Penerapan Simhash dan Hamming distance dalam Deteksi kemiripan Teks Berita

Mayesti Anggelina[1], Lucia Dwi Krisnawati [2], Danny Sebastian [3]
Informatika, Universitas Kristen Duta Wacana
Jl. Dr. Wahidin Sudirohusodo 5-25, Yogyakarta 55224
[1]mayesti.angelina@ti.ukdw.ac.id
[2]krisna@staff.ukdw.ac.id
[3]danny.sebastian@staff.ukdw.ac.id

Abstract*— Text reuse is defined as the reuse of existing written sources for creating a new text. The degree of reuse varies from duplicate, near-duplicate to topically similar text. Though some genres of text reuse are acceptable, their existence causes inefficiency of searching and waste of storage. To overcome this problem, a textual similarity detection system is needed. This study focuses on detecting the text similarity by applying the Simhash algorithm. It is used to create document fingerprints which function as document features through which the degree of text similarity can be compared. The similarity of a suspicious text to the source documents are measured then by Hamming Distance. Focusing on the duplicate and near-duplicate detection, the experiments conducted show that the recall of the duplicate detection reaches 80%, meaning that the system is capable of retrieving the duplicate sources of the suspicious document.

Intisari— Daur Ulang Text didefinisikan sebagai pemanfaatan sumber tulisan yang ada untuk penulisan sebuah teks baru. Persentase penggunaan ulang teks dari sumber sebelumnya sangatlah bervariasi. Jika prosentase penggunaan tersebut tinggi dan berasal dari sebuah sumber, maka teks yang baru menjadi teks duplikat atau hampir duplikat dengan teks sumbernya. Meskipun beberapa genre teks bisa diterima, keberadaan teks duplikat dan hampir dupilkat ini menyebabkan ketidak-efisienan penyimpanan dan pencarian. Untuk itu diperlukan sebuah system deteksi kemiripan teks yang akan mengidentifikasi teks mana saja yang dupilkat dan hampir duplikat. Untuk itu, penelitian ini berfokus pada deteksi kemiripan teks dengan mengaplikasikan algoritma Simhash. Algoritma ini digunakan untuk menghasilkan fingerprint dokumen yang berfungsi sebagai fitur dokumen yang digunakan sebagai dasar pembanding tingkat kemiripan teks. Kemiripan sebuah teks terhadap teks lainnya diukur dengan menggunakan jarak Hamming. Dalam ekperimen yang difokuskan pada dokumen duplikat dan hampir duplikat, tingkat Recall dokumen cukup tinggi yakni 80%. Ini berarti bahwa sistem yang dikembangkan mampu menemenukan pasangan dokumen duplikat dengan baik.

Kata Kunci*— daur ulang teks, deteksi kemiripan teks, *Simhash, hamming distance.

I. PENDAHULUAN

A. Latar Belakang

Daur Ulang Teks (*text reuse*) didefiniskan sebagai usaha pemanfaatan berbagai sumber tulisan yang ada untuk menghasilkan sebuah tulisan baru tanpa mencantumkan sumber aslinya [1]. Berdasarkan pengamatan terhadap berbagai genre penulisan, maka Daur Ulang Teks (DaUT) bisa dikategorikan dalam 2 kelompok yaitu DaUT yang legal dan DaUT illegal [2]. DaUT illegal bisa dijumpai di genre artikel ilmiah, dan karya sastra yang disebut sebagai karya plagiasi [3]. DaUT legal adalah penggunaan ulang teks tanpa sitasi namun bisa diterima oleh masyarakat, sebagai contohnya adalah artikel berita di media cetak maupun elektronik. Kemiripan artikel berita ini terjadi karena media sering menerima rilis dari sumber berita dan hanya menggubahnya sedikit saja. Meskipun demikian, kemiripan teks di artikel berita dan naskah hukum tidak akan diklaim sebagai hasil karya plagiasi.

Spektrum kemiripan teks baik di DaUT legal maupun illegal cukuplah bervariasi yakni kemiripan di level topik, leksikal, sintaksis, semantik atau struktur penulisan. Selain itu, prosentase kemiripan antara dua atau lebih teks menentukan juga apakah sebuah teks termasuk dalam kategori DaUT atau tidak. Baik spektrum maupun prosentase kemiripan, keduanya menentukan area penelitian. Sebagai contohnya, kemiripan dua teks di tingkat leksikal dengan prosentase yang tinggi sekitar 85-100% akan menjadi bidang penelitan Deteksi Duplikat dan Hampir Duplikat (*duplicate and near-duplicate detection*), sedangkan kemiripan di level leksikal sebatas topik dengan prosentase kemiripan leksikal yang rendah menjadi focus penelitian Temu Kembali Dokumen (Information Retrieval) [1].

Beberapa manfaat dari deteksi kemiripan teks di tingkat leksikal dan sintaksis dengan prosentase yang tinggi (bidang *near-duplicate detection*) adalah untuk efisiensi pengindeksan situs web [4], [5] pengarsipan artikel di perpustakaan elektronik [6], atau efisiensi penyusunan pasal-pasal peraturan undang-undang [7], inferensi model App berbasi web [8] , dan juga untuk pelacakan konten historis dalam naskah-naskah bersejarah [9].

Algoritma maupun model yang digunakan untuk mendeteksi teks duplikat dan hampir duplikat bervariasi juga. Beberapa diantaranya menggunakan TLSH [8], singles dan Simhash [6], [8], Smith Waterman Local Alignment [7], Word Local Score [1], atau Algoritma klasifikasi yang dikembangkan sendiri yang dinamakan Reuse Classification Algorithm [9].

Penelitian ini berfokus pada penerapan Simhash sebagai perwujudan dari algoritma *Local Sensitive Hashing*. Simhash digunakan untuk menghasilkan fingerprint dokumen sebagai upaya untuk merepresentasikan konten dokumen dalam bentuk bilangan biner. Sedangkan untuk penghitungan jarak antar dokumen akan digunakan *Hamming Distance*. Aplikasi

yang dibangun diharapkan mampu menampilkan daftar nama-nama dokumen, id paragraf dokumen yang saling mirip dan mengelompokkan tingkat kemiripan antar paragraf dalam dokumen yaitu *duplicate* atau *Near-duplicate*.

II. LANDASAN TEORI

A. *Hashing dan Fingerprint Dokumen*

Computersciencewiki mendefinisikan *hashing* sebagai proses transformasi dari sebuah string karakter menjadi nilai integer yang lebih pendek[1]. Hashing sering digunakan untuk membuat indeks dan menemukan kembali (*retrieve*) beberapa data di basis data karena lebih cepat menemukan data dalam bentuk nilai hash yang lebih pendek daripada data dalam bentuk aslinya. Hashing merupakan langkah awal dalam pembentukan fingerprint dokumen. Meminjam konsep dari fenomena sidik jari manusia yang selalu unik, maka fingerprint dokumen adalah kumpulan integer yang merepresentasikan isi-isi penting dari sebuah dokumen. Fingerprint dokumen diharapkan unik dan selalu berbeda sama seperti sidik jari manusia. Dengna demikian fingerprint dokumen ini bukan sekedar berfungsi sebagai fitur dokumen namun sebagai representasi dokumen.

Menurut Hoad dan Zobel [10], ada 4 area yang perlu dipertimbangkan dalam mengenerasi fingerprint dokumen. Ke-empat area tersebut adalah:

1) **Fungsi Hash** yang digunakan untuk menghasilkan fingerprint dari substring dokumen. Fungsi yang dipilih ini perlu konsisten dalam menghasilkan fingerprint yang sama dari substring yang sama.
2) **Ukuran substring** yang diekstraksi dari dokumen. Ukuran ini merujuk pada granularitas atau panjang substring seperti pentagram karakter, n-gram kata atau bahkan kalimat.
3) **Jumlah fingerprint** yang digunakan sebagai fingerprint dokumen. Jumlah fingerprint ini sering disebut sebagai resolusi fingerprint dokumen.
4) **Strategi seleksi** merujuk pada pilihan algoritma yang digunakan untuk menyeleksi substring dokumen

Dalam kaitannya dengan ke-empat area ini, Simhash menempati posisi sebagai fungsi yang digunakan untuk generasi fingerprint.

