Available at https://jtsiskom.undip.ac.id (20 March 2020)
DOI:10.14710/jtsiskom.8.2.2020.140-149

Jurnal Teknologi dan Sistem Komputer, 8(2), 2020, 140-149

Temu kembali dokumen sumber rujukan dalam sistem daur ulang teks

Retrieval of source documents in a text reuse system

Nathaniel Clarence Haryanto, Lucia Dwi Krisnawati[*)], Antonius Rachmat Chrismanto

Program Studi Informatika, Universitas Kristen Duta Wacana
Jl. Dr. Wahidin Sudirohusodo no 5-25, Yogyakarta , Indonesia 55224

Cara sitasi: N. C. Haryanto, L. D. Krisnawati, and A. P. Chrismanto, "Temu kembali dokumen sumber rujukan dalam sistem daur ulang teks," *Jurnal Teknologi dan Sistem Komputer*, vol. 8, no. 2, pp. 140-150, 2020. doi: 10.14710/jtsiskom.8.2.2020.140-149, [Online].

Abstract **-** The architecture of the text-reuse detection system consists of three main modules, i.e., source retrieval, text analysis, and knowledge-based postprocessing. Each module plays an important role in the accuracy rate of the detection outputs. Therefore, this research focuses on developing the source retrieval system in cases where the source documents have been obfuscated in different levels. Two steps of term weighting were applied to get such documents. The first was the local-word weighting, which has been applied to the test or reused documents to select query per text segments. The tf-idf term weighting was applied for indexing all documents in the corpus and as the basis for computing cosine similarity between the queries per segment and the documents in the corpus. A two-step filtering technique was applied to get the source document candidates. Using artificial cases of text reuse testing, the system achieves the same rates of precision and recall that are 0.967, while the recall rate for the simulated cases of reused text is 0.66.

Keywords *– text reuse detection; source retrieval; significant words; local-word weighting scheme*

Abstrak **-** Arsitektur deteksi daur ulang teks terdiri dari tiga modul utama yakni temu kembali dokumen sumber, analisis teks, dan pascapemrosesan. Masing-masing modul memiliki peran penting dalam menentukan akurasi luaran deteksi. Penelitian ini berfokus pada pembangunan sistem temu kembali dokumen sumber yang telah didaur ulang dengan berbagai jenis penyamaran. Dalam penelitian ini, proses temu kembali dikembangkan dengan memanfaatkan pemilihan kata kunci yang diberi bobot lokal. Setelah kata kunci ditemukan, dokumen kandidat dipilih dengan cara membandingkan query dengan pembobotan tf-idf dan ukuran kemiripan kosinus (*cosine similarity*). Dokumen yang terambil disaring berdasarkan nilai persamaan kosinus dan redudansi dokumen di tiap segmen hasil pencariannya. Dokumen uji dibedakan menjadi dokumen uji artifisial yang dibangkitkan secara algoritmik dan dokumen simulasi yang ditulis oleh manusia sebagai kasus daur ulang. Nilai presisi dan recall terbaik dari dokumen tes artifisial mencapai 0,967, sedangkan dari dokumen tes tersimulasi nilai diperoleh recall 0,66.

Kata kunci *– deteksi daur ulang teks; temu kembali dokumen sumber; kata-kata penting; pembobotan lokal*

I. PENDAHULUAN

Daur ulang teks (*text reuse*) dilakukan untuk menggunakan ulang sumber tulisan yang telah ada untuk menghasilkan sebuah teks yang baru [1]. Penggunaan ulang teks ini dapat berupa kutipan langsung, parafrase, maupun penggunaan kembali ide yang disampaikan oleh sumber tersebut dengan mencantumkan atau tanpa mencantumkan sumber aslinya. Dengan demikian, daur ulang teks (DaUT) ini bisa dikategorikan menjadi dua kelompok besar, yakni DaUT legal, yang bisa diterima oleh masyarakat dan DaUT ilegal, yakni DaUT yang tidak bisa diterima oleh masyarakat [2].

Salah satu contoh dari DaUT legal adalah penulisan berita yang bisa dijumpai di surat kabar yang berbeda oleh jurnalis yang berbeda, tetapi memiliki isi yang hampir identik. Namun demikian, penulisan berita ini tidak diklaim sebagai hasil plagiasi karena beritanya bersumber dari satu kantor berita. Di dunia akademis, DaUT tanpa menyebutkan sumbernya dinyatakan sebagai tindakan plagiasi yang ilegal, sedangkan DaUT yang mencantumkan sumber sitasi dinyatakan sebagai DaUT legal.

Peningkatan keberadaan daur ulang teks sendiri didorong oleh kemudahan akses internet yang membuat informasi mudah diperoleh. Kondisi inilah yang pada akhirnya mendorong munculnya deteksi daur ulang teks dan plagiasi [3]. Sebagian besar program aplikasi deteksi plagiasi sebenarnya mendeteksi DaUT, bahkan TurnItIn (www.turnitin.com) pun mendeteksi bagian teks yang diduga sama sekalipun tercantum sumber kutipannya sehingga tidak layak disebut sebagai teks plagiasi. Dengan demikian, deteksi plagiasi bisa diklaim sebagai bagian dari deteksi DaUT.

Alur kerja sistem deteksi daur ulang teks dipilah menjadi tiga modul utama, yaitu temu kembali sumber dokumen (*source retrieval*), analisis teks (*text*

[*)] Penulis korespondensi Lucia Dwi Krisnawati
Email: krisna@staff.ukdw.ac.id

Submitted: 17 October 2019; Revised: 26 February 2020; Accepted: 13 March 2020; Published: 30 April 2020

alignment), dan pascapemrosesan yang berbasiskan pengetahuan. Arsitektur sistem deteksi plagiasi ditunjukkan dalam Gambar 1 yang diadopsi oleh Potthast dkk. [4].

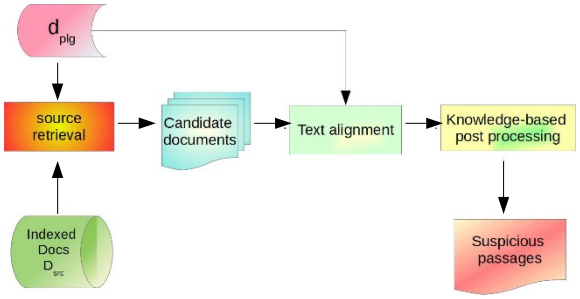
Menurut Krisnawati dan Schülz [2] tantangan sistem deteksi daur ulang teks termanifestasikan ke dalam dua tugas utamanya. Yang pertama adalah bagaimana sistem mampu memungut sekelompok kandidat dokumen dari korpus yang benar-benar mengandung dokumen sumber yang sesungguhnya tanpa dipengaruhi panjang-pendek kemiripan dokumen. Kedua adalah bagaimana mengekstraksi bagian dokumen yang didaur-ulang dengan berbagai cara penyamaran.

