Sistem Pendeteksi Plagiarisme Proposal Tugas Akhir Mahasiswa Dengan Algoritma Rabin Karp
(Studi Kasus: Jurusan Teknik Informatika Universitas Halu Oleo)

Nur'Aziza Tadjuddin[*1], Bambang Pramono[2], L.M Tajidun[3]
[1,2,3] Jurusan Teknik Informatika, Fakultas Teknik, Universitas Halu Oleo, Kendari
e-mail: [*1]nurazizatadjuddin@gmail.com [2] Bambang.pramono@uho.ac.id,[3] lmtajidun@uho.ac.id

ABSTRAK

Plagiarisme sering di jumpai dalam sektor akademis maupun non akademis, Dalam sektor akademis, plagiarisme di anggap sebagai tindak pidana serius karena di anggap pengambilan karangan, pendapat, ide dan gagasan orang lain. Salah satu plagiarisme yang sering terjadi di kalangan mahasiswa adalah plagiarisme di dalam penulisan Proposal tugas akhir. Proposal tugas akhir mahasiswa bertujuan untuk menyampaikan rancangan penelitiannya kepada dosen pembimbing agar disetujui sebelum membuat skripsi sehingga sistem ini dapat di gunakan oleh dosen pembimbing untuk mendeteksi tingkat plagiarisme proposal mahasiswa yang melakukan bimbingan sebelum melakukan ujian seminar proposal. Tindakan plagiarisme secara perlahan harus di cegah dan dihilangkan dengan melakukan pendeteksian plagiat secara manual maupun dengan memanfaatkan metode pencocokan string. Dengan demikian melakukan pendeteksian plagiarisme secara manual sangat tidak efektif sehingga Algoritma Rabin Karp dapat menjadi solusi untuk menyelesaikan hal tersebut. Algoritma Rabin Karp merupakan algoritma pencarian string yang menggunakan fungsi hashing untuk membandingkan string yang dicari (m) dengan string yang dibandingkan (n).

Kata kunci— *Kata kunci : Proposal tugas akhir dan Algoritma Rabin-Karp.*

ABSTRACT

Plagiarism is often encountered in the academic and non-academic sectors. In the academic sector, plagiarism is considered a serious crime because it is considered taking other people's essays, opinions, ideas, and ideas. One of the plagiarism that often occurs among students is plagiarism in writing the final project proposal. The student's final project proposal aims to submit the research design to the supervisor for approval before making the thesis so that this system can be used by the supervisor to detect the level of plagiarism in student proposals who conduct guidance before conducting the proposal seminar exam. Plagiarism must be slowly prevented and eliminated by detecting plagiarism manually or by using string matching methods. Thus, manual plagiarism detection is not very effective, so the Rabin Karp Algorithm can be a solution to solve this problem. Rabin Karp algorithm is a string search algorithm that uses a hashing function to compare the searched string (m) with the compared string (n).

Keywords: *Final project proposal and Rabin-Karp Algorithm.*

I. PENDAHULUAN

Informasi dapat beredar dengan cepat dan luas dari berbagai belahan dunia melalui Internet. Kemudahan akses internet yang sudah bisa dinikmati di mana saja merupakan salah satu faktor utama yang menyebabkan informasi dapat dengan mudah diperoleh. Namun, dikarenakan kemudahan informasi yang didapat tersebut, tidak sedikit tindak kejahatan yang terjadi di dunia maya contohnya adalah plagiarisme.

Plagiarisme sering di jumpai dalam sektor akademis maupun non akademis. Salah satu faktor pemicu plagiarisme adalah penulis (mahasiswa) ingin segera menyelesaikan skripsinya agar bisa meraih gelar akademik secepatnya tanpa harus bekerja keras sesuai proses riset dan penulisan ilmiah yang benar. Salah satu plagiarisme yang sering terjadi di kalangan mahasiswa adalah plagiarisme di dalam penulisan Proposal tugas akhir.