B. *Pra-pemrosesan*

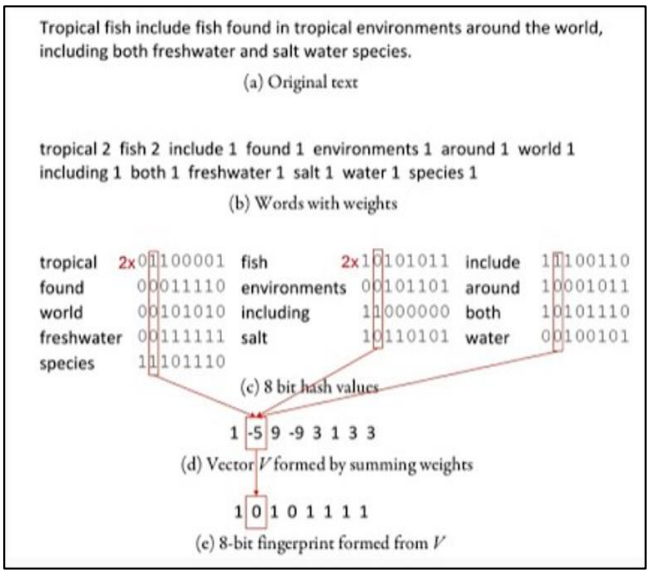
Pra-pemrosesan adalah proses awal dalam rangkaian pemrosesan teks. Prapemrosesan teks diterapkan untuk menormalisasi atau membersihkan teks agar siap diproses. Secara umum tahapan pra-pemrosesan, teks adalah sebagai berikut:

Contoh kalimat: Universitas K@risten duTA waCana ini

1) **Lower case** atau sering disebut case folding, adalah usaha untuk mengubah semua karakter ke dalam huruf kecil.
   a. Lower case: universitas k@risten duta wacana ini!!
2) **Tokenisasi**, adalah proses memecah konten dokumen dari tipe data string kedalam array berdasarkan spasi kosong. Setiap kumpulan karakter yang dipisahkan dengan spasi kosong disebut sebagai token. Singkatnya, tokenisasi adalah memecah string dokumen menjadi token. Tokenisasi: “universitas”, “k@risten”, “duta”, “wacana”, “ini!!”
3) **Penghapusan Tanda Baca** adalah proses menghapus tanda baca, yaitu karakter-karakter unik seperti tanda seru, tanda tanya, koma dsb.
   a. Penghapusan Tanda Baca: universitas kristen duta wacana ini
4) **Penghapusan Stopword** adalah penyaringan token dari list stopword. Token yang terdapat dalam daftar stopword akan dihapus sehingga tidak dikenakan proses di tahapan utama. terdapat dalam list akan diabaikan.
   a. Penghapusan Stopword: universitas kristen duta wacana

III. ALGORITMA SIMHASH

Simhash merupakan teknik untuk mengestimasi secara cepat elemen yang sama dari dua himpunan. Simhash dikenal juga dengan nama *Locality Sensitive Hashing* atau *Rounding Algorithm* yang ditemukan oleh Moses S Charikar [11]. Salah satu keunggulan Simhash adalah kemampuannya mengidietntifikasi string yang sama dari kumpulan data yang besar dan sangat efisien. Berkat keunggulan ini, Google menggunakan Simhash untuk mendeteksi teks duplikat sebagai bagian dari proses crawling-nya.



Gambar 1. Cntoh penerapan algoritma *Simhash* berdasarkan *[12]*.

Simhash membentuk *fingerprint* dokumen dari *shingles* yang merupakan rangkaian kata yang saling bertumpang tindih dengan panjang tertentu. Dengan kata lain, shingles adalah n-gram yang dibentuk dari kata atau token. Normalisasi teks yang sering diterapkan adalah penghilangan tanda baca, pengubahan huruf kecil, dan penghilangan stopwrod. Setelah itu penentuan panjang jendela untuk membentuk shingles (granularitas) ditentukan. Untuk

[1] https://computersciencewiki.org/index.php/Hashing

JUTEI Edisi Volume.6 No.2 Oktober 2022
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2022.62.216

tahapan pembentukan *fingerprint* menggunakan algoritma *Simhash* [12] bisa dilihat di tahapan berikut ini:

1) Proses dokumen menjadi *shingles* beserta bobotnya. Dalam contoh ini panjang jendela shingles adalah kalimat, dan frekuensi kemunculan kata (tf) digunakan sebagai bobot dalam tiap *shingle*. Ilustrasinya bisa dilihat di Gambar 1.
2) Generasikan nilai hash dengan b bit untuk tiap tipe kata (*type*).
   a. Nilai hash harus unik untuk tiap tipe kata (*type*).
   b. Nilai b yang sering digunakan adalah 32 dan 64 bit agar persyaratan a terpenuhi.
3) Dalam vektor kata v dengan dimensi b, update komponen vektor di tiap bit dengan cara:
   a. Tambahkan nilai bobot jika nilai hashnya adalah 1
   b. Kurangi sebesar bobot jika nilai vektor hash tersebut adalah 0.
4) Setelah semua tipe kata telah diproses, generasikan fingerprint b-bit dengan nilai bit ke-i 1 jika komponen ke-i dari vector v bernilai positif atau 0, jika tidak konversikan ke 0.

IV. HAMMING DISTANCE

No	Numbers	16-bit simhash	Hamm-dist
1	37586	1001001011010010	
2	50086	1100001110100110	7
3	2648	0000101001011000	11
4	934	0000001110100110	9
5	40957	1001111111111101	9
6	2650	0000101001011010	9
7	644475	1111101111011011	7
8	40955	1001111111111011	4

Gambar 2. Menghitung jarak hamming tanpa pengurutan

No	Numbers	16-bit simhash	Hamm-dist
4	934	0000001110100110	
3	2648	0000101001011000	9
6	2650	0000101001011010	1
1	37586	1001001011010010	5
8	40955	1001111111111011	6
5	40957	1001111111111101	2
2	50086	1100001110100110	9
7	644475	1111101111011011	9

Gambar 3. Menghitung jarak hamming dengan pengurutan

Hamming distance (jarak Hamming) merupakan metrik yang menghitung jumlah simbol atau posisi yang berbeda dari dua string dengan panjang yang sama [13]. Jarak Hamming sangatlah sesuai diterapkan dalam pengukuran kemiripan dokumen yang direpresentasikan dengan fingerprint hasil dari teknik Simhash dalam bentuk bilangan biner. Perhitungan jarak Hamming didasarkan pada operasi bitwise [12]. Gambar 2 memberikan gambaran dalam perhitungan jarak Hamming antara 2 fingerprint dokumen dengan asumsi penggunan 16 bit dalam pembentukan fingerprintnya. Dalam penghitungan jarak Hamming, ada beberapa teknik misalnya dengan atau tanpa pengurutan nilai hash sebelum diubah ke bilangan biner 16 bit. Gambar 2 menunjukkan Contoh penghitungan jarak Hamming tanpa adanya pengurutan nilai hash terrendah ke tinggi sedangkan Gambar 3 menunjukkan dengan pengurutan.

Jarak Hamming terendah yang didemonstrasikan di Gambar 2 adalah 4. Tentu saja penghitungan jarak Hamming seperti ini kurang efektif. Untuk dilakukan pengurutan atau sortir berdasarkan nilai hashnya kemudian barulah dilakukan penghitungan jarak Hamming. Dengan pengurutan dari Contoh yang sama, maka jarak terendah adalah 1 dari fingerprint nomor urut 3 dan 6.