Dalam DaUT, penyamaran teks dikategorikan menjadi tiga, yakni penyamaran literal, manipulasi teks dan plagiasi semu [5]. Penyamaran literal adalah penjiplakan persis yang dilakukan dengan cara salin dan tempel (*copy paste*), dan kocok dan tempel (menyalin persis hanya mengubah susunan kalimatnya – *shake and paste*). Penyamaran yang dilakukan dengan mengubah teks, baik dengan cara melakukan parafrase, ringkasan, alih bahasa, maupun pengubahan media sumber gagasan, dikelompokkan dalam penyamaran manipulasi teks [5]. Plagiasi semu meliputi plagiasi karya sendiri dan plagiasi sumber sekunder. Gambar 2 menunjukkan bagan penyamaran DaUT berdasarkan delik perkaranya yang merupakan adaptasi dari taksonomi dalam [6].

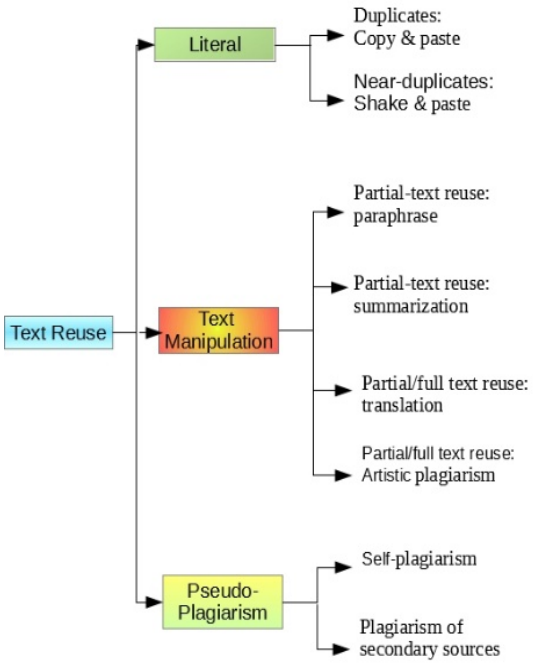
Berdasarkan tantangan DaUT serta memperhatikan tingkat penyamaran seperti yang diuraikan sebelumnya, maka Leilei dkk. [7] membangun sistem temu kembali dokumen sumber dengan menggunakan beberapa metode pembobotan, yakni metode *term frequency-inverse document frequency* (tf-idf), PatTree, dan *weighted tf-idf* untuk mendapatkan kata kunci dari dokumen uji. Kata kunci ini selanjutnya dijadikan *query* dalam sebuah mesin pencari yang bekerja secara daring. Hasil dari penelitian ini menunjukkan bahwa *query* yang disusun dari 10 kata kunci dengan nilai tf-idf tertinggi mendapatkan hasil yang lebih baik dengan nilai presisi 0,004 dan recall 0,2091.

Salah satu metode DaUT termutakhir (*state-of-the-art*) diajukan oleh Gipp [8] yang menggunakan pola-pola sitasi, baik dalam modul temu kembali maupun analisis teks. Hal ini dimungkinkan karena sistem DaUT yang dibangun tersebut merupakan sistem DaUT luring sehingga *quer*y maupun representasi dokumen memiliki bentuk non-token yang mengabaikan isi dokumen sumber maupun dokumen terdaur-ulang. Metode ini diklaim mampu mendeteksi dokumen sumber serta bagian teks y ang didaur ulang dengan berbagai penyamaran, bahkan penyamaran yang telah dialih-bahasakan. Sayangnya, metode ini tidak akan berfungsi dengan baik, jika dokumen sumber kutipan tidak dicantumkan dalam sitasi sehingga pola sitasi yang terbentukpun tidak akan pernah mampu menemukan dokumen sumber yang sesungguhnya.

Sebagian besar penelitian sistem DaUT untuk teks berbahasa Indonesia belum membagi alur kerja seperti yang ditunjukkan dalam Gambar 1. Suryana dkk. [9] menggunakan index *fingerprint* dalam bentuk pohon 2-3



Gambar 1. Sistem arsitektur deteksi daur ulang teks



Gambar 2. Taksonomi DaUT berdasarkan delik penyamarannya

untuk menyeleksi kandidat dokumen sumber. Jika *fingerprint* dokumen terduga didaur-ulang sama dengan *fingerprint* yang ada di *posting list*, maka frekuensinya akan dinaikkan. Nilai frekuensi inilah yang digunakan untuk mengeliminasi kandidat dokumen sumber. Syahputra [10] menerapkan metode Rolling-hash dan Winnowing untuk menentukan apakah sepasang dokumen memiliki kemiripan yang tinggi. Berbeda dari [9] dan [10], Alfikri dan Purwarianti [11] menyelesaikan permasalahan deteksi DaUT dengan pendekatan pembelajaran mesin terbimbing (*supervised learning*) yang menerapkan *support vector machine* (SVM) dan naïve Bayes untuk memprediksi bagian teks yang sama. Kurniati dkk. [12] menggunakan algoritme Winnowing untuk mendeteksi kemiripan judul tugas akhir.

Penelitian ini berfokus pada pembangunan sistem temu kembali kandidat dokumen yang menjadi sumber daur ulang atau lebih dikenal sebagai kandidat dokumen sumber sesuai Gambar 1. Dengan asumsi bahwa dokumen sumber tersembunyi dalam kumpulan dokumen atau korpus, maka sistem DaUT perlu dirancang untuk mampu mengidentifikasi dokumen

yang memiliki potensi sebagai dokumen sumber di modul temu kembali dokumen sumber (*source retrieval*). Luaran sistem temu kembali ini adalah kandidat dokumen sumber yang kemudian akan dianalisis lebih detil di modul analisis teks untuk mendapatkan bagian teks (*passages*) yang diduga mirip, serupa atau terdaur ulang. Bagian-bagian teks ini akan disaring berdasarkan pola-pola yang telah ditetapkan di dalam modul pascapemrosesan untuk memastikan bahwa bagian teks yang diduga terdaur-ulang tersebut benar-benar didaur ulang dari dokumen yang diduga sebagai sumbernya. Modul analisis teks dan pascapemrosesan berada di luar lingkup penelitian ini.

II. METODE PENELITIAN

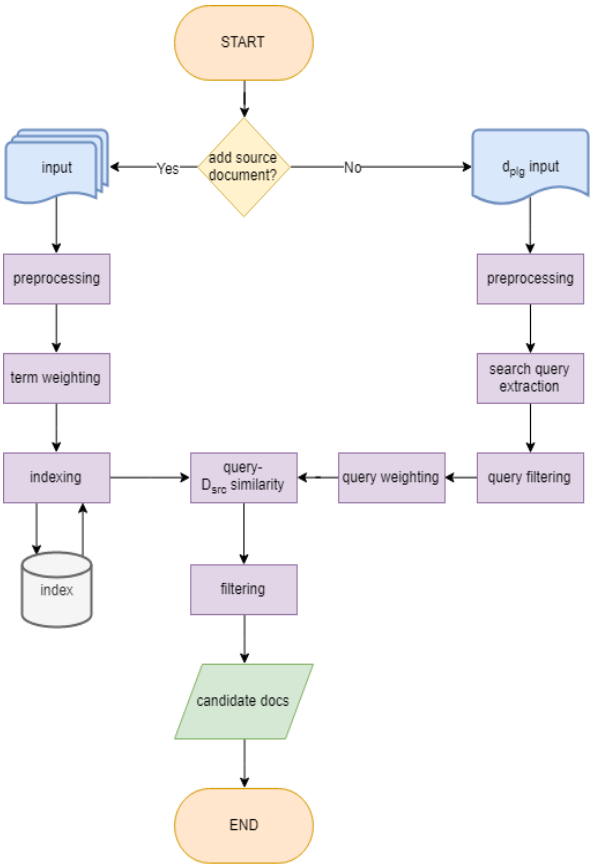
Arsitektur sistem DaUt lengkap terdiri dari tiga modul utama (Gambar 1). Penelitian ini berfokus pada modul pertama, yakni temu kembali dokumen sumber (*source document retrieval*) yang luarannya adalah kandidat dokumen sumber. Berkaitan dengan modul ini, Potthast dkk. [4] mengulas 17 artikel DaUT dan menyimpulkan bahwa modul temu kembali dokumen sumber terdiri dari beberapa proses, yaitu pra-pemrosesan teks, ekstraksi frasa-frasa kunci, pembentukan query, penghitungan kemiripan, dan proses penyaringan sebagai pascaproses. Penelitian ini mengikuti tahapan alur kerja yang didefinisikan dalam [4] dengan menambahkan segmentasi teks untuk ekstraksi kata-kuncinya.