Proposal merupakan sebuah rancangan dan rencana kerja yang sistematis dan terperinci. Pada umumnya sebuah proposal bertujuan untuk menyampaikan rencana kegiatan pada pihak terkait, sehingga kegiatan tersebut dapat diterima dengan tujuan mendapatkan dukungan, mendapatkan izin, memperoleh dana dan sponsor, dan sebagainya. Sama halnya dengan proposal skripsi, yang mana mahasiswa bertujuan untuk menyampaikan rancangan penelitiannya kepada dosen pembimbing agar disetujui sebelum membuat skripsi sehingga sistem ini dapat di gunakan oleh dosen pembimbing untuk mendeteksi tingkat plagiarisme proposal mahasiswa yang melakukan bimbingan sebelum melakukan ujian seminar proposal. Dengan demikian melakukan pendeteksian plagiarisme secara manual sangat tidak efektif sehingga Algoritma Rabin Karp dapat menjadi solusi untuk menyelesaikan hal tersebut.

Algoritma Rabin Karp merupakan algoritma pencarian string yang ditemukan oleh Michael Rabin dan Richard Karp. Algoritma Rabin Karp merupakan algoritma pencarian string yang menggunakan fungsi hashing untuk membandingkan string yang dicari (m) dengan string yang dibandingkan (n). Fungsi hash adalah fungsi matematis yang digunakan untuk mengubah data menjadi bilangan bulat yang relatif kecil yang dapat berfungsi sebagai indeks pada array. Fungsi hashing merupakan proses inti pada perhitungan algoritma Rabin Karp. Hashing merupakan representasi ASCII yang menggantikan atau mentransformasikan karakter atau tanda baca menjadi sebuah nilai atau angka [1].

II. METODE PENELITIAN

2.1 Pengertian Plagiarisme

Plagiarisme atau sering disebut plagiat adalah penjiplakan atau pengambilan karangan, pendapat, dan sebagainya dari orang lain dan menjadikannya seolah karangan dan pendapat sendiri. Plagiat dapat dianggap sebagai tindak pidana karena mencuri hak cipta ggorang lain tanpa meminta izin dan menyertakan sumber yang dicatutnya. Sejak abad ke-19, plagiat atau plagiarisme telah menjadi masalah serius dalam dunia pendidikan yang masih berlangsung sampai saat ini. Ini tentu memerlukan pertimbangan khusus karena memiliki dampak yang tidak sehat dalam dunia pendidikan [2].

2.2 Information Retrieval (IR)

Information retrieval adalah studi tentang sistem pengindeksan, pencarian, dan mengingat data, khususnya teks atau bentuk tidak teratur lainnya. Pada proses tahap indexing didalam *Information retrieval* terdapat proses *stemming* yaitu proses mengubah suatu kata bentukan menjadi kata dasar. Proses *stemming* sangat tergantung kepada bahasa dari kata yang akan di stem. Hal ini dikarnakan, dalam melakukan proses stemming harus mengaplikasikan aturan morfologikal dari suatu bahasa. Bahasa Indonesia memiliki kata berimbuhan yang lebih kompleks dibandingkan dengan Bahasa Lainnya. Terdapat dua pendekatan dalam proses stemming yaitu: *Light Stemming* dan *Dictionary Base Stemming*. pendekatan dengan proses *Dictionary Base Stemming* dapat memberikan solusi untuk men-stemm kata berimbuhan dalam Bahasa Indonesia, karena menggunakan struktur morfologi untuk mengekstrak kata berimbuhan menjadi kata dasar (*root word*). Stemming adalah metote yang digunakan untuk menghilangkan imbuhan dari sebuah kata. *Stemming* merupakan salah satu tahapan dalam *pre processing*. Memiliki pengaruh terhadap tingkat relevansi temu kembali informasi

seperti dalam pencarian informasi berdasarkan *query*. Untuk mendapatkan kata dasar beberapa imbuhan yang dihilangkan adalah awalan, akhiran dan sisipan [3].

2.3 *American Standard Code for Information Interchange* (ASCII)

Kode Standar Amerika untuk Pertukaran Informasi atau ASCII (*American Standard Code for Information Interchange*) merupakan suatu standar internasional dalam kode huruf dan simbol seperti *Hex* dan *Unicode* tetapi ASCII lebih bersifat universal, contohnya 124 adalah untuk karakter "|". Itu selalu digunakan oleh komputer dan alat komunikasi lain untuk menunjukkan teks. Kode ASCII sebenarnya memiliki komposisi bilangan biner sebanyak 7 bit. Namun, ASCII disimpan sebagai sandi 8 bit dengan menambakan satu angka 0 sebagai bit significant paling tinggi. Bit tambahan ini sering digunakan untuk uji prioritas. Karakter control pada ASCII dibedakan menjadi 5 kelompok sesuai dengan penggunaan yaitu berturut-turut meliputi *logical communication, device control, information separator, code extention, dan physical communication*. Jumlah kode ASCII adalah 255 kode [4]

