KLASIFIKASI ONLINE DOKUMEN BERITA DENGAN MENGGUNAKAN ALGORITMA SUFFIX TREE CLUSTERING

Agus Zainal Arifin [1)], **Roby Darwanto** [2)], **Dini Adni Navastara** [3)], **Henning Titi Ciptaningtyas** [4)]
[1] Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
[2]Gedung Teknik Informatika, Kampus ITS, Keputih, Sukolilo, Surabaya, 60111
Telp : (031) 593 9214, Fax : (031) 599 6813
E-mail : agus.za@its-sby.edu[1)], robin_ftif@cs.its.ac.id[2)], dini_navastara@if.its.ac.id[3)], henning@its-sby.edu[4)]

Abstract

Together with fast progressively information technology growth now here cause the number spreading of information exploiting internet. The number of document in the virtual world keeps on increasing up to billions of documents. Therefore, the document clustering is very required to search information easily a certain news topic or occurrence.

This paper shall implement suffix tree clustering algorithm for news document clustering which enable the existence of sharing topic between document. This algorithm have three steps: document cleaning for clean up HTML tags, stoplist removing, and stemming process, Identifying base clusters as suffix tree constructed phase to get base cluster, and the last step is combining base clusters to reduce the number of base clusters by merger between base clusters so that will yield a new clusters.

Based on the experiment, suffix tree clustering algorithm proved that have very high precision mean level. It indicated that by using phrase as base clustering will have high precision level.

Keywords: *suffix tree, base cluster, phrase, sharing topic*

Abstrak

Seiring dengan semakin pesatnya perkembangan teknologi informasi sekarang ini menyebabkan banyaknya penyebaran informasi yang memanfaatkan media internet. Jumlah dokumen yang beredar di dunia maya ini pun semakin bertambah hingga mencapai milyaran dokumen. Oleh karena itu, pengelompokan dokumen sangat dibutuhkan untuk mempermudah pencarian informasi mengenai suatu kejadian atau topik berita tertentu.

Paper ini akan mengimplementasikan algoritma suffix tree clustering untuk pengelompokan dokumen berita yang memperbolehkan adanya sharing topik berita antar dokumen. Algoritma ini memiliki 3 tahapan yaitu: tahap pembersihan dokumen yang meliputi pembersihkan tag-tag HTML, penghilangan stoplist, dan proses stemming, tahap identifikasi base cluster merupakan tahap pembentukan suffix tree untuk menemukan base cluster, dan yang terakhir adalah tahap kombinasi base cluster untuk mereduksi jumlah base cluster dengan cara melakukan merger antar base cluster sehingga akan menghasilkan suatu cluster baru.

Berdasarkan percobaan, algoritma suffix tree clustering terbukti memiliki tingkat rata-rata precision yang sangat tinggi. Dimana data hasil uji coba menunjukkan bahwa dengan menggunakan frase sebagai dasar pembentukan cluster akan mempunyai tingkat ketepatan yang sangat tinggi.

Keywords *: suffix tree, base cluster, frase, sharing topik*

1. PENDAHULUAN

Dengan semakin berkembangnya Teknologi Informasi yang dibutuhkan oleh pengguna mengakibatkan munculnya suatu ilmu baru dalam Teknologi Informasi yaitu Sistem Informasi Temu Kembali (*Information Retrieval*) yang mempelajari cara-cara temu kembali dan penelusuran dokumen. Temu kembali informasi berbasis *query* (*query based retrieval*) yang digunakan secara tradisional sangat berguna untuk temu kembali terarah tetapi tidak efisien untuk temu kembali yang melibatkan data dalam jumlah yang besar. *Query Based Retrieval* atau temu kembali dengan menggunakan *query* sederhana berguna ketika kita mengetahui benar kejadian asli atau fakta yang dicari dalam kategori kecil. Cara ini tidak begitu efisien ketika

kita membutuhkan informasi yang spesifik/khusus dalam kategori yang besar.