Sistem temu kembali dokumen sumber ini dirancang sebagai sistem luring (*offline*). Langkah-langkah pembangunan sistem diawali dengan menerapkan prapemrosesan, pembobotan token, dan pengindeksan pada kumpulan dokumen atau korpus (D_{corp}). Indeks secara otomatis disusunsebagai acuan saat proses pencocokan kemiripan antara dokumen uji d_{plg} dengan dokumen D_{corp} dilakukan.

Dokumen yang terindikasi sebagai DaUT atau dokumen uji d_{plg} kemudian diproses melalui tahap prapemrosesan, ekstraksi *query* pencarian, formulasi *query*, dan pembobotan *query* sebelum dibandingkan dengan $d_{corp} \in D_{corp}$. Hasil pemrosesan d_{plg} kemudian disaring sehingga tertinggal dokumen yang digunakan sebagai dokumen kandidat sumber. Tahapan pembangunan sistem ditunjukkan dalam bentuk diagram alir dalam Gambar 3.

A. Prapemrosesan

Tahap prapemrosesan berlaku untuk D_{corp} dan D_{plg} sebelum diproses lebih lanjut. Pada prapemrosesan D_{plg} terdapat proses tambahan, yaitu segmentasi untuk membagi d_{plg} menjadi beberapa *window*. D_{corp} mewakili semua dokumen yang ada di korpus, baik dokumen sumber (D_{src}), dokumen duplikat (D_{dup}), maupun dokumen lainnya (D_{lain}), yaitu dokumen yang tidak termasuk ke dalam kategori yang telah disebutkan sebelumnya. Dengan demikian, D_{corp} bisa didefinisikan pada Persamaan 1.



Gambar 3. Diagram alir pembangunan sistem

Algoritme 1. Konversi docx ke txt

```
import docxpy

procedure convertDocx(file docx)
begin
1: text ⟵ docxpy.process(file docx)
2: saveFile ⟵ open("convertedFile.txt")
3: saveFile.write(text)
end procedure
```

$$D_{corp} = D_{src} \cup D_{dup} \cup D_{lain} \quad (1)$$

Korpus yang digunakan adalah korpus yang sama dalam [3]. Korpus ini merupakan kumpulan dokumen berbahasa Indonesia yang berasal dari artikel opini, saintifik, dan berita. Artikel-artikel tersebut memiliki format docx dan odt yang perlu diubah ke format txt.

Langkah awal di tahap praproses adalah konversi format dokumen yang menggunakan pustaka odf dan docxpy untuk mengubah teks dari format odt dan docx menjadi txt. Pustaka docxpy dapat secara langsung membaca seluruh isi teks dari *file* docx. Oleh karena itu, pustaka tersebut dapat digunakan dengan menjalankan *pseudocode* pada Algoritme 1.

Dalam melakukan pembacaan, pustaka odf memiliki perbedaan dengan docxpy. Pustaka odf melakukan pembacaan *file* odt per paragraf sehingga perlu melakukan penggabungan sebelum dituliskan ke *file*

Algoritme 2. Konversi odf ke txt

```
from odf.opendocument import load
from odf import text.teletype

procedure convertOdf(file odf)
begin
1:  text ⟵ docxpy.process(file odf)
2:  textfile ⟵ load(file odf)
3:  allparas ⟵textfile.getElementsByType(text.P)
4:  i ⟵ 0
5:  for texts in allparas do
6:      txtString[i] ⟵ texts
7:      i ⟵ i + 1
8:  end for
9:  saveFile ⟵ open(“convertedFile.txt”)
10: saveFile.write(‘ ‘.join(txtString))
end procedure
```

luaran. Penggunaan odf dapat dilakukan dengan menjalankan *pseudocode* pada Algoritme 2.

Langkah berikutnya adalah normalisasi teks yang telah memiliki format teks (.txt). Dokumen teks tersebut kemudian diproses dengan menggunakan pustaka *natural language toolkit* (NLTK) dan Sastrawi. Pustaka Sastrawi yang digunakan adalah pustaka untuk menghilangkan *stopword* dalam bahasa Indonesia, sedangkan NLTK digunakan untuk membentuk *token* dari dokumen. Selain menghilangkan *stopword*, karakter khusus beserta angka juga dihilangkan dengan cara mengantikannya dengan spasi kosong di tahap ini.

Teks yang diindeks perlu dibaca terlebih dahulu dalam bentuk *string*. Teks kemudian diproses dengan menjalankan *pseudocode* pada Algoritme 3. Pustaka *re* (*regular expression*) digunakan untuk mengubah karakter khusus menjadi spasi, menemukan dan menghapus angka. Berbeda dengan karakter khusus, angka pada proses ini tidak digantikan dengan spasi, tetapi langsung dihilangkan. *Stopword* pada variabel *txtString* dihilangkan dengan menggunakan pustaka Sastrawi. Pustaka NLTK digunakan untuk proses tokenisasi.

Proses selanjutnya adalah segmentasi teks yang hanya diberlakukan pada D_{plg} yang sebelumnya juga mengalami proses konversi format, normalisasi teks, dan tokenisasi. Setelah itu, segmentasi dilakukan terhadap teks. Tiap teks terbagi menjadi beberapa segmen dengan 250 token di setiap segmennya. Setelah dilakukan segmentasi, dilakukan penghitungan bobot token dengan menggunakan Persamaan 2. Bobot lokal yang dihasilkan dicari nilai ambangnya dengan menggunakan Persamaan 3. Dengan demikian, token yang muncul di segmen yang berbeda memiliki nilai bobot yang berbeda pula karena setiap segmen memiliki nilai ambang yang berbeda-beda tergantung dari bobot lokal kata dalam segmen tersebut.

$$WLScore = \alpha \times TF + (1-\alpha) \times SCount \quad (2)$$

Algoritme 3. Prapemrosesan

```
Input: text from document
Output: tokenized word
import re
import nltk
from string import digit
from Sastrawi.StopWordRemover.StopWordRemover
    Factory import StopWordRemoverFactory

procedure preProcessing (documentText)
begin
1:  factory ⟵ StopWordRemoverFactory()
2:  stopwords ⟵ factory.create_stopword_remover()
3:  doc ⟵ remove digits from documentText
4:  doc ⟵ replace special character with ‘’ from doc
5:  doc ⟵ lowercase doc
6:  doc ⟵ stopword.remove(doc)
7:  return nltk.word_tokenize(doc)
end procedure
```

