistem dengan konsep pemrograman berorientasi objek yang dapat analisa dan menjabarkan secara rinci apa yang diperlukan sistem.

Alat bantu yang digunakan dalam perancangan berorientasi objek berbasiskan UML adalah sebagai berikut:

1. Use case Diagram

Use case diagram merupakan pemodelan untuk kelakukan (*behavior*) sistem informasi yang akan dibuat. *Use case* mendeskripsikan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Dapat dikatakan *use case* digunakan untuk mengetahui fungsi apa saja yang ada di dalam sistem informasi dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut. Simbol-simbol yang digunakan dalam *use case* diagram dapat dilihat pada tabel II.1 dibawah ini :

Tabel II.1. Simbol *Use Case*

Simbol	Nama	Keterangan
	*Actor*	Merupakan peran orang, sistem yang lain, atau alat ketikaberhubungan dengan *use case*.
	*Use Case*	Abstraksi dari penghubung antara aktor dengan *Use case*.
	Association	Abstraksi dari penghubung antara aktor dengan *use case*.
	Generalization	Menunjukkan spesialisasi aktor untuk dapat berpartisipasi dengan *use case*.
	Include	Menunjukkan bahwa suatu *use case* seluruhnya merupakan fungsionalitas dari *use case* lainnya.
	Extend	Menunjukkan bahwa suatu *use case* merupakan tambahan fungsional dari *use case* lainnya jika suatu kondisi terpenuhi.

(Sumber :* Yunahar Heriyanto*: 2018)

2. Diagram Aktivitas (*Activity Diagram*)

Activity Diagram menggambarkan *workflow* (aliran kerja) atau aktivitas dari sebuah sistem atau proses bisnis. Simbol-simbol yang digunakan dalam *activity diagram* dapat dilihat pada tabel II.2:

Tabel II.2. Simbol *Activity Diagram*

Simbol	**Nama**	**Keterangan**
	Status awal	Sebuah diagram aktivitas memiliki sebuah status awal
	Aktivitas	Aktivitas yang dilakukan sistem, aktivitas biasanya diawali dengan kata kerja.
	*Decision*/percabangan	Percabangan dimana ada pilihan aktivitas yang lebih dari satu.
	Join/penggabungan	Penggabungan yang mana lebih dari satu aktivitas lalu digabungkan jadi satu.

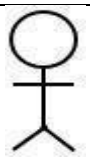
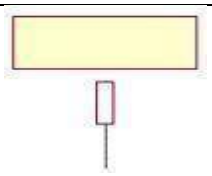
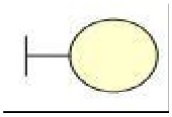
	Status akhir	Status akhir yang dilakukan sistem, sebuah diagram aktivitas memiliki sebuah status akhir.
	*Swimlane*	*Swimlane* memisahkan organisasi bisnis yang bertanggung jawab terhadap aktivitas yang terjadi.

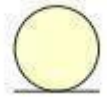
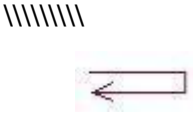
(Sumber :* Yunahar Heriyanto*: 2018)

3. Diagram Urutan (*Sequence Diagram*)

Sequence diagram menggambarkan kelakuan objek pada *use case* dengan mendeskripsikan waktu hidup objek dan pesan yang dikirimkan dan diterima antar objek. Simbol-simbol yang digunakan dalam *sequence diagram* dapat dilihat pada tabel II.3:

Tabel II.3. Simbol *Sequence Diagram*

Simbol	**Nama**	**Keterangan**
	*Actor*	Mempresentasikan entitas yang berada diluar sistem dan berinteraksi dengan sistem
	*Lifeline*	Mengubungkan objek selama *sequence* (message dikirim atau diterima dan aktifasinya).
	*General*	Merepresentasikan entitas tunggal dalam *sequence diagram.*
	*Bondary*	Berupa tepi dari sistem, seperti *user interface* atau suatu alat yang berinteraksi dengan sistem yang lain.

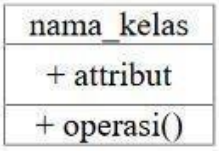
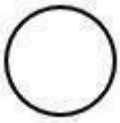
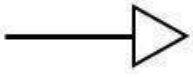
	*Control*	Elemen mengatur aliran dari informasi untuk sebuah skenario. Objek ini umumnya mengatur perilaku bisnis.
	*Entitas*	Elemen yang bertanggung jawab menyimpan data atau informasi. Ini dapat berupa *beans* atau model objek.
	*Activation*	Suatu titik dimana objek mulai berpartisipasi di dalam sebuah *sequence* yang menunjukkan kapan sebuah objek mengirim atau menerima objek.
	*Message*	Berfungsi sebagai komunikasi antar objek yang menggambarkan aksi yang akan di lakukan.
	*Message Entry*	Berfungsi untuk menggambarkan pesan/hubungan antar objek yang menunjukkan urutan kejadian yang terjadi.
	*Message to Self*	Simbol ini menggambarkan pesan/hubungan objek itu sendiri, yang menunjukkan urutan kejadian yang terjadi.
	*Message Return*	Menggambarkan hasil dari pengiriman message dan digambarkan dengan arah dari kanan dan ke kiri.

(Sumber :* Yunahar Heriyanto*: 2018)

4. Diagram Kelas (*Class Diagram)*

Class diagram juga menunjukkan atribut-atribut dan operasi-operasi dari sebuah kelas dan *constraint* yang berhubungan dengan objek yang dikoneksikan. *Class diagram* secara khas meliputi: Kelas (*Class*), Relasi, *Associations*, *Generalization* dan *Aggregation*, Atribut (*Attributes*), Operasi (*Operations/ Method*), *Visibility*, tingkat akses objek eksternal kepada suatu operasi atau atribut. Hubungan antar kelas mempunyai keterangan yang disebut dengan *multiplicity* atau kardinaliti. Simbol *class diagram* dan *multiplicity class diagram* dapat dilihat pada Tabel II.4 dan Tabel II.5. dibawah ini :

Tabel II.4. Simbol *Class Diagram*

Simbol	**Nama**	**Keterangan**
	*Class*	Kelas pada struktur sistem.
	*Interface*	Sama dengan konsep *interface* dalam pemrograman berorientasi objek.
	Association	Relasi antar*class* dengan arti umum, asosiasi biasanya juga disertai dengan *Multiplicity.*
	*DirectedAssociation*	Relasi antarkelasdengan makna kelas yang atau digunakan oleh kelas yang lain, asosiasi iassanya juga disertai dengan *multiplicity.*
	*Generalisasi*	Relasi antarkelas dengan makna *generalisasi spesialisasi* (umum khusus).

	*Dependency*	Relasi antarkelas dengan makna kebergantungan antarkelas.
	*Aggregation*	Relasi antarkelas dengan dengan makna semua-bagian *(whole-part).*

(Sumber :* Yunahar Heriyanto*: 2018)

Tabel II.5. *Multiplicity Class Diagram*

Multiplicity	Penjelasan
1	Satu dan hanya satu
0..*	Boleh tidak ada atau 1 atau lebih
1..*	1 atau lebih
0..1	Boleh tidak ada, maksimal 1
n..n	Batasan antara. Contoh 2..4 mempunyai arti minimal 2 maksimum 4

(Sumber :* Yunahar Heriyanto*: 2018)