V. EVALUASI

Evaluasi dilakukan untuk mengukur kinerja sistem yang dibangun menggunakan algoritma Simhash dan dengan penghitungan jarak Hamming. Metrik yang digunakan untuk pengukuran adalah Presisi, Recall, dan F-1score. Untuk penghitungan ketiga metric tersebut dibutuhkan confusion matrix seperti yang ditunjukkan di Tabel 1. Dalam Confusion Matrix ini, sebagai contohnya, kita membangun sistem prediksi terjangkit Covid-19. Maka kita memiliki 4 kemungkinan yakni saat kondisi riil terjangkit covid-dan hasil tes juga positif, kondisi ini disebut dengan True Positive (TP). Jika kondisi riil seseorang positive Covid-19 namun hasil tes dan predisksinya negative, maka kesalahan ini disebut sebagai False Negative (FN). Kondisi False Positive (FP) merujuk pada saat sistem memprediksi positive covid-19 namun kondisi riilnya adalah negative. True Negative (TN) adalah bila kondisi riil seseorang adalah negative dan hasil prediksi tes juga negative.

Tabel 1 *Confusion matrix*

	Riil positive	**Riil negative**
Prediksi positive	TP	FP
Prediksi negative	FN	TN

Presisi adalah metrik yang menghitung hasil prediksi benar dibagi dengan data dari fraksi riil positif. Presisi dihitung menggunakan rumus berikut ini:

$$Prec = \frac{TP}{TP+FN} \quad [1]$$

Sedangkan Recall dikomputasi dengan menghitung hasil prediksi benar dibagi dengan data dari fraksi prediksi negatif seperti yang ditunjukkan di persamaan 2 berikut ini:

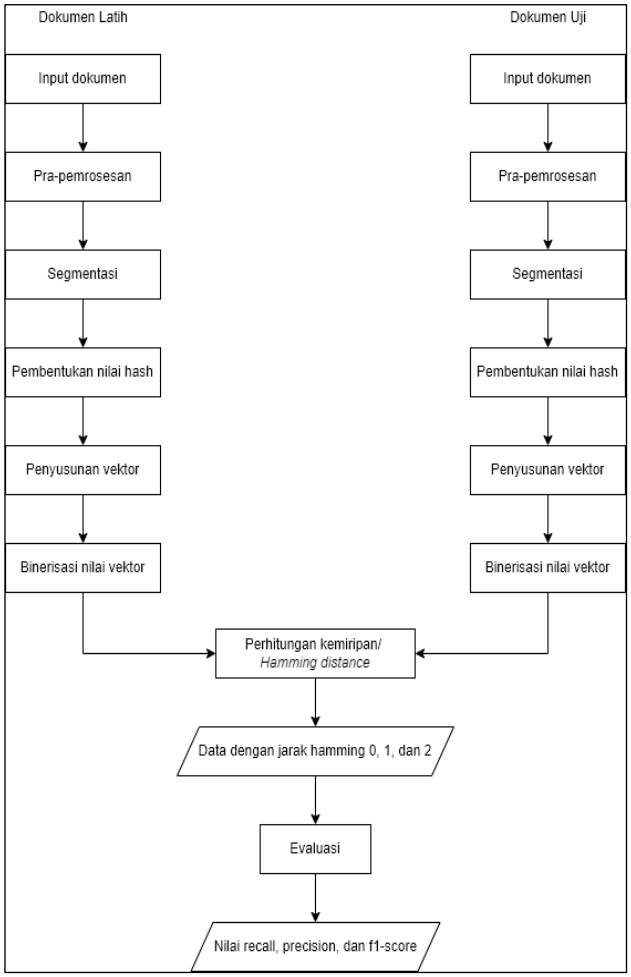
$$Rec = \frac{TP}{TP+FP} \quad [2]$$

F-1 score merupakan nilai rata-rata harmonis antara Presisi dan Recall saat nilai β didefiniskan sama dengan 1. F-1 score dikomputasi dengan menggunakan persamaan 3 sebagai berikut:

$$F_1\ score = \frac{2*Prec*Rec}{Prec+Rec} \quad [3]$$

VI. METODOLOGI PENELITIAN

Tahap proses pembangunan sistem ditunjukkan di Gambar 4. Tahap pemrosesan dokumen latih di korpus dan dokumen uji sama yakni keduanya dilakukan praproses teks, kemudian segmentasi untuk menentukan lebah jendela dari shingles. Setelah itu diterapkan konversi string ke dalam nilai hash, penyusunan vector v dan binerisasi vector v dari tiap fingerprint dokuumen. Setelah fingerprint terbentuk, maka penghitungan jarak Hamming antara fingerprint dokumen yang ada di korpus dengan fingerprint dokumen uji dilakukan. Penyaringan dilakukan dengan memilih fingerprint yang memiliki jarak 0-2 dan setelah itu dilakukan evaluasi. Untuk detail tiap tahapan akan diuraikan di subbahasan berikut ini.



Gambar 4. Tahapan proses pembangunan sistem

A. *Pra Pemrosesan*

Pra-pemrosesan yang pertama dilakukan sistem yaitu lower case atau case folding menggunakan fungsi lower(). Langkah pra-pemrosesan selanjutnya adalah menghapus angka dan kata yang kurang relevan dalam penelitian ini karena penomoran list sangat mungkin berbeda dan apabila penomoran list tidak dihapus maka akan mempengaruhi kemiripan teks, misalnya nomor list pada Gambar 5. Untuk proses ini program menggunakan modul Regular Expression (regex) dengan ekpresi “\n\d+[.]”, “\n\w+[.]”.

```
press release: ukdw yogyakarta rayakan dies natalis di tengah pandemi

total mahasiswa baru yang diterima pada tahun 2020 ini adalah sebanyak 1.007 orang yang terdi
ri dari:

1. 839 mahasiswa program sarjana, 96 mahasiswa program profesi dokter
2. 72 mahasiswa program pascasarjana.

hal ini menunjukkan bahwa meskipun di tengah situasi pandemi covid-19 ini, menunjukkan masih
banyak orang yang ingin melanjutkan kuliah di ukdw. “meski mengalami penurunan jumlah penerim
aan mahasiswa sekitar 8-10 persen dari tahun lalu, kita harus tetap bersyukur atas penyertaan
 tuhan di tengah pandemi ini,” ungkap rektor.
```

Gambar 5. Contoh teks dengan penomoran list

B. *Segmentasi teks*

Pada penelitian ini dilakukan segmentasi kalimat. Namun untuk mendapatkan segmentasi kalimat, penulis terlebih dahulu melakukan segmentasi paragraf, dengan memecah teks berdasarkan baris baru (“\n”). Segmentasi paragraf menghasilkan adanya list kosong karena segmentasi paragraf mengartikan bahwa newline adalah sebuah paragraf. Untuk mengatasi hal ini program dibangun untuk melakukan pengecekan isi list paragraf terlebih dahulu. Jika paragraf kosong (“”) maka list akan diabaikan dan jika paragraf tidak kosong maka akan disegmentasi per kalimat seperti yang ditunjukkan Gambar 6.

```
press release ukdw yogyakarta rayakan dies natalis di tengah pandemi
total mahasiswa baru yang diterima pada tahun 2020 ini adalah sebanyak 1007 orang yang terdir
i dari
 839 mahasiswa program sarjana 96 mahasiswa program profesi dokter
 72 mahasiswa program pascasarjana
hal ini menunjukkan bahwa meskipun di tengah situasi pandemi covid19 ini menunjukkan masih ba
nyak orang yang ingin melanjutkan kuliah di ukdw “meski mengalami penurunan jumlah penerimaan
 mahasiswa sekitar 810 persen dari tahun lalu kita harus tetap bersyukur atas penyertaan tuha
n di tengah pandemi ini” ungkap rektor
```

Gambar 6. Contoh penghapusan tanda baca pada segmentasi kalimat

Program menghasilkan kalimat dengan beberapa tanda baca yang kurang relevan seperti tanda koma (,), titik (.), titik dua (:), dan petik dua (“”), maka dari itu program akan menghapus tanda-tanda baca tersebut. Penghapusan tanda baca dilakukan setelah segmentasi kalimat untuk menghindari kesalahan karena segmentasi kalimat menggunakan tanda titik untuk melakukannya.