Parameter α merepresentasikan nilai parameter (0-1) dan ditetapkan dengan nilai 0,5. Parameter *TF* menunjukkan *term frequency*, yaitu frekuensi *token*/kata yang dinormalisasi atau frekuensi relatif (*relative frequency*). *SCount* mengacu pada jumlah segmen (*window*) yang mengandung *token* tertentu dibagi dengan jumlah total segmen. Persamaan 2 menghitung bobot kata secara lokal, sedangkan seleksi kata kunci dilakukan dengan Persamaan 3. Parameter *PF* merupakan *pruning factor* yang nilainya menunjukkan prosentase banyaknya fitur yang hendak digunakan. Nilai yang ditetapkan sebagai *PF* adalah 0,6.

$$WLScoreThreshold = \frac{\sum_i WLScore_i}{\#text\ words} \times PF \quad (3)$$

B. Pengindeksan

Proses pengindeksan adalah proses pembuatan indeks untuk mempermudah dan mempercepat proses perbandingan antara D_{corp} dengan d_{plg} pada proses temu kembali. Dalam indeks yang dibuat, *token* beserta bobot token tersebut disimpan sesuai dengan aturan yang telah ditetapkan. Tahap ini diterapkan terhadap D_{corp} yang dijabarkan dalam Persamaan 1. Data yang diindeks berasal dari D_{corp} yang digunakan pada [2] dan [3]. Pembobotan token di $d_{corp} \in D_{corp}$ dilakukan dengan menerapkan tf-idf yang dijabarkan di Persamaan 4. Dalam menghitung nilai tf-idf, *token* yang ada dihitung frekuensi atau jumlah kemunculannya dalam sebuah dokumen. Selain itu, dihitung frekuensi dokumen, yakni jumlah dokumen dimana kata tersebut muncul (*df*).

$$tf - idf_t = tf_t \cdot \log_{10} \frac{N}{df_t} \quad (4)$$

Posting list merupakan bagian dari indeks yang berfungsi sebagai basis data dan berisi nilai tf-idf tiap kata dalam dokumen tertentu. Dalam sistem ini, *posting*

list yang dibentuk tidak menyimpan dokumen yang memiliki nilai tf-idf nol. Basis data disimpan dalam dokumen csv yang berisi kata beserta nilai tf-idf dalam dokumen tertentu. Bentuk penyimpanan data *posting list* dalam dokumen csv dapat dilihat dalam Tabel 1. *Posting list* yang dicontohkan tersebut merupakan cara penyimpanan yang disarankan dalam [13].

C. Ekstraksi query pencarian

Proses ekstraksi *query* diterapkan pada dokumen uji atau d_{plg} saja. Sebelum ekstraksi *query* dilakukan, dilakukan prapemrosesan yang menghasilkan *token* yang dibagi menjadi beberapa segmen. Keseluruhan *token* dokumen disegmen menjadi 250 *token* per segmen. Setelah itu, token dihitung bobotnya dengan menggunakan Persamaan 2. Bobot lokal yang dihasilkan kemudian dicari nilai ambangnya dengan menggunakan Persamaan 3. Setiap segmen memiliki nilai ambang (*threshold)* yang berbeda-beda tergantung dari bobot lokal kata dalam segmen tersebut.

D. Formulasi query

Dalam formulasi *query*, pertama-tama d_{plg} dibagi menggunakan segmen. *Query* dipilih menggunakan metode pembobotan lokal kata yang telah digunakan pada [14]. Perhitungan bobot lokal kata per segmen menggunakan Persamaan 2 dan Persamaan 3. Implementasi kedua persamaan tersebut tersebut dijabarkan dalam pseudocode yang ditunjukkan pada Algoritme 4. Setelah nilai ambang (*threshold*) untuk setiap segmen ditemukan, kata-kata dalam setiap segmen diseleksi sehingga tersisa kata-kata yang memiliki bobot di atas nilai ambang. Kata-kata tersebut diurutkan mulai dari bobot tertinggi hingga terendah. *Query* yang diambil adalah sepuluh token dengan bobot tertinggi dari setiap segmen.

Query yang sudah diperoleh, tidak langsung digunakan sebagai *query* pencarian. Apabila jumlah segmen lebih dari tiga, maka dilakukan seleksi lanjutan. Kata yang sama di segmen berikutnya dihapus. Jika sebuah *query* dari sebuah segmen berjumlah kurang dari lima token, maka kata-kata dalam query tersebut digabungkan (*merging*) dengan *query* sebelumnya. Penyatuan *query* ini bertujuan untuk mengurangi bias yang terjadi karena jumlah *token* yang tak seimbang dalam tiap *query*-nya.. Dengan cara ini, dokumen yang didapatkan menjadi lebih relevan dibanding hanya dengan menggunakan *query* yang kurang dari lima token.

E. Pengukuran kemiripan query dengan dokumen

Salah satu metode pengukuran kemiripan dokumen yang dapat digunakan adalah metode *vector space model* (VSM) yang digunakan dalam [14]. Metode VSM adalah metode yang digunakan untuk merepresentasikan sebuah dokumen dengan vektor. Metode VSM yang diterapkan dalam [15] membagi dokumen menjadi kalimat yang kemudian ditokenisasi. Metode VSM digunakan untuk mengukur seberapa

Tabel 1. Ilustrasi indeks dan *posting list* yang digunakan dalam penelitian ini

Indeks	Daftar Posting
arsitektur	{'ARS0001.txt': 113.167, 'ARS0002.txt': 10.288, 'ARS0003.txt': 5.144, …, 'SEN0301.txt': 2.572}
nusantara	{'ARS0001.txt': 12.046, 'ARS0004.txt': 4.015, 'BAH0014.txt': 10.038, …, 'SEN0306.txt': 4.015}
ala	{'ARS0001.txt': 2.526, 'BAH0019.txt': 2.526, 'BAH0043.txt': 2.526, …, 'SEN0290.txt': 7.579}

Algoritme 4. Pembobotan lokal kata

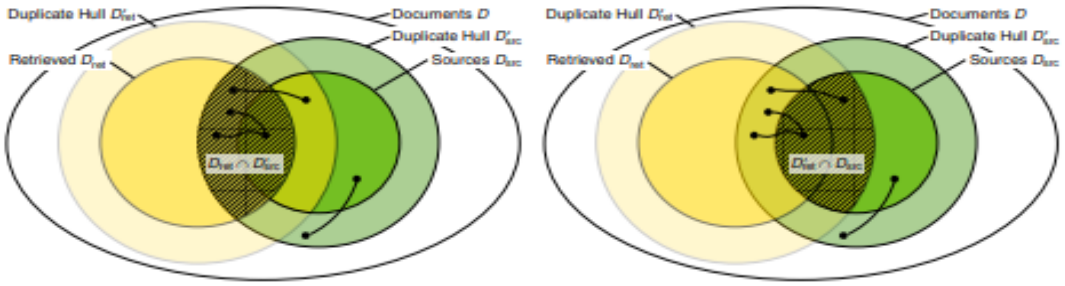
Input: local score of word, word list each section
Output: search query
Algoritme
1: local_word_threshold ⟵ WLScoreThreshold (Eq. 3)
2: **for** word **in** word_list **do**
3: **if** local_word_score < local_word_threshold **then**
4: delete word from word_list
5: **end if**
5: **end for**
6: sort word_list by local_word_score descending
7: **return** top 10 word_list

dekat kemiripan dokumen satu dengan lainnya. Penelitian ini menggunakan pembobotan *term frequency-inverse document frequency* (tf-idf) dan kemiripan kosinus sehingga kemiripan topik bahasan antara satu dokumen dengan lainnya dapat diketahui.