2.4 Algoritma Rabin Karp

Metode Algoritma Rabin Karp adalah algoritma pencarian kata yang mencari sebuah pola berupa substring dalam sebuah teks menggunakan hashing. Algoritma ini sangat efektif untuk pencocokan kata dengan pola banyak. Salah satu aplikasi praktis dari Algoritma Rabin Karp adalah dalam pendeteksian plagiarisme. Dalam ilmu komputer, Algoritma Rabin-Karp adalah algoritma pencarian string yang dibuat oleh Michael O. Rabin dan Richard M. Karp pada tahun 1987 yang menggunakan hashing untuk menemukan salah satu dari satu set string pola dalam teks *Hashing* merupakan representasi ASCII yang menggantikan atau mentransformasikan karakter atau tanda baca menjadi sebuah nilai atau angka.

Proses perhitungan algoritma rabin karp terdiri atas beberapa langkah sebagai berikut:

1. Tahap input parameter
   Proses sistem yang akan dibuat dimulai dari memasukkan dokumen yaitu dokumen asli dan dokumen uji.
2. Tahapan Preprocessing

Pada tahap preprocessing terdapat beberapa proses yang dilakukan oleh sistem terhadap dokumen yang diinputkan. Proses-proses tersebut antara lain case folding (mengubah huruf kapital menjadi huruf kecil) filtering proses membuang kata yang tidak penting (stop word) dan membuang tanda baca, selanjutnya *stemming*, yaitu proses mengubah suatu kata bentukan menjadi kata dasar dan Tokenizing. Pada proses Tokenizing di bagi menjadi dua sub proses yaitu proses parsing k-gram dan proses hashing.

a. Proses Parsing K-gram

Pada proses ini kata dipecah menjadi potongan-potongan, setiap potongan mengandung karakter sebanyak k. Nilai k-gram sangat berpengaruh terhadap nilai persentase suatu kemiripan dokumen. pemilihan k-gram yang semakin kecil akan memperoleh nilai persentase kemiripan yang besar.

b. Proses Hashing

Pada proses ini hash berfungsi untuk mengkonvert setiap string menjadi bilangan. Dengan cara mengalikan nilai ASCII masing-masing huruf hasil k-gram dengan basis bilangan tertentu, dimana Dalam prosesnya digunakan basis yang biasanya adalah bilangan prima yang cukup besar, dengan tujuan agar meminimalkan terjadinya tabrakan. Persamaan proses hashing :

$$H_{(C1 \ldots CK)} = C1.b^{(K-1)} + C2.b^{(K-2)} + C3.b^{(k-1)} + C4.b^{(k-1)} + (C5.b^{(k-1)}).$$

3. Deteksi Kesamaan

Proses selanjutnya adalah menghitung similarity yaitu tingkat kesamaan dua dokumen dengan menggunakan persamaan berikut:

$$\text{similarity (asli, uji)} = \frac{2 * \sum H\ asli \cap H\ uji}{\sum H\ asli + \sum H\ uji} \times 100\ \%$$

2.5 *Hypertext Preprocessor* (PHP)

PHP Kepanjangan dari PHP adalah "*Hypertext Preprocessor*" (ini merupakan singkatan rekursif). PHP adalah bahasa *scriptingweb* HTML-*embedded*. Ini berarti kode PHP dapat disisipkan ke dalam HTML halaman Web. Ketika sebuah halaman PHP diakses, kode PHP dibaca atau"diurai" oleh server. Output dari fungsi PHP pada halaman biasanya dikembalikan sebagai kode HTML, yang dapat dibaca oleh *browser*. Karena kode

PHP diubah menjadi HTML sebelum halaman dibuka, pengguna tidak dapat melihat kode PHP pada halaman. Ini membuat halaman PHP cukup aman untuk mengakses database dan informasi aman lainnya. Banyak sintaks PHP yang hasil adaptasi dari bahasa lain seperti bahasa C, Java dan Perl. Namun, PHP memiliki sejumlah fitur unik dan fungsi tertentu juga. Tujuan dari bahasa pemrograman PHP adalah untuk memungkinkan pengemban web untuk menulis halaman yang dihasilkan secara dinamis dengan cepat dan mudah. PHP juga bagus untuk menciptakan situs Web *database-driven* .