Saat melakukan segmentasi kalimat berdasarkan tanda titik akan menyebabkan pemotongan kalimat yang kurang tepat, misalnya terdapat nama dengan gelar akamedik seperti “Dr.Phil”. Sistem akan mendeteksi Dr dan Phil merupakan dua kalimat yang berbeda padahal itu adalah sebuah nama, untuk mengurangi kesalahan demikian sistem dibangun untuk menggabungkan token kalimat yang memiliki panjang di bawah 6 dengan token kalimat setelahnya. Pemilihan batas panjang kalimat tersebut didasari penelitian terdahulu yang dilakukan penulis dan disimpulkan bahwa panjang gelar akademik yang biasanya dituliskan pada teks berukuran tidak kurang dari 6.

C. *Pembentukan nilai hash*

Teks akhir hasil proses segmentasi kalimat akan ditokenisasi menggunakan modul nltk. Tokenisasi pada penelitian ini memecah kalimat menjadi potongan-potongan kata. Program menggunakan spasi sebagai acuan dalam membentuk token dan menyimpan seluruh token dalam list of string. Contoh tokenisasi sebuah kalimat ditunjukkan Gambar7.

```
['press', 'release', 'ukdw', 'yogyakarta',
'rayakan', 'dies', 'natalis', 'pandemi'],
```

Gambar 7. Contoh token sebuah kalimat

JUTEI Edisi Volume.6 No.2 Oktober 2022
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2022.62.216

Selanjutnya, sistem yang dibangun membaca setiap huruf dari setiap token dan mengubahnya menjadi nilai ASCII menggunakan fungsi python yaitu ord() dan menyimpan hasilnya dalam list of list. Contoh nilai ASCII dari Gambar 7 ditunjukkan pada Gambar 8.

```
[[112, 114, 101, 115, 115], [114, 101, 108, 101, 97, 115, 101], [117,
 107, 100, 119], [121, 111, 103, 121, 97, 107, 97, 114, 116, 97], [11
4, 97, 121, 97, 107, 97, 110], [100, 105, 101, 115], [110, 97, 116, 9
7, 108, 105, 115], [112, 97, 110, 100, 101, 109, 105]]
```

Gambar 8. Contoh nilai ASCII dari token sebuah kalimat

Sebagai Contoh, dari Gambar 7, token ‘press’ terdiri dari huruf p,r,e,s,s. Nilai ASCII hurus p=112, r=114, e=101, s=115 sehingga list token ‘press’ berisi [112, 114, 101, 115, 115]. Konversi ke nilai ASCII ini dilakukan secara iteratid sampai seluruh token terkonversikan ke kode ASCII-nya. Selanjutnya program menjumlahkan semua nilai ASCII tiap token dalam kalimat dan menyimpannya dalam sebuah list seperti yang ditunjukkan pada Gambar 9.

```
[557, 737, 443, 1084, 743, 421, 748, 734]
```

Gambar 9. Contoh nilai hash token kalimat

Langkah selanjutnya adalah mengkonversi nilai dalam list menjadi nilali hash dengan fungsi has sebagai berikut:

$$h(x) = (Ax + B) \bmod p$$

dimana:
A = 7210, X = nilai ascii token, p = 7211 dan B = nomor index token dalam korpus

Nilai p ditentukan berdasarkan jumlah token yang unik dan harus bilangan prima. Penelitian ini menggunakan 40 dokumen dan memiliki 7210 token unik, dan bilangan prima terdekat dengan 7210 adalah 7211. Hasil dan fungsi tersebut merupakan nilai hash setiap token.

D. Penyusunan vektor

Penyusunan vektor akan dilakukan dengan dua tahap yaitu yang pertama konversi nilai hash setiap token ke biner 16 bit. Dalam algoritma *Simhash* tidak ada peraturan tetap yang mengharuskan jumlah biner, penulis memilih 16 bit untuk menghindari terjadinya overload bit karena nilai hash yang akan dikonversi lumayan besar. Kemudian, pembobotan bit biner. Pembobotan dilakukan berdasarkan posisi bit dengan menambahkan bobot 1 jika bit biner ke-i bernilai 1, dan mengurangi bobot 1 jika bit biner ke-i bernilai 0. Contoh pembobotan bit biner sebuah kalimat sebagai ditunjukkan oleh Gambar 10 berikut ini:

Gambar 10. Proses pembentukan vektor v

Hasil pembobotan disimpan dalam array satu dimensi yaitu vektor.

E. Binerisasi nilai vektor

Program melakukan binerisasi nilai vektor dengan cara melakukan pengecekan setiap nilai vektor dengan ketentuan:

1) Jika nilai vektor < 1, maka nilai vektor digantikan dengan angka 0
2) Jika nilai vektor >=1, maka nilai vektor digantikan dengan angka 1

Maka dari itu hasil biner dari vektor di atas adalah 0000001010101101. Langkah ini dilakukan sampai seluruh vektor kalimat dokumen input dibinerisasi. Hasil binerisasi merupakan fingerprint yang digunakan untuk proses deteksi. Seluruh fingerprint disimpan secara otomatis ke dalam file csv bernama Fingerprint.csv dengan mode baca “a” yaitu mode menulis data baru pada row terakhir tanpa menghapus data lain. Contoh isi file Fingerprint.csv ditunjukkan pada Gambar 11.

	A	B	C	D
1	ID Dokumen	ID Kalimat	Vektor	Fingerprint
2	Diesnatalis_ebahana.txt	1	-10,-10,-10,-10,-10,-8,2,-6,4,2,6,-2,2,6,-2,4	1011101101
3	Diesnatalis_ebahana.txt	2	-17,-17,-17,-17,-17,-17,-1,1,1,-5,1,-3,-3,3,5,-1	110100110
4	Diesnatalis_ebahana.txt	3	-9,-9,-9,-9,-9,-9,5,-5,7,1,7,1,5,-1,-3,-5	1011111000
5	Diesnatalis_ebahana.txt	4	-4,-4,-4,-4,-4,-2,0,-2,2,2,4,0,4,-2,-2,-2	11101000
6	Diesnatalis_ebahana.txt	5	-22,-22,-22,-22,-22,-16,0,-8,-6,-2,-2,0,-4,6,6,2	111
7	Diesnatalis_ebahana.txt	6	-26,-26,-26,-26,-26,-22,6,-8,4,-10,6,-6,0,-2,-4,0	1010100000
8	2karyaukdw_seremonia.txt	1	-11,-11,-11,-11,-11,-9,1,-1,-1,-1,-1,5,1,1,-3,-3	1000011100
9	2karyaukdw_seremonia.txt	2	-32,-32,-32,-32,-32,-32,2,2,-8,2,-4,4,-8,10,10,-6,-2	1010101100
10	2karyaukdw_seremonia.txt	3	-24,-24,-24,-24,-24,-18,-2,-4,4,-4,-8,6,-14,4,-4,0	10010100
11	2karyaukdw_seremonia.txt	4	-9,-9,-9,-9,-9,-9,-9,1,3,1,-3,-1,1,-3,-5,3,-3	1110010010
12	2karyaukdw_seremonia.txt	5	-44,-44,-44,-44,-44,-30,6,-10,-12,-2,24,-4,-2,0,8,16	1000100011
13	2karyaukdw_seremonia.txt	6	-28,-28,-28,-28,-28,-20,0,8,8,4,-8,0,2,0,2,-2	111001010
14	2karyaukdw_seremonia.txt	7	-20,-20,-20,-20,-20,-18,8,-6,6,-2,0,-2,2,4,2,-2	1010001110
15	2karyaukdw_seremonia.txt	8	-34,-34,-34,-34,-34,-22,6,-10,4,0,8,-6,10,2,8,0	1010101110
16	2karyaukdw_seremonia.txt	9	-55,-55,-55,-55,-55,-43,17,-25,-15,-13,-1,5,9,-1,-7,-9	1000011000
17	2karyaukdw_seremonia.txt	10	-8,-8,-8,-8,-8,-8,0,-2,2,0,4,2,0,0,6,0	10110010
18	2karyaukdw_seremonia.txt	11	-5,-5,-5,-5,-5,-5,-5,-1,-3,3,-1,3,-3,-3,1,1,1	10100111

Gambar 41. Contoh isi file Fingerprint.csv

F. Penghitungan Kemiripan Dokuumen

Program menghitung kemiripan antar fingerprint menggunakan metode *Hamming distance* yaitu jarak hamming antara dua string dengan panjang yang sama, adalah banyaknya posisi di kedua string yang berbeda simbol. Pada penelitian ini seluruh fingerprint memiliki panjang yang sama yaitu 16 bit. Jarak hamming dilakukan terhadap fingerprint ke-i dan fingerprint ke-i+1 sehingga satu jarak Hamming terdiri dari dua data. Misalnya terdapat sebuah daftar fingerprint sebagai berikut:

Fingerprint ke –
1→ 0000001000011100
2→ 0000001010101100
3 → 0000000010010100
4 → 0000001110010010
5 → 0000001000100011

Urutan merupakan hal yang penting untuk mendapatkan jarak Hamming, misalnya *fingerprint* 1 dan 2 akan menghasilkan satu nilai jarak Hamming dengan cara menghitung perbedaan simbol dari kedua *fingerprint* sesuai nomor index yang sama.