Topik (*Topic*) adalah perubahan kejadian secara dinamis. Dalam metode ini kita menggunakan beberapa cara untuk temu kembali informasi (*Information Retrieval*) dan teknik *machine learning* untuk keefektifan pendeteksian dan klasifikasi (*detection*) [Yiming][J.Allan]. Penerapan algoritma *Suffix tree Clustering* didasarkan karena satu dokumen berita dapat memiliki lebih dari satu topik. Algoritma ini memiliki tingkat *precision* yang sangat tinggi. Hal ini dikarenakan dalam algortima ini menggunakan *phrase* sebagai dasar pembentukan *cluster*.

Oleh karena itu untuk mendapatkan berita–berita yang diinginkan oleh pengguna dengan cepat dan mudah diperlukan suatu sistem yang cerdas (*Intelligent System*) yaitu sistem yang secara otomatis dapat :

- Mendeteksi dan mengelompokan berita menjadi kelompok berita yang berkaitan.
- Memasukan berita-berita kejadian baru pada kelompok berita yang sudah ada.
- Membuat kelompok berita baru apabila suatu berita tidak memenuhi kelompok berita yang sudah ada.
- Menampilkan kejadian yang diperlukan oleh user dalam bentuk berita-berita terkait yang ada hubungannya dengan keinginan user.

Dari penjelasan diatas maka diperlukan suatu sistem temu kembali informasi untuk pendeteksian dan pengelompokan berita yang merupakan salah satu alternatif untuk menangani masalah tersebut. Dengan menggunakan algoritma-algoritma yang dikembangkan untuk menangani masalah-masalah temu kembali atau penelusuran informasi seperti *Suffix tree Clustering*, tingkat *precision* pencarian beritanya bisa ditingkatkan dan bias informasi yang terjadi bisa diminimalisasi.

2. DASAR TEORI

2.1 *Suffix tree Clustering*

Inti dari suatu hasil pencarian yang menerapkan *clustering* adalah penggunaan algoritma *clustering*. Algoritma *Suffix tree Clustering* (STC) memiliki dua kunci utama, yaitu :

1. Menggunakan *phrase* sebagai dasar pembentukan *cluster*nya.
2. Menggunakan suatu definisi *cluster* sederhana.

Suffix tree Clustering memiliki dua langkah utama. Dalam langkah pertama, pencarian *shared phrase* untuk semua dokumen berita yang dikoleksi. Kita menyebut *shared phrase* sebagai *phrase cluster* atau *base cluster*, yang ditemukan dengan menggunakan suatu struktur data yang dinamakan *suffix tree* [Novan]. Dalam langkah kedua, kita mengkombinasikan *base cluster-base cluster* ke dalam suatu *cluster*. Penggabungan antar dua *base cluster* didasarkan pada jumlah dokumen yang melakukan *overlap* diantara kedua *base cluster* tersebut [Zamir]. Suatu *phrase* yang dimaksud dalam konteks algoritma ini adalah urutan satu atau lebih kata-kata. STC memiliki tiga langkah utama, yaitu :

1. *Cleaning* Dokumen.
2. Identifikasi *Base Cluster* menggunakan *Suffix tree.*
3. Mengkombinasikan *Base Cluster* ke dalam suatu *cluster*.

Beberapa karakteristik yang membuat *Suffix tree Clustering* cocok digunakan untuk pengelompokan dokumen. Pertama adalah men*generate cluster-cluster* untuk pengelompokan dokumen berdasarkan *phrase. Phrase* juga bermanfaat untuk membangun uraian dan keakuratan deskripsi dari *cluster-cluster*. Kedua, tidak tergantung pada model data. Hal itu mengasumsikan hanya dokumen-dokumen dengan topik yang sama yang akan memiliki *shared phrase.* Ketiga, STC memperbolehkan adanya *overlaping cluster*. Hal itu sangat penting untuk menghindari pembatasan bahwa setiap dokumen hanya memiliki satu *cluster* saja, karena sering kita jumpai satu dokumen mempunyai lebih dari satu topik dan dengan begitu terdapat kemiripan yang lebih dari satu kelompok dokumen. Keempat, STC menggunakan definisi *cluster* yang sederhana. Semua dokumen yang berisi salah satu *phrase cluster* akan menjadi anggota dari *cluster* tersebut.