2.6 Website

Website dapat diartikan sebagai kumpulan halaman yang digunakan untuk menampilkan informasi berupa teks, gambar diam atau gambar gerak, animasi, suara, dan atau gabungan dari semuanya baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang dihubungkan dengan menggunakan jaringan halaman. *Hyperlink* adalah hubungan antara satu halaman *web* dengan halaman *web* yang lainnya, sedangkan *hypertext* adalah teks yang dijadikan media penghubung.

2.7 *HyperText Markup Language* (HTML)

HTML yang merupakan singkatan dari *HyperText Markup Language* adalah serangkaian kode program yang merupakan dasar dari representasi visual sebuah halaman Web. Didalamnya berisi kumpulan informasi yang disimpan dalam tag-tag tertentu, dimana tag-tag tersebut digunakan untuk melakukan format terhadap informasi yang dimaksud. Berbagai pengembangan telah dilakukan terhadap kode HTML dan telah melahirkan teknologi-teknologi baru di dalam dunia pemrograman web. Kendati demikian, sampai sekarang HTML tetap berdiri kokoh sebagai dasar dari bahasa web seperti PHP, ASP, JSP dan lainnya. Bahkan secara umum, mayoritas situs web yang ada di Internet pun masih tetap menggunakan HTML sebagai teknologi utama.

2.8 *Database Management System* (DBMS)

Database Management System (DBMS) merupakan perangkat lunak untuk mengendalikan pembuatan, pemeliharaan, pengolahan, dan penggunaan data yang berskala besar. Penggunaan DBMS saat ini merupakan hal yang sangat penting dalam segala aspek, baik itu dalam skala yang besar atau kecil. Sebagai contoh media sosial facebook menggunakan DBMS untuk menyimpan data-data pengguna facebook yang sangat banyak kedalam DBMS MySQL. Beberapa DBMS yang digunakan adalah MySQL dan MariaDB. Berdasarkan survey yang dilakukan, MySQL dan MariaDB merupakan DBMS yang banyak digunakan sebagai contoh survey yang terdapat pada *db-engines.com, db-Engines Ranking* menempatkan MySQL pada posisi ke-2 sedangkan MariaDB pada posisi ke-20 namun pada *survey* yang terdapat di *serverwatch*.com *Top 10 Enterprise Database Sistem Of 2016*, MariaDB menempati posisi ke-6 dan MySQL menempati posisi ke-7.

2.9 *Unified Modeling Language* (UML)

UML adalah singkatan dari *Unified Modelling Language*. UML adalah bahasa pemodelan untuk sistem atau perangkat lunak yang berparadigma berorientasi objek. Abstraksi konsep dasar UML terdiri atas *structural classification, dynamic behavior,* dan model *management* dapat dipahami main *concepts* sebagai term yang akan muncul pada saat membuat diagram dan *view* adalah kategori dari diagram tersebut. UML terdiri tas diagram-diagram yaitu *Use Case diagram, Class diagram, Statechart diagram, Activity diagram*, *Sequence diagram*, *Collaboration diagram*, *Component diagram*, dan *deployment diagram* [5].

III. HASIL DAN PEMBAHASAN

3.1 Data

Data yang digunakan dalam penelitian ini yakni skripsi mahasiswa jurusan Teknik informatika Pengetahuan Alam Universitas Halu Oleo.

3.2 Penerapan Perhitungan Manual Algoritma Rabin Karp

Berikut ini adalah contoh perhitungan kesamaan dokumen yang akan di proses dengan menggunakan metode *Rabin Karp*.

1. Tahap Input Parameter

Proses sistem yang akan dibuat dimulai dari memasukkan dokumen yaitu dokumen asli dan dokumen uji.

Dokumen asli yang di input :

Komputer adalah perangkat “Elektronik”

Dokumen uji yang di input :

Komputer adalah perangkat yang membutuhkan listrik.

Dengan *K-gram* k= 4
Basis bilangan b= 11

2. Tahapan *Preprocessing*

Tahap *Preprocessing* harus dilalui untuk menentukan *keyword* pada kedua dokumen yang akan dilakukan pengujian, yaitu dokumen asli dan dokumen uji. Pada tahap *preprocessing* terdapat beberapa proses yang dilakukan oleh sistem terhadap dokumen yang diinputkan. Proses-proses tersebut antara lain *case folding, filtering, Tokenizing.*

a. *S*ub Proses *Case Folding*

Proses *case folding* (mengubah huruf kapital menjadi huruf kecil) merupakan tahap pertama yang akan dilakukan dari rangkaian tahapan yang terdapat pada *preprocessing.* Berikut adalah contoh proses *case folding*:

Dokumen asli setelah melalui proses *case folding* (proses 1.1)

“komputer adalah perangkat “elektronik”

dokumen uji setelah melalui proses *case folding* (proses 1.1)

Komputer adalah perangkat yang membutuhkan listrik.