Fingerprint ke –
1→ 0000001000011100
2→ 0000001010101100

Fingerprint 1 index ke-1 hingga ke-8 memiliki simbol yang sama dengan fingerprint 2 index ke-1 hingga ke-8. Namun pada index ke-9, 11, dan 12, fingerprint 1 memiliki

simbol yang berbeda dengan fingerprint 2 index ke-9, 11, dan 12. Program menemukan adanya tiga kali perbedaan simbol dan jumlah perbedaan itu adalah jarak hamming antara fingerprint 1 dengan fingerprint 2. Selanjutnya program menghitung jarak hamming fingerprint 2 dengan fingerprint 3, jarak hamming fingerprint 3 dengan fingerprint 4 dan begitu seterusnya hingga seluruh baris data selesai dihitung.

Perhitungan jarak hamming dilakukan beberapa kali sampai jarak hamming yang dihasilkan tidak mengalami perubahan yang signifikan. Pada penelitian ini program akan melakukan 4 iterasi perhitungan jarak hamming agar fingerprint yang sama atau mirip bisa ditemukan dan memiliki jarak terendah. Iterasinya adalah sebagai berikut:

1) **Menghitung jarak Hamming fingerprint tanpa sortir** atau pengurutan dari data yang tersimpan di berkas Fingerprint.csv. File Fingerprint.csv merupakan data yang disimpan berdasarkan urutan input dan urutan id kalimat. Dokumen yang diinputkan lebih dahulu akan berada dibaris teratas dan fingerprint dokumen tersebut akan tersusun berdasarkan id kalimat secara *ascending*. Dokumen yang lebih dahulu diinputkan adalah dokumen "Diesnatalis_ebahana.txt" dan fingerprint dokumen dimulai dari id kalimat 1. Hasil dari proses penghitungan jarak Hamming ini disimpan program dalam file Jarak_1.csv seperti pada Gambar 12.

	ID Dokumen	ID Kalimat	Hash	Fingerprint	Jarak
2	Diesnatalis_ebahana.txt	1	749	1011101101	
3	Diesnatalis_ebahana.txt	2	422	110100110	6
4	Diesnatalis_ebahana.txt	3	760	1011111000	7
5	Diesnatalis_ebahana.txt	4	232	11101000	2
6	Diesnatalis_ebahana.txt	5	7	111	7
7	Diesnatalis_ebahana.txt	6	672	1010100000	6
8	2karyaukdw_seremonia.txt	1	540	1000011100	5
9	2karyaukdw_seremonia.txt	2	684	1010101100	3
10	2karyaukdw_seremonia.txt	3	148	10010100	4
11	2karyaukdw_seremonia.txt	4	914	1110010010	4
12	2karyaukdw_seremonia.txt	5	547	1000100011	5
13	2karyaukdw_seremonia.txt	6	458	111001010	7
14	2karyaukdw_seremonia.txt	7	654	1010001110	4
15	2karyaukdw_seremonia.txt	8	686	1010101110	1
16	2karyaukdw_seremonia.txt	9	536	1000011000	5
17	2karyaukdw_seremonia.txt	10	178	10110010	5
18	2karyaukdw_seremonia.txt	11	167	10100111	3
19	2karyaukdw_seremonia.txt	12	189	10111101	3

Gambar 52. Contoh isi file Jarak_1.csv

2) **Mengurutkan data** berdasarkan nilai hash dari nilai terkecil ke nilai hash tertinggi. Pada proses ini program membaca file Jarak_1.csv dan mengurutkan data secara ascending berdasarkan nilai hash fingerprint menggunakan fungsi sorted milik python dengan k = nilai hash pada file Jarak_1.csv. Setelah data diurutkan, program melakukan **perhitungan jarak hamming** dan hasil disimpan dalam file Jarak_2.csv.
3) **Melakukan pergeseran nilai bit ke kanan** (*shift right*) dengan n=2. Pada proses ini program membaca file Jarak_2.csv dan melakukan shift right pada fingerprint. Setelah bilangan biner fingerprint di shift right, program melakukan **perhitungan jarak hamming** kembali. Operasi geser kanan akan mengubah nilai desimal biner semula, namun tidak menambah atau mengurangi jarak Hamming-nya. Untuk itu, baik jarak hamming maupun nilai hash dari fingerprint yang digeser tersebut disimpan ke dalam file Jarak_3.csv.
4) Mengurutkan data berdasarkan nilai hash fingerprint secara ascending dan menghitung ulang jarak Hamming. Langkah ini adalah iterasi dari langkah kedua. Pada proses ini program membaca file Jarak_3.csv. Setelah data diurutkan, program melakukan perhitungan jarak hamming dan hasil disimpan dalam file Jarak_4.csv seperti yang ditunjukkan Gambar 13.

Setiap sebelum menyimpan hasil sistem pada proses ini kedalam file csv, program terlebih dahulu mengosongkan file untuk menghindari bias. Program menyimpan setiap hasil kedalam sebuah file untuk mempermudah penulis melihat perubahan jarak hamming setiap proses. Penulis melakukan percobaan terlebih dahulu untuk mengetahui jumlah iterasi langkah yang harus dilakukan agar jarak hamming tidak mengalami perubahan yang signifikan, dan disimpulkan bahwa keempat langkah di atas cukup dilakukan sekali iterasi saja.

	ID Dokumen	ID Kalimat	Hash	Fingerprint	Jarak
2	Diesnatalis_ebahana.txt	5	28	11100	
3	2karyaukdw_seremonia.txt	3	592	1001010000	4
4	2karyaukdw_seremonia.txt	11	668	1010011100	4
5	2karyaukdw_seremonia.txt	10	712	1011001000	3
6	2karyaukdw_seremonia.txt	12	756	1011110100	4
7	Diesnatalis_ebahana.txt	4	928	1110100000	4
8	Diesnatalis_ebahana.txt	2	1688	11010011000	5
9	2karyaukdw_seremonia.txt	6	1832	11100101000	4
10	2karyaukdw_seremonia.txt	15	2072	1E+11	6
11	2karyaukdw_seremonia.txt	16	2104	1E+11	1
12	2karyaukdw_seremonia.txt	9	2144	1.00001E+11	3
13	2karyaukdw_seremonia.txt	1	2160	1.00001E+11	1
14	2karyaukdw_seremonia.txt	5	2188	1.0001E+11	6
15	2karyaukdw_seremonia.txt	13	2224	1.0001E+11	4
16	2karyaukdw_seremonia.txt	7	2616	1.01E+11	3
17	2karyaukdw_seremonia.txt	14	2664	1.01001E+11	2
18	Diesnatalis_ebahana.txt	6	2688	1.0101E+11	4
19	2karyaukdw_seremonia.txt	2	2736	1.0101E+11	2

Gambar 63. Contoh isi file Jarak_4.csv

G. Memilih data dengan jarak hamming terkecil

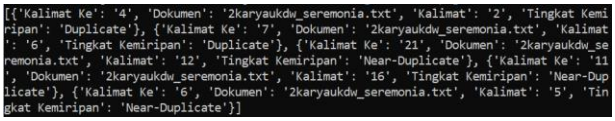
Sistem melakukan beberapa proses penyaringan untuk menyelesaikan proses pemilihan data ini yaitu:

1) Sistem membagi data menjadi dua yaitu data dokumen yang sedang diuji dan data dokumen tidak diuji (dokumen latih). Tahap ini menggunakan nama dokumen yang sedang diuji sebagai parameter. Kemudian sistem menyaring data dokumen uji. Apabila jarak dokumen bernilai 0 dan jarak dokumen bernilai <=5.
2) Sistem menyaring data dokumen latih dengan mencocokkan fingerprint dalam list dokumen latih dengan fingerprint dalam list data jarak 0. Dokumen latih yang tersaring oleh sistem disimpan dalam sebuah list.
3) Menggunakan list data berjarak <=5 pada langkah 1 untuk mendapatkan kemiripan tingkat duplikat dalam list data latih. Sistem menyaring data latih dengan mencocokkan *fingerprint* dalam list dokumen latih dengan *fingerprint* dalam list data berjarak <= 5. Dokumen latih yang tersaring oleh sistem disimpan dalam list bernama newlisting_dup.
4) Setelah langkah 2 dan 3 ada data dengan nama dokumen yang sama pada dokumen uji dan dokumen

JUTEI Edisi Volume.6 No.2 Oktober 2022
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2022.62.216

latihnya. Hal ini bisa terjadi karena pada file Jarak_4.csv posisi data diurutkan berdasarkan nilai hash fingerprint. Sangat memungkinkan jika satu dokumen memiliki nilai hash fingerprint yang hampir sama. Maka dari itu program melakukan penyaringan lagi untuk menangani kasus ini. Program hanya akan mengambil data dan menyimpannya pada list baru apabila nama dokumen pada index data sebelumnya dengan data jarak <=5 tidak sama.

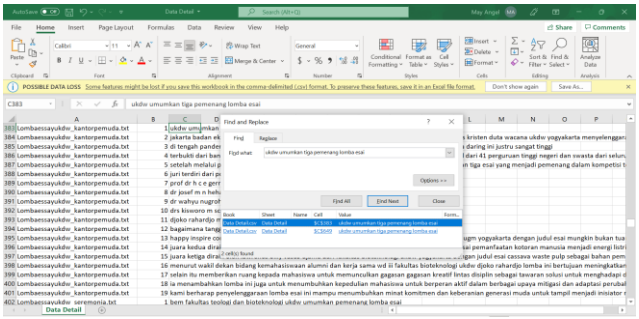
5) Langkah terakhir, program menggabungkan list terakhir pada langkah 2 sebagai data duplikat dan list terakhir pada langkah 4 sebagai data *Near-duplicate*. Hasil penggabung ini merupakan hasil akhir yang ditampilkan program kepada pengguna. Contoh hasil penggabungan ditunjukkan pada Gambar 14.



Gambar 74. Contoh hasil akhir program

H. Skenario pengujian

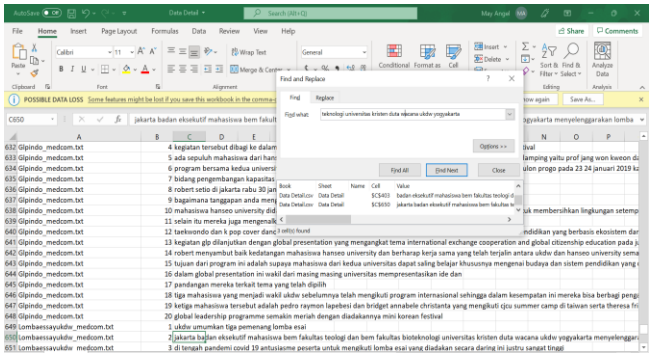
Nilai evaluasi yaitu precision, recall dan F1 score dihitung untuk tingkat kemiripan dua jenis yaitu Duplicate dan *Near-Duplicate*. Penulis mencari kalimat setiap dokumen pada file Data Detail.csv yang berisi data latih berupa nama dokumen, id kalimat dokumen berserta kalimatnya. Contoh ditemukkannya kalimat duplikat pada data latih ditunjukkan pada Gambar 15.



Gambar 85. Contoh penemuan kalimat duplicate

Gambar diatas yaitu pencarian kalimat 1 milik dokumen Lombaessayukdw_medcom.txt yaitu “ukdw umumkan tiga pemenang lomba esai” ditemukan kalimat yang sama persis pada kalimat ke-1 milik dokumen Lombaessayukdw_kantor pemuda.txt. Sehingga nama dokumen Lombaessayukdw _kantorpemuda.txt dan id kalimat 1 dimasukkan dalam daftar duplikat dokumen Lombaessayukdw_medcom.txt.

Pada Gambar 16 dicari kalimat ke-2 dari dokumen Lombaessayukdw_medcom.txt yaitu “jakarta badan eksekutif mahasiswa bem fakultas teologi dan bem fakultas bioteknologi universitas kristen duta wacana ukdw yogyakarta menyelenggarakan lomba penulisan esai 2020 tema yang diangkat adalah perubahan iklim dan pembangunan berkelanjutan untuk memperingati hari bumi pada 22 april lalu”. Ditemukan kalimat ke-2 pada dokumen Lombaessayukdw_seremonia.txt yaitu “badan eksekutif mahasiswa bem fakultas teologi dan bem fakultas bioteknologi universitas kristen duta wacana ukdw yogyakarta menyelenggarakan lomba penulisan esai 2020 dengan tema perubahan iklim dan pembangunan berkelanjutan untuk memperingati hari bumi pada tanggal 22 april lalu”. Ada kata “jakarta” pada dokumen Lombaessayukdw_medcom.txt yang tidak ditemukan pada dokumen Lombaessayukdw_seremonia.txt sehingga kalimat ke-2 dari dokumen Lombaessayukdw_seremonia.txt tidak bisa dikegorikan duplicate melainkan *Near-duplicate*.



Gambar 96. Contoh kalimat *Near-duplicate*

Kemudian, hasil pencarian kemiripan kalimat secara manual akan disusun dalam file csv dan dibandingkan dengan hasil pencarian kemiripan kalimat yang dilakukan oleh sistem. Lalu menghitung True Postive (TP), False Negative(FN), dan False Positive(FP) pada kedua tingkat kemiripan.

VII. HASIL DAN PEMBAHASAN

Hasil dan pembahasan berdasarkan metode penelitian dan pengujian yang telah dilakukan sebelumnya dipaparkan dalam dua sub bab berikut ini.

A. Hasil Pengujian

Pengujian dilakukan dengan 10 dokumen uji dan 30 dokumen latih. Contoh hasil deteksi sistem ditunjukkan pada Gambar 17.

Kalimat Ke	ID Dokumen	ID Kalimat	Tingkat Kemiripan
2	11dokter_seremonia.txt	2	Duplicate
4	Awsdeep_tagarid.txt	4	Duplicate
4	Inovasiproduk_siedoo.txt	7	Duplicate
5	Awsdeep_tagarid.txt	5	Duplicate
3	Awsdeep_tagarid.txt	3	Duplicate
11	Awsdeep_tagarid.txt	9	Duplicate
11	Designcamp_siedoo.txt	4	Duplicate
6	Innojogja_allrelease.txt	2	Duplicate
6	Innojogja_seremonia.txt	3	Duplicate
6	Innojogja_simpony.txt	3	Duplicate
12	Awsdeep_tagarid.txt	10	Duplicate
12	Gooddesign_allrealease.txt	1	Duplicate
12	Gooddesign_allrealease.txt	2	Duplicate
10	Awsdeep_tagarid.txt	8	Duplicate
1	Hibahinovasi_ebahana.txt	2	Near-Duplicate
8	11dokter_gudegnet.txt	6	Near-Duplicate
13	Awsdeep_tagarid.txt	11	Near-Duplicate

Gambar 17. Contoh hasil deteksi sistem pada dokumen uji “Awsdeep_siedoo.txt”

Sistem hanya menampilkan data 2 tingkat kemiripan yaitu Duplicate dengan jarak hamming 0 dan *Near-Duplicate*

dengan jarak hamming 1,2,3,4 dan 5. Data dengan jarak hamming >=6 tidak ditampilkan karena data tersebut dianggap tidak dalam kategori mirip atau hampir mirip.