STC menggunakan *phrase* untuk mendeteksi kemiripan antar dokumen. STC menggunakan *suffix tree* untuk mengidentifikasi *phrase*. Fitur yang membuat suksesnya STC sebagai algoritma *clustering* adalah adanya *overlaping cluster*. Kualitas *cluster* yang terbentuk dari algoritma STC ini akan menurun jika tanpa menggunakan multiword *phrase* dan tidak memperbolehkan adanya *overlaping cluster*.

2.2 Document Cleaning

Document Cleaning adalah tahap awal dalam algoritma *Suffix tree Clustering*. Pada tahap ini, dokumen yang telah didapat dari proses download akan dibersihkan dan dipersiapkan untuk tahap selanjutnya. Proses

untuk mempersiapkan dokumen meliputi proses pembersihan dokumen dari tag-tag HTML, proses analisa leksikal teks, proses penghapusan *stopword*, dan proses *stemming*.

2.3 *Stemming* Bahasa Indonesia

Dalam morfologi kata Bahasa Indonesia dikenal adanya 3 imbuhan yaitu awalan (*prefiks*), sisipan, dan akhiran (*sufiks*). Untuk penanganan dokumen yang mengandung kata jadian pada tugas akhir ini hanya akan menghilangkan awalan dan akhiran [William].

Metode ini didahului dengan pembacaan tiap kata dari file sampel. Sehingga input dari algoritma ini adalah sebuah kata yang kemudian dilakukan :

Pemeriksaan semua kemungkinan bentuk kata. Setiap kata dalam Bahasa Indonesia umumnya maksimal memiliki 2 Awalan (*prefiks*) dan 3 Akhiran (*sufiks*).

Sehingga bentuknya menjadi :

P1 + P2 + KD + S3 + S2 + S1

Urutan pemotongan awalan dan akhiran adalah sebagai berikut :

1. Bandingkan isi dokumen dengan kamus kata dasar yang sebelumnya diinisialisasi pada memori
2. Pecah isi dokumen dalam bentuk array sejumlah kata yang terdapat dalam dokumen
3. Ambil semua kata dasar yang terdapat dalam dokumen yang sama dengan kamus simpan dalam bentuk array
4. Reduksi sisa pengambilan kata dasar dalam array baru
5. Lakukan pemotongan imbuhan menurut aturan pemotongan.

Aturan pemotongan dilakukan secara berurutan sebagai berikut :

a. Pemotongan Awalan
b. Pemotongan Akhiran
c. Pemotongan Awalan dan akhiran

Untuk awalan dan akhiran semua gabungan dua awalan atau akhiran dilakukan dalam 1 kali pemotongan seperti awalan “memper”. Awalan ini akan dianggap sebagai 1 awalan sehingga akan dilakukan pemotongan satu kali saja. Bukan 2 kali dengan memotong “mem” dan “per”.

Pada setiap tahap pemotongan di atas diikuti dengan pemeriksaan di kamus apakah hasil pemotongan itu sudah berada dalam bentuk dasar. Kalau pemeriksaan ini berhasil maka proses dinyatakan selesai dan tidak perlu melanjutkan proses pemotongan imbuhan lainnya

Contoh pemenggalan kata “mempermainkannya”

Langkah 1 :
Cek apakah kata ada dalam kamus
Ya : Success
Tidak : lakukan pemotongan Awalan
Kata = mainkannya
Langkah 2 :
Cek apakah kata ada dalam kamus
Ya : Success
Tidak : lakukan pemotongan Akhiran
Kata = mempermain
Langkah 3 :
Cek apakah kata ada dalam kamus
Ya : Success
Tidak : lakukan pemotongan kombinasi awalan dan akhiran
Kata = main
Langkah 4 :
Cek apakah kata ada dalam kamus
Ya : Success
Tidak : lakukan Incremental Noun Group