Pada contoh proses *case folding* diatas, kalimat yang terdapat huruf kapitalnya adalah pada kata “Komputer”. Dimana pada kata tersebut setelah dilakukan proses *case folding* akan diubah menjadi huruf kecil. Pada kata “Komputer” akan berubah menjadi “komputer”.Sehingga secara keseluruhan kata-kata pada dokumen yang terdapat huruf kapitalnya akan berubah menjadi huruf kecil semua.

b. Sub Proses *Filtering*

Setelah proses case folding, dokumen selanjutnya masuk ketahap *filtering*, yaitu proses membuang kata yang tidak penting (stop word) dan membuang tanda baca. Berikut ini contoh proses *filtering*:

Dokumen asli setelah melalui proses *filtering* (proses 1.2)

komputer perangkat elektronik

Dokumen uji setelah melalui proses *filtering* (proses1.2)

komputer perangkat membutuhkan listrik

Pada contoh filtering diatas yaitu proses menghilangkan tanda baca dan menghilangkan *stop word.* Pada contoh dokumen asli tanda baca yang di hilangkan adalah tanda petik (“ “) dan Pada contoh dokumen uji tanda baca yang di hilangkan adalah tanda titik (.) sedangkan kata yang di anggap tidak penting dan dihilangkan pada dokumen asli yaitu kata “adalah” dan pada dokumen uji yaitu kata “adalah” dan kata “yang”.

c. Sub Proses *Tokenizing*

Setelah dilakukan proses *filtering*, selanjutnya adalah proses *tokenizing*, proses ini merupakan proses pembentukan pola kata, dimana pola katanya dalam bentuk gram dengan panjang k. Pada proses *Tokenizing* di bagi menjadi dua sub proses yaitu proses parsing *k-gram* dan proses *hashing*.

1. Proses Parsing *K-gram*

Langkah selanjutnya adalah parsing *k-gram*, K-gram adalah rangkaian terms dengan panjang K. Kebanyakan yang digunakan sebagai terms adalah kata, dimana pada proses ini kata dipecah menjadi potongan-potongan, setiap potongan mengandung karakter sebanyak k. Nilai *k-gram* sangat berpengaruh terhadap nilai persentase suatu kemiripan dokumen. pemilihan k-gram yang semakin kecil akan memperoleh nilai persentase kemiripan yang besar. Hal ini terjadi karena

pada k-gram yang lebih sedikit, string yang dipotong lebih kecil sehingga kemungkinan untuk ditemukannya rangkaian karakter yang sama semakin besar. Semakin besarnya *k-gram*, maka mengandung karakter yang lebih banyak dibandingkan dengan *k-gram* yang lebih kecil sehingga menyebabkan rangkaian karakter yang ditemukan semakin berkurang sehingga menurunkan nilai similaritas. pada sistem ini nilai *k-gram* nya telah di tetapkan yaitu nilai *k-gram* = 4.

Hasil *K-gram* dokumen asli yang telah melalui proses tokenisasi (proses 2.1)

{komp} {ompu} {mput} {pute} {uter} {terp} {erpe} {rper} {pera} {eran} {rang} {angk} {ngka} {gkat} {kate} {atel} {tele} {elek} {lekt} {ektr} {ktro} {tron} {roni} {onik}

Hasil *K-gram* dokumen uji yang telah melalui proses tokenisasi (proses 2.1)

{komp} {ompu} {mput} {pute} {uter} {terp} {erpe} {rper} {pera} {eran} {rang} {angk} {ngka} {gkat} {katb} {atbu} {tbut} {butu} {utuh} {tuhl} {uhli} {hlis} {list} {istr} {stri} {trik}

Pada contoh parsing *K-gram* diatas adalah proses pembentukan pola kata dalam bentuk gram dengan panjang karakter k=4, sehinga isi kalimat pada dokumen asli dan dokumen uji dirubah dalam bentuk *k-gram*.