Pada gambar di atas hasil deteksi sistem terhadap dokumen Awsdeep_siedoo.txt yang menghasilkan 14 data duplikat dan 3 data hampir duplikat. Kalimat ke-2 dokumen Awsdeep_siedoo.txt diduga duplikat dengan kalimat ke-2 dokumen 11 dokter_seremonia.txt. Kalimat ke-1 diduga hampir duplikat dengan kalimat ke-2 dokumen Hibahinovasi_ebahana.txt. Hasil pengujian yang disusun berdasarkan *confusion matrix* dan dievaluasi menggunakan metric F-score ditunjukkan pada Tabel 2 dan Tabel 3, dimana Tabel 2 menunjukkan nilai untuk kategori duplikat, sedangkan Tabel 3 untuk kategori hamper duplikat.

Tabel 2
Hasil perhitungan nilai evaluasi dan F1 score setiap dokumen terhadap tingkat kemiripan *duplicate*

No	Nama Dokumen	*Duplicate*		
		Precision	***Recall***	***F1 Score***
1	Awsdeep_siedoo.txt	0.384615385	1	0.555555556
2	Designcamp_jogyacom.txt	0.379310345	1	0.55
3	Hongkong_seremonia.txt	0.142857143	1	0.25
4	Jointsummer_medcom.txt	0.076923077	1	0.142857143
5	Kknntt_wartakita.txt	0.05	1	0.095238095
6	Lulusan150_krjogja.txt	0	0	0
7	Rmo2019_jogjatribun.txt	0.230769231	1	0.375
8	Glpindo_medcom.txt	0.269230769	1	0.424242424
9	Lombaessayukdw_medcom.txt	0.384615385	1	0.555555556
10	Startup_medcom.txt	0.379310345	1	0.55

Tabel 3
Hasil perhitungan nilai evaluasi dan F1 score setiap dokumen terhadap dua tingkat kemiripan *near-duplicate*

No	Nama Dokumen	*Near-Duplicate*		
		Precision	***Recall***	***F1 Score***
1	Awsdeep_siedoo.txt	0.333333333	0.25	0.285714286
2	Designcamp_jogyacom.txt	0	0	0
3	Hongkong_seremonia.txt	0	0	0
4	Jointsummer_medcom.txt	0	0	0
5	Kknntt_wartakita.txt	0	0	0
6	Lulusan150_krjogja.txt	0	0	0
7	Rmo2019_jogjatribun.txt	0	0	0
8	Glpindo_medcom.txt	0	0	0
9	Lombaessayukdw_medcom.txt	0.581818182	0	0
10	Startup_medcom.txt	0	0	0

B. Pembahasan

Dalam memilih jarak hamming terkecil, ada beberapa hal yang dipertimbangkan oleh penulis. Awalnya penulis menerapkan dua cara yang berbeda untuk mengelompokkan tingkat kemiripan yaitu Duplikat dan Hampir Duplikat. Untuk Duplikat sistem menggunakan fingerprint data dengan jarak hamming 0 untuk mendapatkan data duplikat lainnya tujuannya agar data uji yang memiliki data latih lebih dari 1 dokumen dapat terdeteksi. Namun setelah implementasi dan diuji, ternyata pengelompokkan dengan cara tersebut menimbulkan banyaknya data yang terambil oleh sistem. Sedangkan sebuah kalimat latih bisa saja memiliki fingerprint yang sama dengan kalimat uji tanpa memiliki kesamaan topik. Sistem menerapkan cara pengelompokkan data Hampir Duplikat yaitu mengambil data latih apabila data tepat selanjutnya merupakan data uji dengan jarak Hamming 1,2,3,4 dan 5.

Berdasarkan hasil pengujian yang dilakukan penulis, mendeteksi teks duplikat memberikan hasil yang lebih baik dibandingkan mendeteksi teks Hampir Duplikat. Rata-rata nilai evaluasi Recall mencapai 80% untuk deteksi teks duplikat. Sistem mampu menemukan kalimat duplikatnya dengan sempurna di 8 dokumen dari total 10 dokumen uji. Namun deteksi teks duplikat juga memiliki kekurangan. Terlihat dari nilai rata-rata presisi yaitu 27%, ini menunjukkan dari banyaknya dokumen duplikat yang terdeteksi oleh sistem, namun hanya sedikit yang benar-benar duplikat (TP).

Tabel 4
Contoh detail hasil deteksi sistem untuk kategori Hampir Duplikat

Awsdeep_siedoo.txt	**Hibahinovasi_ebahana.txt**
mahasiswi ukdw jawara aws deepracer women s league indonesia	com pendidikan hak warga negara indonesia warga berkebutuhan khusus
	Awsdeep_tagarid.txt
thailand negara pembuka kejuaraan virtual aws deepracer women s league digelar 1 agustus 2020 seminggu balap virtual digelar malaysia 8 agustus 2020 indonesia 15 agustus 2020 balap virtual digelar filipina 22 agustus 2020 singapura 29 agustus 2020 penyelenggaraan babak grand final september 2020	thailand negara pembuka kejuaraan virtual aws deepracer women s league digelar 1 agustus 2020 seminggu balap virtual digelar malaysia 8 agustus 2020 indonesia 15 agustus 2020

Detail perbandingan kalimat di atas menunjukkan bahwa kalimat ke 1 Awsdeep_siedoo.txt dengan kalimat ke 2 Hibahinovasi_ebahana.txt tidak memiliki kesamaan arti atau makna. Namun secara leksikal teks pada kedua kolom ada yang memiliki beberapa kata yang sama seperti pada dokumen Hibahinovasi_seremonia.txt, terdapat kata

JUTEI Edisi Volume.6 No.2 Oktober 2022
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2022.62.216

"indonesia" pada kedua kolom. Adanya beberapa kata yang sama ini mendorong penulis untuk melakukan evaluasi secara leksikal terhadap hasil sistem dan diperoleh pada tabel 5

Tabel 5
Beberapa Contoh evaluasi leksikal hasil deteksi sistem terhadap dokumen "Awsdeep_siedoo.txt"

ID Kalimat	Kalimat Uji	Kalimat Latih	Jumlah yang Sama	Total Token
2	siedoo nathania saphira mahasiswi program studi prodi informatika universitas kristen duta wacana ukdw yogyakarta berhasil memenangkan aws deepracer women s league indonesia digelar tanggal 15 agustus 2020	1 februari 2020 bertempat ballroom indraprasta 3 hotel sahid yogyakarta fakultas kedokteran universitas kristen duta wacana fk ukdw yogyakarta melantik 11 dokter acara sumpah dokter periode xv	14	54
4	nathania berhasil meraih tercepat 0 11 416 berhak mewakili indonesia babak grand final tingkat asia tenggara	nathania berhasil meraih tercepat 0 11 416 berhak mewakili indonesia babak grand final tingkat asia tenggara	32	32
5	nathania babak grand final tingkat asia tenggara indonesia diwakili mahanti indah rahajeng mahasiswi institut teknologi bandung itb kejuaran aws deepracer tingkat nasional menduduki peringkat	nathania babak grand final tingkat asia tenggara indonesia diwakili mahanti indah rahajeng mahasiswi institut teknologi bandung itb kejuaran aws deepracer tingkat nasional menduduki peringkat	48	48
3	nathania bersaing tingkat asia tenggara pemenang singapura malaysia thailand filipina september	nathania bersaing tingkat asia tenggara pemenang singapura malaysia thailand filipina september	22	22
11	kondisi normal acara diselenggarakan offline situasi covid 19	kondisi normal acara diselenggarakan offline situasi covid 19	20	20
	diselenggarakan online	diselenggarakan online		
11	kondisi normal acara diselenggarakan offline situasi covid 19 diselenggarakan online	mengusung tema bamboo experience for creative millennials kegiatan diharapkan meningkatkan sinergi pendidikan desain produk indonesia dunia usaha dunia industri dudi	0	30
6	willy sudiarto raharjo s kom m cs aws deepracer women s league ajang mahasiswi	acara didukung nus enterprise innovation factory dinas komunikasi informatika daerah istimewa yogyakarta diy	0	27
12	acara diselenggarakan acara build on asean 2020	acara diselenggarakan acara build on asean 2020	14	14
13	thailand negara pembuka kejuaraan virtual aws deepracer women s league digelar 1 agustus 2020 seminggu balap virtual digelar malaysia 8 agustus 2020 indonesia 15 agustus 2020 balap virtual digelar filipina 22 agustus 2020 singapura 29 agustus 2020 penyelenggaraan babak grand final september 2020	thailand negara pembuka kejuaraan virtual aws deepracer women s league digelar 1 agustus 2020 seminggu balap virtual digelar malaysia 8 agustus 2020 indonesia 15 agustus 2020	58	69