Pemeriksaan kombinasi ini diperlukan, karena adanya fenomena *overstemming* pada algoritma pemotongan imbuhan. Kelemahan ini berakibat pada pemotongan bagian kata yang sebenarnya adalah milik Kata dasar itu sendiri yang kebetulan mirip dengan salah satu jenis imbuhan yang ada. Dengan kombinasi itu, pemotongan yang sudah terlanjur tersebut dapat dikembalikan sesuai posisinya

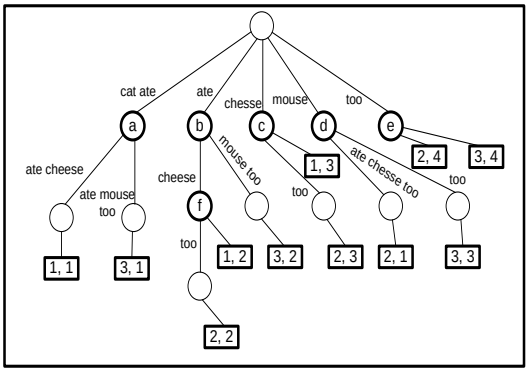
2.4 Identifikasi *Base Cluster*

Tahap identifikasi *base cluster* merupakan tahap terpenting dalam algoritma *suffix tree clustering*, karena pada tahap ini akan menghasilkan *cluster-cluster* dasar [Zamir]. Pembentukkan *base cluster* dilakukan dengan cara menemukan share *phrase* antar dokumen. Untuk menemukan share *phrase* digunakan struktur data *suffix tree*. Dengan menggunakan struktur data ini, maka setiap dokumen akan direpresentasikan menjadi suatu kalimat. Untuk menemukan *base cluster* dapat dilakukan dengan cara membuat suatu *invert index* dari *phrase* untuk semua dokumen.

Contoh pembentukkan *suffix tree* untuk kalimat *cat ate cheese, mouse ate cheese too,* dan *cat ate mouse too* ditunjukkan pada Gambar 1.

Pada Gambar 1 menunjukkan adanya internal node yang terbentuk. Setiap internal node merepresentasikan suatu kelompok dokumen dan share *phrase* untuk kelompok tersebut. Oleh karena itu, setiap internal node juga merepresentasikan *base cluster* yang terbentuk. Semua *base cluster* yang terbentuk

dapat ditunjukkan pada Tabel 1.



Gambar 1. Generate Suffix tree

Tabel 1. Base Cluster yang terbentuk

Base Cluster	***Phrase***	**Documents**
a	cat ate	1, 3
b	ate	1, 2, 3
c	cheese	1, 2
d	mouse	2, 3
e	too	2, 3
f	ate chesse	1, 2

Setiap *base cluster* yang terbentuk memiliki suatu *score*. Penghitungan score merupakan suatu fungsi dari jumlah dokumen yang masuk anggota *base cluster* dan jumlah kata yang menyusun *phrase* dari *base cluster*. Fungsi untuk menghitung *score base cluster* ditunjukkan oleh persamaan (1).

$$s(B) = |B|.f(|P|) \quad (1)$$

dimana pada persamaan (1) :

$|B|$ = jumlah dokumen di dalam *base cluster B* dan $|P|$ = jumlah kata yang menyusun frase *P*.

$f(|P| = 0$, jika $|P| = 1$ dan $f(|P| = 6$, jika $|P| \geq 6$

2.5 Kombinasi *Base Cluster*

Tahap ini digunakan untuk menangani *overlaping cluster*. Dalam tahap ini, *phrase* tidak dipertimbangkan. Sebelum melakukan kombinasi antar *base cluster* [Zamir], kita harus menghitung dulu nilai *similiarity* antar *base cluster* yang didasarkan pada jumlah dokumen yang *overlap*. Adanya *overlaping* dokumen ini didasarkan karena dokumen memiliki lebih dari satu topik sehingga dokumen dapat memiliki lebih dari satu *phrase* yang dishare.