Teknik tf-idf melakukan pembobotan *term* dengan mengambil kata-kata penting berdasarkan frekuensi kemunculannya dalam dokumen. Pada prosesnya, metode ini digunakan untuk memberi nilai/bobot dari suatu kata dalam dokumen. Tf-idf dapat memberi nilai secara proporsional dari suatu kata sesuai dengan jumlah kemunculannya pada dokumen. Semakin banyak kemunculan suatu kata di sebuah dokumen, namun semakin jarang kata tersebut muncul di dokumen lainnya, maka semakin besar pula nilai tf-idf kata tersebut.

Tf-idf disebut sebagai pembobotan kata secara global karena menghitung kemunculan di keseluruhan dokumen dan kemunculan kata di korpus.. Pada metode ini, nilai yang dihasilkan merepresentasikan topik dari dokumen itu sendiri. Pembobotan ini tidak hanya terbatas pada dokumen itu sendiri, namun bergantung pada dokumen pada korpus yang digunakan [3].

Setelah dilakukan pembobotan dengan metode tf-idf, penhitungan kemiripan dilakukan antara *query* yang dibentuk dari d_{plg} dengan dokumen-dokumen yang ada di korpus. Metrik kemiripan yang digunakan adalah *cosine similarity* yang merupakan metode untuk mengkomputasi kemiripan dari dua buah dokumen. Metode ini memiliki tiga buah keunggulan, yaitu dapat



Gambar 4. Presisi dan recall untuk kasus DaUT [4]

mengukur kemiripan secara global dari dua buah dokumen, dapat mengatasi masalah pembobotan yang diakibatkan oleh panjang yang berbeda, dan memberi bobot tinggi untuk kata dengan frekuensi dokumen (*df*) rendah [3].

Penelitian ini menggunakan metode *cosine similarity* untuk mengukur kemiripan *query* dengan nilai yang dihasilkan dari tf-idf. Persamaan 5 digunakan untuk mendapatkan kemiripan dari *query* yang didapatkan dengan dokumen yang dicari. Simbol d_i merepresentasikan vektor dokumen yang ada di D_{corp}, sedangkan q_i merepresentasikan vektor dokumen uji d_{plg}. Dokumen kandidat dapat diperoleh menggunakan Persamaan 4 dan Persamaan 5 sesuai dengan alur yang ditunjukkan dalam Algoritme 5.

$$S_{\cos} = \frac{\sum_{i=1}^{|d|} \vec{d_i} \vec{q_i}}{\sqrt{\sum_{i=1}^{|d|} \vec{d_i}^2} \sqrt{\sum_{i=1}^{|d|} \vec{q_i}^2}} \quad (5)$$

F. Penyaringan atau filtering

Kemiripan antara dokumen sumber dengan *query* yang ada dicari dengan menggunakan Algoritme 5. Vektor yang digunakan adalah nilai tf-idf kata tersebut pada d_{plg} beserta nilai tf-idf yang diambil dari indeks. Dari seluruh indeks, dokumen yang mengandung kata tertentu dihitung kemiripannya masing-masing dengan *query* yang ada.

Setelah didapatkan hasil pencarian, penyaringan kembali dilakukan untuk menggabungkan dokumen yang sama beserta dokumen duplikat yang ikut masuk dalam hasil pencarian. Setelah dokumen terambil dua kali dari *query* berbeda, dokumen dicari duplikatnya. Pencarian dokumen duplikat dilakukan dengan menggunakan database dari file .csv yang sudah berisi nama dokumen dan duplikatnya.

G. Metriks ukuran pengujian

Penelitian ini menggunakan cara seperti Potthast dkk. [4] untuk memperoleh presisi dan recall yang lebih baik untuk mendeteksi teks (Gambar 4). Cara tersebut dilakukan dengan menambahkan dokumen yang ditandai sebagai duplikat atau sangat mirip dengan salah satu dokumen sumber (D_{src}). Namun, dokumen duplikat tersebut tidak terdeteksi sebagai *true positive* karena tidak ada dalam pelabelan dalam sumber (D_{src}). Algoritme 5 menunjukkan secara lebih jelas tentang cara pemilihan dokumen sehingga dapat meningkatkan presisi dan recall.

Algoritme 5. Pencarian dokumen kandidat

Input: search query, posting list
Output: candidate document
Algoritme
1: **for** word **in** search_query **do**
2: calculate dot product for each doc containing word
3: **end for**
4: **for** word **in** documents **do**
5: calculate cosine similarity with search_query
6: **end for**
7: sort documents by cosine similarity descending
8: **return** top 10 documents for each search_query

Parameter D_{dup} menunjukkan dokumen yang tidak mendapat label dokumen sumber, tetapi memiliki persentase kemiripan yang cukup tinggi dengan salah satu dokumen sumber. Jika ditemukan oleh sistem, maka d_{dup} biasanya dianggap sebagai *true positive*. Dokumen d_{dup} seharusnya dapat dimasukkan ke dalam D_{src} karena pada kenyataannya banyak dokumen yang didapatkan dengan cara *crawling* merupakan duplikat satu sama lain atau sangat mirip. Dengan alasan inilah Potthast dkk. [4] memasukkan $d_{dup} \in D_{dup}$ ke dalam D'_{src}, sehingga D_{src} menjadi $D`_{src}$. Secara matematis, $D`_{src}$ dan $D`_{ret}$ dinyatakan dengan Persamaan 6 dan Persamaan 7.

$$D'_{src} = \{ d_{dup} \vee d_{dup} \in D \wedge \exists d_{src} \in D_{src} : d_{dup} \text{ is a true positive detection of } d_{src} \} \quad (6)$$

$$D'_{ret} = \{ d_{src} \vee d_{src} \in D_{src} \wedge \exists d_{ret} \in D_{ret} : d_{ret} \text{ is a true positive detection of } d_{src} \} \quad (7)$$

Presisi mengukur seberapa tepat dan relevan dokumen yang diidentifikasi sebagai dokumen sumber dibagi dengan jumlah total dokumen yang terambil sepert dinyatakan dalam Persamaan 8. Recall mengukur berapa banyak dokumen relevan yang terambil oleh

sistem seperti dinyatkaan dalam Persamaan 9. Nilai F1 merupakan nilai rata-rata seimbang antara nilai presisi dan recall dan dinyatakan dalam Persamaan 10.

$$Precision = \frac{|D_{ret} \cap D'_{src}|}{|D_{ret}|} \quad (8)$$

$$Recall = \frac{|D'_{ret} \cap D_{src}|}{|D_{src}|} \quad (9)$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \quad (10)$$