Highligt warna kuning merupakan token yang sama terdapat pada dokumen uji dan dokumen latih. Kalimat ke 4 yaitu "nathania berhasil meraih tercepat 0 11 416 berhak mewakili indonesia babak grand final tingkat asia tenggara" terdapat pada kalimat ke 4 dokumen Awsdeep_tagarid.txt dan jumlah token yang sama terdapat pada kedua kolom adalah 14 dan total token adalah 14. Sedangkan pada kalimat ke 12 token yang sama hanya "2020" sehingga jumlah token yang sama adalah 2 yaitu 1 pada dokumen uji dan 1 pada dokumen latih, dan total token adalah 15 yang merupakan

total panjang kedua token. Pada tabel 6 dipaparkan persentase hasil deteksi sistem secara leksikal.

Tabel 6
Persentase hasil deteksi secara leksikal

NO	Dokumen Uji	Duplicate	Near-duplicate
1	Awsdeep_siedoo.txt	45.58%	35.89%
2	Designcamp_jogyacom.txt	82.97%	6.58%
3	Hongkong_seremonia.txt	46.51%	3.77%
4	Jointsummer_medcom.txt	17.36%	16.01%
5	Kknntt_wartakita.txt	22.27%	4.23%
6	Lulusan150_krjogja.txt	19.62%	4.76%
7	Rmo2019_jogjatribun.txt	57.94%	8.52%
8	Glpindo_medcom.txt	62.25%	3.13%
9	Lombaessayukdw_medcom.txt	79.17%	4.00%
10	Startup_medcom.txt	0.57%	0.77%

Pada tabel 6, tingkat kemiripan duplicate memiliki persentase kemiripan lebih tinggi dibandingkan tingkat kemiripan near-duplicate seperti yang diharapkan. Perbedaan paling menonjol terdapat pada dokumen Designcamp_jogyacom.txt, sistem mendeteksi kesamaan token mencapai 82.97% dari 100% token sedangkan deteksi near-duplicate 6.58%. Dari 10 dokumen uji hanya 1 percobaan yang menghasilkan deteksi near-duplicate lebih tinggi dari deteksi duplikat yaitu pada dokumen Startup_medcom.txt. Hal ini terjadi karna penulis sengaja tidak menyediakan dokumen berita yang mirip dengan dokumen ke-10 tersebut dengan tujuan mendapatkan keakuratan pengelompokan tingkat kemiripan. Dan dihasilkan persentase duplikatnya lebih rendah dibandingkan near-duplicate. Penulis menerapkan algoritma Simhash dalam ekstraksi fitur dokumen untuk mendeteksi kemiripan teks dan dihasilkan sistem dengan nilai recall 80%, precision 27% dan F1-score 37% untuk deteksi teks duplikat.

VIII. KESIMPULAN

Berdasarkan penelitian yang dilakukan oleh penulis dalam menerapkan algoritma *Simhash* untuk mendeteksi kemiripan teks, penulis menarik beberapa kesimpulan berikut:

1. Penerapan algoritma Simhash untuk mendeteksi kemiripan dengan tingkat kemiripan dibawah 100% yaitu Near-duplicate adalah F1-score bernilai 0.028, precision 0.033 dan recall 0.025.
2. Pengujian algoritma Simhash untuk deteksi Near-duplicate secara leksikal lebih baik dibandingkan secara semantik.
3. Nilai evaluasi tingkat kemiripan Duplicate memiliki rata-rata precision 0,27 dan recall 0,8 dan F1 score 0,37. Nilai recall yang cukup tinggi menunjukkan bahwa sistem cukup berhasil mendeteksi kalimat duplikat. Meskipun demikian sistem mengambil terlalu banyak dokumen sehingga membuat nilai presisi hanya berrnilai 0,27.
4. Sistem dirancang untuk menggabungkan segmentasi kalimat yang sangat pendek dengan panjang < 6 kata dengan segmentasi kalimat berikutnya. Sebagai akibatnya kalimat gabungan tersebut tidak terdeteksi sebagai kalimat duplicate.

Berdasarkan dari pengkajian penelitian ini, ada beberapa hal yang belum dilakukan namun diasumsikan akan memperbaiki kinerja siste. Untuk itu disarankan menggunakan n-gram kata atau karakter dalam mendapatkan nilai hash sebelum proses binerisasi dlakukan. Yang kedua, hasil luaran sistem dapat ditampilkan dengan diberi penekanan atau *highlight* bagi teks yang terduga mirip dibandingkan dengan menampilkan luaran sistem dalam bentuk tabel.

REFERENSI

[1] L. D. Krisnawati and K. U. Schulz, "Significant Word-based Text Alignment for Text Reuse Detection," in *Conference: Int. Conference on Research and Innovation in Computer, Electronic, and Manufacturing Enginerring (RICEME-17)*, Denpasar, Bali, 2017.

[2] L. Krisnawati, "The use of phraseword and local-weighted terms as features for text reuse and plagiarism detection," in *Seminar Hasil Penelitian Bagi Civitas Akademika UKDW*, Yogyakarta, Indonesia, 2017.

[3] L. D. Krisnawati, "Plagiarism Detection for Indonesian Texts," Muenchen, 2016.

[4] M. Coe, "Website Indexing," *Indexer*, vol. 34, no. 1, pp. 20-25, 2016.

[5] L. Pamulaparty, C. Rao and M. Rao, "A Near-Duplicate Detection Algorithm to Facilitate Document Clustering," *Intl. Journal of Data Mining and Knowledge Management Process*, vol. 4, no. 5, pp. 39-49, 2014.

[6] K. Williams and C. L. Giles, "Near Duplicate Detection in an Academic Digital Library," in *Proceedings of the 2013 ACM symposium on Document engineering*, 2013.

[7] M. Burgess, E. Giraudy, J. Katz-Samule and J. Walsh, "The Legislative Influence Detector: Finding Text Reuse in State Legislation," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data*, California, 2016.

[8] R. Yandrapally, A. Stocco and A. Mesbah, "Inference, Near-Duplicate Detection in Web App Model," in *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, Seol, Korea, 2020.

[9] M. Moritz, W. A. B. Pavlek, Y. Bizzoni and M. Buchler, "Non-Literal Text Reuse in Historical Texts: An Approach to Identify Reuse Transformations and its Application to Bible Reuse," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, 2016.

[10] T. Hoad and J. Zobel, "Methods for Identifying Versioned and Plagiarized Documents," *Journal of the American Society for Informtion Science and Technology*, vol. 54, no. 3, pp. 203-215, 2003.

JUTEI Edisi Volume.6 No.2 Oktober 2022
ISSN 2579-3675, e-ISSN 2579-5538
DOI 10.21460/jutei.2022.62.216

[11] M. S. Charikar, "Similarity Estimation Techniques from Rounding Algorithms," in *Proceedings of the 34th Annual ACM Symposium on Theory of Computing,*, 2002.

[12] F. Naumann and M. Herschel, "An Introduction to Duplicate Detection," in *Synthesis Lectures on Data MAnagement*, Postdam, Morgan & Claypool Publisher, 2010, pp. 1-87.

[13] T. Kopelowitz and E. Porat, "A Simple Algorithm for Approximating the Text-To-Pattern Hamming Distance," in *1st Symposium on Simplicity in Algorithms*, Dagstuhl, 2018.

[14] N. C. Haryanto, L. D. Krisnawati and . A. R. Chrismanto, "Retrieval of source documents in a text reuse system," 2020.

JUTEI Edisi Volume.6 No.2 Oktober 2022
ISSN 2579-3675, e-ISSN 2579-5538