Ukuran nilai *similiarity* menggunakan nilai biner. Rumus untuk menghitung nilai *similiarity* antar *base cluster* ditunjukkan pada persamaan (2) dan (3)

$$|Bm \cap Bn| / |Bm| > 0{,}5 \quad (2)$$

$$|Bm \cap Bn| / |Bn| > 0{,}5 \quad (3)$$

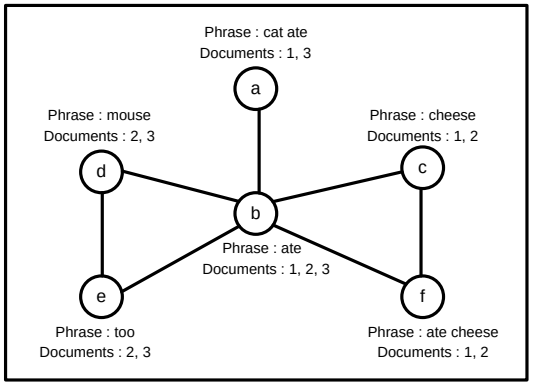
dimana pada persamaan (2) dan (3) :

$|Bm \cap Bn|$ = jumlah dokumen yang overlap terhadap *base cluster Bm* dan *Bn*.

$|Bm|$ dan $|Bn|$ = jumlah dokumen dalam *base cluster Bm* dan *Bn*.

Dalam persamaan diatas, menunjukkan penggunaan nilai *threshold* 0,5 karena nilai tersebut merupakan nilai tengah antara 0 sampai 1. Jika persamaan (2) dan (3) bernilai benar maka *similiarity* akan bernilai 1 sehingga antara kedua *base cluster* tersebut akan terhubung. Jika salah satu dari persamaan (2) dan (3) bernilai benar atau keduanya bernilai salah maka *similiarity* akan bernilai 0 sehingga antara kedua *base cluster* tersebut tidak terhubung.

Keterhubungan antar *base cluster* dapat dicrkan dalam bentuk *base cluster graph* yang ditunjukkan pada Gambar 2.



Gambar 2. Base Cluster Graph

Dari Gambar 3 menunjukkan bahwa antar *base cluster* terhubung sehingga dari 6 *base cluster* tersebut akan membentuk satu *cluster* tunggal.

Untuk pembentukkan *cluster* digunakan algoritma *single link* dimana nilai *similiarity* minimal antar *base cluster* sebagai kriteria berhenti. Kita menggunakan algoritma ini dalam *domain base cluster* dimana kita hanya

menemukan keterhubungan antara *base cluster* yang terkecil.

3. HASIL DAN EVALUASI

3.1 Uji Coba Fungsionalitas

Pada uji coba fungsionalitas akan dilakukan percobaan untuk melihat apakah fungsi-fungsi utama dari perangkat lunak ini dapat berjalan dengan benar sebagaimana mestinya. Fungsi-fungsi utama dari perangkat lunak meliputi fungsi untuk download data, fungsi untuk *clustering*, dan fungsi untuk mencari berita.

3.1.2 Uji coba Fungsi Download Data

Fungsi download data merupakan bagian awal dalam program, fungsi ini sangat penting karena hasil dari fungsi ini akan digunakan untuk proses-proses selanjutnya. Pada fungsi ini akan dilakukan download dokumen berita yang ingin dikelompokkan. Dokumen berita yang didownload besumber dari beberapa situs berita online. Pada fungsi ini akan diuji proses koneksi ke internet, proses pengaksesan proxy, dan proses download data.

3.1.2 Uji Coba Fungsi *Clustering*

Fungsi *Clustering* merupakan fungsi utama dalam aplikasi ini, karena pada fungsi ini akan dilakukan pengelompokkan dokumen berita sehingga akan mempermudah dalam pencarian berita. Pada fungsi ini menerapkan algoritma *suffix tree clustering*, dimana pada algoritma tersebut memiliki 3 tahap utama. Pada fungsi ini akan menguji 3 tahap tersebut apakah berjalan sebagaimana mestinya. Pengujian ini meliputi tahap praproses, tahap pembentukkan *suffix tree*, dan tahap penghitungan *similiarity*.