III. HASIL DAN PEMBAHASAN

A. Pemeriksaan dokumen yang duplikat

Dalam korpus yang sudah dibentuk menjadi txt, terdapat beberapa teks yang memiliki kemiripan sangat tinggi. Untuk dapat menghitung presisi dan recall seperti pada [4], perlu diketahui dokumen yang memiliki duplikasi dalam korpus. *Jaccard similarity score* digunakan untuk memeriksa apakah korpus tersebut memiliki duplikasi atau tidak. Jika skor yang dihasilkan melebihi 0,9, maka kedua teks tersebut dianggap sebagai duplikat. Nilai *Jaccard similarity* dapat diperoleh dengan menggunakan Persamaan 11.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \quad (11)$$

Hasil dari proses pengecekan duplikat disimpan dalam *file* berformat csv. Data yang disimpan dalam file ini adalah pasangan ID dokumen yang dinyatakan duplikat karena tingkat kemiripannya yang tinggi. Format csv ini berbeda dengan format anotasi dokumen duplikat yang dilakukan dalam [2], [4], [6] yang menggunakan format xml. Dalam kajian tersebut, luaran proses temu kembali dokumen sumber juga dinyatakan dalam format dokumen xml. Contoh bentuk penyimpanan dalam penelitian ini dinyatakan dalam Tabel 2.

B. Pembangunan korpus

Data yang diujikan didapatkan dari pengolahan oleh manusia. Sumber korpus yang digunakan sebagai referensi data uji berasal dari korpus yang digunakan dalam [6]. Data uji yang digunakan terdiri atas dua jenis, yaitu dokumen buatan yang digunakan pada [2] dan dokumen tersimulasi

Dokumen buatan/artifisial berasal dari kumpulan dokumen sumber yang dipilih dan diberlakukan tiga operasi teks secara acak, yaitu penghapusan, pengacakan, dan penyisipan. Penghapusan atau *deletion* melakukan operasi penghapusan sebanyak 30, 50, dan 60 persen dari *token* berdasarkan panjang *window* atau dokumen. Pengacakan atau *shuffle* melakukan operasi pengacakan posisi *token* sebanyak satu hingga dua iterasi berdasarkan panjang *window* atau dokumen. Penyisipan atau *insertion* melakukan operasi untuk menyisipkan kata yang berasal dari leksikon sebanyak 10 hingga 100 persen dari *token* berdasarkan panjang *window* atau dokumen.

Tabel 2. Penyimpanan dokumen sumber dan duplikat

Sumber	Dok_duplikat	# kata
1.txt	['ARS0004.txt']	671
10.txt	['ARS0001.txt']	5.008
100.txt	[]	3.054

Tabel 3. Data statistik D_{plg}

Jenis dokumen	Jenis penyamaran	Jumlah	Sumber
Uji artifisial	Penghapusan	6	[3]
	Penyisipan	6	
	Pengacakan	6	
	Sinonim	6	
	Hapus + sisipkan	6	
Uji simulasi	Parafrase	20	[2]
	Hibrida	7	Mhs prodi PBII UNY
	Salin dan tempel	3	Penulis D_{corp}
	Salin dan Kocok	3	Penulis D_{corp}

Dokumen artifisial digenerasikan oleh sistem dengan algoritme yang sudah ditentukan [3]. Jumlah dokumen artifisial yang digunakan dalam penelitian ini adalah 30 dokumen. Sinonim kata dasar yang digunakan dalam dokumen artifisial berasal dari WordNet bahasa Indonesia, sedangkan untuk operasi penyisipan digunakan leksikon kata dasar bahasa Indonesia.

Dokumen uji tersimulasi merupakan dokumen hasil tulisan manusia yang dengan sengaja ditulis untuk menghasilkan teks daur ulang/plagiasi. Tujuan dari dokumen tes tersimulasi adalah untuk mendapatkan kasus-kasus asli plagiasi/daur ulang teks yang cukup untuk pengujian. Kasus-kasus asli disimulasikan karena mendapatkan teks daur ulang/plagiasi di ranah ilmiah cukup sulit dan memakan waktu yang jauh lebih banyak.

Sumber dokumen secara detail dinyatakan dalam Tabel 3. Dokumen tambahan merupakan dokumen dengan jenis penyamaran hibrida, salin dan tempel, serta salin dan kocok. Dokumen yang disamarkan dengan jenis hibrida merupakan dokumen uji yang ditulis oleh mahasiswa jurusan Pendidikan Bahasa Inggris dan Indonesia Universitas Negeri Yogyakarta. Dokumen salin dan tempel serta salin dan kocok merupakan dokumen yang dibuat oleh penulis dengan memodifikasi menggabungkan beberapa D_{corp}.

C. Pengujian

Dokumen artifisial diuji dengan menggunakan sebelas skenario pengujian berbeda untuk mendapatkan nilai optimal. Pengujian terbagi menjadi dua jenis, yaitu dengan penyaringan dan tanpa penyaringan. Pengujian tanpa penyaringan langsung menghitung nilai presisi, recall, dan F1 dari hasil pencarian. Pengujian dengan

penyaringan mengambil satu hingga sepuluh dokumen teratas berdasarkan hasil pencarian.

Gambar 5 menunjukkan nilai presisi dan recall serta rata-rata nilainya dari dokumen uji artifisial. Dari ketiga dokumen, nilai presisi paling tinggi didapatkan pada pengujian dengan jumlah filter dokumen sama dengan 1. Nilai presisi didapatkan dengan pembagian hasil irisan dengan D_{ret}. D_{ret} merupakan jumlah dokumen yang didapatkan dari proses oleh sistem. Nilai presisi berbanding terbalik dengan jumlah D_{ret}, sehingga semakin sedikit D_{ret} yang diperoleh, maka nilai presisi meningkat. Nilai rerata makro recall relatif sama, yakni 0,97.

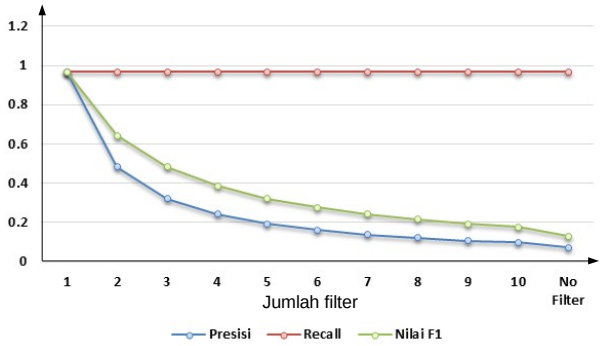
Dalam modul *source retrieval* pada deteksi daur ulang teks, nilai recall lebih diutamakan dibanding presisi. Recall diutamakan karena nilai ini menunjukkan seberapa banyak dokumen kandidat yang berhasil diambil oleh sistem. Dokumen uji yang sudah terambil diproses lebih lanjut dalam proses *text alignment* yang berada di luar lingkup penelitian ini.