2. Proses *Hashing*

Kemudian dilakukan proses *hashing*, dimana pada proses ini hash berfungsi untuk mengkonvert setiap *string* menjadi bilangan. Dengan cara mengalikan nilai ASCII masing-masing huruf hasil *k-gram* dengan basis bilangan tertentu, dimana Dalam prosesnya digunakan basis yang biasanya adalah bilangan prima yang cukup besar, dengan tujuan agar meminimalkan terjadinya tabrakan. Dengan menggunakan persamaan, maka dapat dihitung hasil hashnya. Algoritma *Rabin Karp* didasarkan pada fakta jika dua buah *string* sama maka nilai hash valuenya pasti sama. Sebagai contoh kita ambil kata dari hasil *k-gram* yang pertama pada dokumen asli, yaitu kata {komp}.

Contoh proses hashing untuk menghitung nilai hash dari kata {komp}, dengan nilai **k= 4 dan b=11**

Nilai ASCII dari kata {komp}

ASCII(k)= 107

ASCII(o)= 111

ASCII(m)= 109

ASCII(p)= 112

$H_{(C1 \ldots CK)} = C_1.b^{(K-1)} + C_2.b^{(K-2)} + C_3.b^{(k-1)} + C_4.b^{(k-1)} + (C_5.b^{(k-1)})$

$H = (107 \times 11^3) + (111 \times 11^2) + (109 \times 11^1) + (112 \times 11^0)$

$H = (107 \times 1331) + (111 \times 121) + (109 \times 11) + (112)$

$H = 142.417 + 13.431 + 1.199 + 112$

$H = 157.159$

Jadi nilai hash pada dokumen asli *k-gram* yang pertama adalah 157159, proses perhitungan nilai *hash* di ulang kembali hingga *k-gram* keseluruhan dihitung. Berikut ini adalah nilai hasil hash dari dokomen asli dan *hash* dokumen uji:

Hasil *hashing* dokumen asli:

{157159} {162279} {160034} {164606} {170988} {167983} {149558} {166511} {162644} {149402} {164784} {143657} {160147} {151223} {155531} {144362} {167906} {148717} {157262} {148768} {157818} {169521} {166480} {162313}

Jumlah hash pada dokumen asli adalah= $\sum \boldsymbol{H}_{asli} = 24$

Hasil *hashing* dokumen uji:

{157159} {162279} {160034} {164606} {170988} {167983} {149558} {166511} {162644} {149402} {164784} {143657} {160147} {151223} {155528} {144338} {167657} {145988} {171154} {169805} {169604} {152762} {157834} {155060} {168460} {169452}

Jumlah hash pada dokumen uji adalah= $\sum \boldsymbol{H}_{uji} = 26$

Hash yang sama:

{157159} {162279} {160034} {164606} {170988} {167983} {149558} {166511} {162644} {149402} {164784} {143657} {160147} {151223}

Jumlah hash yang sama dari dokumen asli dan dokumen uji adalah = $\sum \boldsymbol{H}_{\text{asli}} \cap \boldsymbol{H}_{\text{uji}} = 14$
Setelah jumlah hash diketahui, yaitu hash pada dokumen asli sebanyak 24, hash dokumen uji sebanyak 26 dan hash yang sama sebanyak 14, maka proses selanjutnya adalah menghitung *similarity*.

3. Deteksi Kesamaan

Setelah diketahui nilai hashnya, jumlah *hash* pada dokumen asli 24, jumlah *hash* pada dokumen uji 26 dan hash yang sama pada kedua dokumen yaitu 14. Proses selanjutnya adalah menghitung *similarity* yaitu tingkat kesamaan dua dokumen dengan menggunakan persamaan yaitu barapa persen tingkat kasamaannya. *Similarity* didapat dari 2 dikali hasil *hash* yang sama di bagi dengan jumlah hash kedua dokumen dikali dengan seratus persen. Berikut ini adalah proses menghitung *similarity* dua dokumen di atas :

$$\textit{similarity}\ (\text{asli, uji}) = \frac{2 * \sum \ H\ asli \cap H\ uji}{\sum \ H\ asli + \sum \ H\ uji} \text{ x } 100\ \%$$

$$= \frac{2 * 14}{24 + 26} \text{ x } 100\ \%$$
$$= \frac{28}{50} \text{ x } 100\ \%$$
$$= \frac{28}{36} \text{ x } 100\ \%$$
$$= 56\ \%$$

Hasil perhitungan *similaritry* dari kedua dokumen diatas adalah 56%, dokumen yang diuji dapat di kategorikan sebagai plagiarisme sedang.