Tabel 4 menampilkan hasil rata-rata pengujian ke dokumen artifisial dilihat dari penyamaran proses daur ulang. Hasil rata-rata makro nilai recall tidak berbeda kecuali pada penyamaran bentuk sinonim. Pada penyamaran teks lain, kata kunci yang digunakan oleh *query* pencarian masih ada dalam dokumen walaupun posisi dan jumlahnya dapat berubah. Rata-rata makro terendah didapatkan pada kasus penyamaran teks dengan sinonim kata. Hal ini disebabkan karena kata kunci yang seharusnya dapat diperoleh dengan menggunakan algoritme pada bobot kata lokal menjadi berbeda dari dokumen sumber.

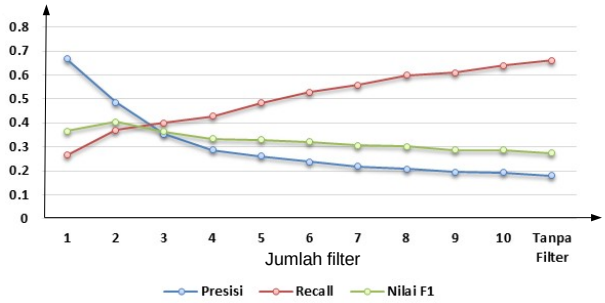
Sama seperti pada pengujian dokumen artifisial, dokumen tes tersimulasi juga diuji dengan menggunakan sebelas skenario pengujian. Pengujian yang digunakan adalah tanpa penyaringan dan penyaringan sebanyak satu hingga sepuluh dokumen. Gambar 6 menampilkan hasil pengujian dari dokumen tes tersimulasi dengan skenario tersebut. Hal ini menunjukkan peningkatan nilai recall saat tidak diberlakukan penyaringan (*filtering*). Selaras dengan [3], nilai recall dari dokumen uji tersimulasi cenderung lebih rendah dari nilai recall dokumen uji artifisial. Ini menunjukkan bahwa penyamaran algoritmik lebih mudah dideteksi daripada penyamaran yang ditulis oleh penulis manusia.

Dalam Gambar 6, semakin sedikit dokumen yang diambil oleh sistem (*retrieved*), maka semakin tinggi pula nilai presisi yang didapatkan. Hal ini sama seperti pada dokumen artifisial dengan menggunakan Persamaan 5. Persamaan tersebut menyatakan nilai presisi berbanding terbalik dengan D_{ret} sehingga nilai presisi meningkat seiring menurunnya jumlah D_{ret}. Kondisi ini menjadi hokum dalam sistem temu kembali seperti yang diulas dalam [13].

Nilai recall diperoleh berdasarkan tiga jenis pengujian. Berbeda dengan presisi, nilai recall meningkat seiring bertambahnya jumlah D_{ret}. Peningkatan pada jumlah D_{ret} menyebabkan irisan yang diperoleh bertambah sehingga meningkatkan nilai recall. Dari ketiga pengujian yang sudah dilakukan,



Gambar 5. Hasil pengujian dokumen artifisial



Gambar 6. Hasil pengujian dokumen tes tersimulasi

Tabel 4. Nilai evaluasi dokumen artifisial

Jenis penyamaran		Pres.	Recall	F1	Rerata Recall
Penghapusan	Ringan	0,053	1	0,100	1
	Sedang	0,077	1	0,143	
	Berat	0,087	1	0,159	
Penyisipan	Ringan	0,043	1	0,083	1
	Sedang	0,040	1	0,077	
	Berat	0,076	1	0,140	
Pengacakan	Sedang	0,074	1	0,137	1
Sinonim	Ringan	0,091	1	0,167	0,889
	Sedang	0,056	0,667	0,102	
	Berat	0,056	1	0,105	
Hapus dan sisipkan	Ringan	0,048	1	0,091	1
	Sedang	0,072	1	0,135	
	Berat	0,072	1	0,134	

pengujian tanpa menggunakan filter menghasilkan nilai recall yang lebih tinggi dibandingkan dengan pengujian lainnya. Oleh karena itu, pemilihan kandidat sumber tanpa penyaringan lanjutan lebih disarankan untuk digunakan dalam komponen *source retrieval*.

Tabel 5 menunjukkan hasil evaluasi dari dokumen uji yang tersusun berdasarkan jenis penyamaran dokumen. Dokumen dengan penyamaran jenis parafrase terdiri atas parafrase ringan, sedang, dan berat. Semakin berat tingkat parafrase yang digunakan, maka recall yang didapatkan juga ikut menurun. Nilai presisi, recall, dan F1 ketiga tingkat parafrase diambil dari tiga dokumen dengan nilai recall tertinggi. Pengambilan ini didasari oleh jumlah dokumen uji yang tidak seimbang, yaitu terdapat sembilan dokumen parafrase ringan,

delapan dokumen parafrase sedang, dan tiga dokumen parafrase berat. Jumlah yang tidak seimbang menyebabkan nilai rata-rata recall dari keseluruhan dokumen berdasarkan tingkat mnjadi lebih tinggi pada kasus parafrase berat.

Apabila diurutkan, nilai rata-rata recall berdasarkan jenis penyamaran dari teringgi ke terendah adalah salin dan temple (*copy & paste*), parafrase, salin dan kocok (*copy & shake*), dan hibrid. Dokumen tes salin dan tempel serta salin dan kocok berasal dari dokumen sumber yang sama. Hanya saja pada dokumen tes salin dan kocok, urutan kalimat dalam dokumen diacak sehingga tidak memiliki urutan yang sama dengan dokumen salin dan tempel. Proses pengacakan ini menyebabkan nilai lokal yang diperoleh untuk mendapatkan *query* dapat meningkat atau menurun. Oleh karena itu, nilai rata-rata recall yang didapatkan dari dokumen tes salin dan kocok lebih rendah dibandingkan salin dan tempel.

Dokumen tes hibrida adalah dokumen tes tersimulasi yang menggabungkan kasus penyamaran teks yang terdiri atas parafrase dan ringkasan. Pada kasus salin dan tempel, salin dan kocok, serta parafrase, kata kunci pada dokumen tes masih dapat ditemukan dengan algoritme yang digunakan. Namun, pada dokumen tes hibrida, kata-kata kunci sulit ditemukan karena sudah mengalami banyak penyamaran. Penyamaran dalam bentuk ringkasan pada dokumen tes hibrida menyebabkan kata kunci yang digunakan sebagai *query* tidak dapat ditemukan dalam dokumen kandidat sumber.

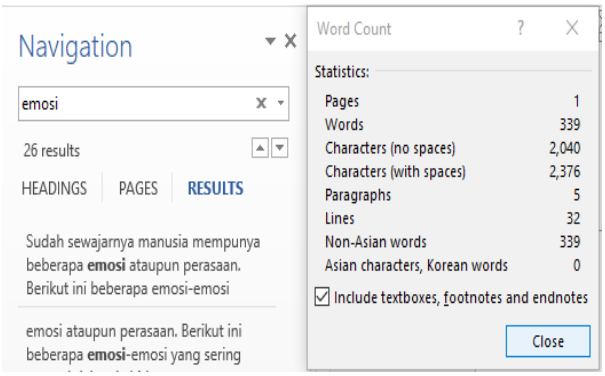
Skenario evaluasi dan hasil yang dipaparkan di atas menunjukkan variasi pengujian dan granularitas pengujian yang jauh lebih detil dan rumit jika dibandingkan dengan kajian [9]-[11]. Suryana dkk. [9] melakukan pengujian berdasarkan tiga skenario berdasarkan panjang dokumen dan waktu pemrosesan. Kajian dalam [10] tidak menampilkan hasil pengujian secara riil karena tidak dijumpainya sebuah tabel dan menggunakan analisis kualitatif. Alfikri dan Purwarianti [11] melakukan pengukuran nilai akurasi dengan 5 skenario yang didasarkan pada nilai *mutual information* yang berbeda-beda.

Dalam proses evaluasi, terdapat beberapa dokumen yang tidak dapat ditemukan kandidat sumbernya. Kasus ini terjadi misalnya pada testdoc019 dalam Gambar 7. Pada dokumen tersebut, kata “emosi” muncul sebanyak 26 kali dari 339 kata yang belum mengalami prapemrosesan. Dengan algoritme pemilihan kata lokal untuk mendapatkan *query* pencarian, kata “emosi” dianggap sebagai kata kunci dalam dokumen tersebut. Hal ini menyebabkan dokumen sumber tidak termasuk dalam kumpulan dokumen yang diambil oleh sistem.

Contoh lainnya terdapat pada dokumen yang banyak menggunakan sinonim kata. Mengacu pada Tabel 4, sinonim memiliki nilai *recall* terendah dibanding penyamaran dokumen lainnya. Kasus yang sama terjadi pada dokumen uji tambahan dalam Tabel 5. Penulis dokumen uji hibrid merupakan mahasiswa jurusan pendidikan yang sudah lama berkecimpung dalam dunia penulisan. Dalam kasus ini, penulis lebih banyak menggunakan parafrase berat dan ringkasan dibanding kasus salin dan kocok. Hal ini menyebabkan kata kunci yang seharusnya digunakan sebagai *query* pencarian tidak dapat ditemukan dengan mudah.

Tabel 5. Nilai evaluasi dokumen tes tersimulasi

Jenis Penyamaran		Pres.	Recall	F1 Score	Rerata Recall
Parafrase	Ringan	0,265	1	0,418	0,880
	Sedang	0,185	0,917	0,261	
	Berat	0,217	0,722	0,324	
Salin dan tempel		0,222	1	0,360	1
Salin dan kocok		0,189	0,778	0,302	0,778
Hibrida		0,060	0,143	0,083	0,143



Gambar 7. Hasil pengamatan testdoc019

IV. KESIMPULAN

Sistem telah berhasil dibangun dan penyamaran dalam bentuk parafrase, *copy and paste*, serta *copy and shake* dapat diatasi dengan menggunakan metode pemilihan kata penting. Nilai presisi dan recall yang memuaskan dapat diperoleh dengan menggunakan penapisan lanjutan sebanyak lima dokumen pada dokumen uji artifisial, sedangkan pada dokumen tes tersimulasi nilai presisi yang memuaskan didapat dari filtering lanjutan lima dokumen dan nilai recall dengan proses tanpa filtering lanjutan. Nilai presisi dan recall terbaik dokumen artifisial didapatkan dari pengujian dengan filtering sebanyak satu dokumen dengan nilai presisi 0,967 dan recall 0,967. Nilai recall 0,66 didapatkan dari dokumen tes tersimulasi dengan pengujian tanpa filtering.

Pengujian lebih lanjut diperlukan untuk data artifisial dokumen dengan menggunakan sinonim. Selain itu, diperlukan pula pengujian lebih lanjut untuk data dokumen tes tersimulasi dengan penyamaran bentuk ringkasan dan pengujian dengan mempertimbangkan semantik dari *query* pencarian.

DAFTAR PUSTAKA

[1] P. Clough, R. Gaizauskas, S. Piao and Y. Wilks, "METER: MEasuring TExt Reuse," in *4th Annual*

Meeting of the Association for Computational Linguistics, Stroudsburg, United States, Jul. 2002, pp. 152-159. doi: 10.3115/1073083.1073110

[2] L. D. Krisnawati and K. U. Schülz, "Significant word-based text alignment for text reuse detection," in *International Conference on Research and Innovation in Computer, Electronics, and MAnufacturing Engineering*, Bali, Indonesia, Feb. 2017, pp. 7-12.

[3] L. D. 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, Nov. 2017, pp. 27-44.

[4] M. Potthast et al., "Overview of the 6th international competition on plagiarism detection," in *CLEF 2014 Evaluation Labs and Workshop*, Sheffield, UK, Sept. 2014, pp. 845-876.

[5] L. D. Krisnawati and K. Schülz, "Plagiarism detection for Indonesian texts," in *International Conference on Information Integration and Web-based Applications & Services*, Vienna, Austria, Dec. 2013, pp. 595-599. doi: 10.1145/2539150.2539213

[6] L. D. Krisnawati, "Plagiarism detection for Indonesian texts," *Dissertation*, Ludwig Maximilian University, Munich, Germany, 2016.

[7] K. Leilei, L. Zhimao, Y. Yong, Q. Haoliang, and H. Zhongyuan, "Source retrieval and text alignment corpus," in *CLEF 2015 Conference and Labs of the Evaluation Forum*, Toulouse, France, Sept. 2015, pp. 1-7.

[8] B. Gipp, *Citation-based plagarism detection: detecting disguised and cross-language plagiarism using citation pattern analysis*. Wiesbaden: Springer, 2014. doi: 10.1007/978-3-658-06394-8

[9] A. F. Suryana, A. T. Wibowo, and A. Romadhany, "Performance efficiency in plagiarism indication detection system using indexing method with data structure 2–3 tree," in *2nd International Conference on Information and Communication Technology*, Bandung, Indonesia, May 2014, pp. 403-408. doi: 10.1109/ICoICT.2014.6914096

[10] A. R. Syahputra, "Implementasi algoritma winnowing untuk deteksi kemiripan," *Pelita Informatika Budi Dharma*, vol. 9, no. 1, pp. 134-138, 2015.

[11] Z. F. Alfikri and A. Purwarianti, "Detailed analysis of extrinsic plagiarism detection system using machine learning approach (naive bayes and svm)," *TELKOMNIKA Indonesian Journal of Electrical Engineering*, vol. 12, no. 11, pp. 7884-7894, 2014. doi: 10.11591/telkomnika.v12i11.6652

[12] N. Kurniati, A. Rahmatulloh, and R. Qomar, "Web scraping and winnowing algorithms for plagiarism detection of final project titles," *Lontar Komputer: Jurnal Ilmiah Teknologi Informasi*, vol. 10, no. 2, pp. 73-83, 2019. doi: 10.24843/LKJITI.2019.v10.i02.p02

[13] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to information retrieval*. New York: Cambridge University Press., 2009.

[14] M. Kiabod, M. N. Dehkordi, and M. Sharafi, "A novel method of significant words identification in text summarization," *Journal of Emerging Technologies in Web Intelligence*, vol. 4, no. 3, pp. 252-258, 2012. doi: 10.4304/jetwi.4.3.252-258

[15] C. Basile, D. Benedetto, E. Caglioti, G. Cristadoro, and M. D. Esposti, "A plagiarism detection procedure in three steps: selection, matches and squares," in *3rd Workshop on Uncovering Plagiarism, Authorship and Social Software Misuse*, San Sebastian, Spain, Sept. 2009, pp. 1-9.