3.3 Pengujian Akurasi

Pengujian akurasi dilakukan untuk menentukan tingkat akurasi teks berdasarkan penggunaan k-gram basis bilangan yang di gunakan di dalam sistem.

Tabel 1 data uji 4 kata & data uji 7 kata

No	K-Gram	Basis	Waktu (s)	*Similarity* (%)
1	2	3	0.743	81.5%
2	2	7	0.762	74.1%
3	2	11	0.491	74.1%
4	3	3	0.676	73.1%
5	3	7	0.733	61.5%
6	3	11	0.75	57.7%
7	4	3	0.992	56%
8	4	7	0.866	56%
9	4	11	0.896	56%

Tabel 2 data asli 840 kata & data uji 708 kata.

No	K-Gram	Basis	Waktu (s)	*Similarity*
1	2	3	0.992	106.2%
2	2	7	0.796	105.9%
3	2	11	1.262	105.2%
4	3	3	0.914	105.9%
5	3	7	1.209	96.9%
6	3	11	1.011	88.7%
7	4	3	1.056	101.3%
8	4	7	1.147	65.9%
9	4	11	0.976	56.2%

IV. KESIMPULAN DAN SARAN

1. Kesimpulan

Berdasarkan hasil penelitian yang dilakukan mengenai sistem pendeteksi plagiarisme pada proposal tugas akhir mahasiswa dengan algoritma *rabin karp* diperoleh beberapa kesimpulan sebagai berikut :

1. Pada penelitian ini melakukan pengujian akurasi berdasarkan tingkatan plagiarisme yaitu plagiarisme berat atau tidak berat .
2. Berdasarkan hasil evaluasi uji coba sistem dapat diketahui bahwa performa hasil persentase similaritas dari algoritma *Rabin karp* memiliki ketergantungan dengan nilai *k-gram* yang diberikan.
3. Sistem ini dapat menampilkan persentase kemiripan Latar belakang proposal antar mahasiswa dengan menggunakan k-gram 4 dan basis 11 berdasarkan beberapa kali uji coba yang di lakukan pada sistem sehingga Algoritma *Rabin-Karp* berhasil diimplementasikan pada sistem pendeteksi plagarisme.

2. SARAN

Beberapa saran yang perlu diperhatikan pada penelitian selanjutnya adalah sebagai berikut.

1. Aplikasi ini diharapkan dapat membantu untuk mencegah secara dini kemungkinan terjadinya kegiatan plagiarisme dalam pengerjaan tugas akhir mahasiswa.
2. Perlu adanya penelitian lebih lanjut yaitu Algoritma yang digunakan pada aplikasi ini dapat dikembangkan dengan algoritma atau metode lain

DAFTAR PUSTAKA

[1] T. Suryati, Y. Wibisono, and Y. Wihardi, “Aplikasi Deteksi Plagiarisme Dokumen Skripsi dengan Algoritma Rabin-Karp,” *JATIKOM J. Teor. dan Apl. Ilmu Komput.*, vol. 1, no. 2, pp. 91–95, 2018.

[2] & A. T. K. Aziz, Lulu A., Ana I., “Upaya Perpustakaan Dalam Mengurangi Plagiarisme Pada Karya Ilmiah Mahasiswa (Studi Kasus Di Upt Perpustakaan Unika Soegijapranata),” *J. Ilmu Perpust.*, vol. 4, no. 3, pp. 1–13, 2015, [Online]. Available: https://www.neliti.com/id/publications/137458/upaya-perpustakaan-dalam-mengurangi-plagiarisme-pada-karya-ilmiah-mahasiswa-stud.

[3] A. A. Magriyanti, “Analisis Pengembangan Algoritma Porter Stemming Dalam Bahasa Indonesia,” 2018, doi: 10.31227/osf.io/7ge4v.

[4] A. Tantoni and M. T. A. Zaen, “Implementasi Double Caesar Cipher Menggunakan Ascii,” *J. Inform. dan Rekayasa Elektron.*, vol. 1, no. 2, p. 24, 2018, doi: 10.36595/jire.v1i2.56.

[5] Nugroho:2014, “BOOK_Adi Nugroho_Rekayasa perangkat lunak_Pengantar.pdf.” .