1. Tahap Praproses

Pada tahap ini akan diuji 3 proses utama yang meliputi proses analisa leksikal teks, proses penghapusan *stopword*, dan proses *stemming*.

a. Pengujian Analisa Leksikal Teks

Pada pengujian ini, akan diberikan inputan suatu kalimat yang didalamnya terdapat tanda baca dan angka. Hasil yang benar dari pengujian ini adalah tampilan suatu kalimat yang tersusun dari kata-kata tanpa adanya tanda baca dan angka. Hasil pengujian ini ditunjukkan pada Tabel 2.

Tabel 2 Hasil Pengujian Leksikal Teks

Status	Kalimat
Input	Karakter :@?!&; akan terhapus.
Output	Karakter akan terhapus

Pada Tabel 2 menunjukkan bahwa hasil dari pengujian ini akan menghapus karakter dan angka. Karakter dan angka yang terhapus pada Tabel 2 dapat ditunjukkan pada Tabel 3.

Tabel 3 Karakter dan Angka yang terhapus

Karakter	Status	Angka	Status
:	Terhapus	**1**	Terhapus
@	Terhapus	**2**	Terhapus
?	Terhapus	**3**	Terhapus
!	Terhapus	**4**	Terhapus
&	Terhapus	**5**	Terhapus
;	Terhapus	**6**	Terhapus
.	Terhapus	**7**	Terhapus

Tabel 4 Hasil Pengujian Penghapusan Stopword

Status	Kalimat
Input	Kalimat ini diuji untuk penghapusan dan juga itu
Output	Kalimat diuji penghapusan

Tabel 5 Kata-kata Stopword yang terhapus

Kata	Status
Ini	Terhapus
Untuk	Terhapus
Dan	Terhapus
Juga	Terhapus
Itu	Terhapus

Tabel 6 Karakter HTML yang dihapus

Kata	Pengganti Karakter
&amp;	&
&cent;	¢
&copy;	©
&gt;	>
&iexcl;	¡
&lt;	<
	[spasi]

b. Pengujian Penghapusan *Stopword*

Pada pengujian ini, akan diberikan inputan suatu kalimat yang didalamnya terdapat kata-kata yang merupakan *stopword*. Hasil yang benar dari pengujian ini adalah tampilan suatu kalimat yang tersusun dari kata-kata yang bukan

stopword. Hasil pengujian ini dapat ditunjukkan pada Tabel 4.

Pada Tabel 4 menunjukkan bahwa hasil dari pengujian ini akan menghapus kata-kata yang termasuk *stopword*. Kata-kata yang termasuk *stopword* dan terhapus pada Tabel 4 dapat ditunjukkan pada Tabel 5.

Pada proses ini juga akan dilakukan penghapusan terhadap kata-kata yang menggantikan karakter HTML. Daftar kata yang menggantikan karakter HTML ditunjukkan pada Tabel 6.

c. Proses Stemming

Hampir setiap kalimat dalam bahasa indonesia tersusun dari kata-kata yang berimbuhan. Untuk mengembalikan kata-kata tersebut ke bentuk dasarnya diperlukan suatu proses, proses inilah yang dinamakan *stemming*. Pada pengujian ini, akan diberikan inputan suatu kalimat yang tersusun dari kata-kata berimbuhan. Hasil yang benar dari pengujian ini adalah kata-kata berimbuhan dalam kalimat tersebut akan berubah ke bentuk dasarnya. Hasil pengujian ini ditunjukkan pada Tabel 7.

Pada Tabel 7 menunjukkan bahwa kata-kata yang menyusun kalimat dikembalikan ke bentuk dasarnya. Contoh hasil *stemming* kata ditunjukkan pada Tabel 8.

Tabel 7 Hasil Pengujian Proses Stemming

Status	Kalimat
Input	Kalimat digunakan sebagai ukuran pemakaian kata yang terurut
Output	Kalimat guna bagai ukur pakai kata urut

Tabel 8 Contoh Hasil Stemming Kata

Kata Berimbuhan	Kata Dasar
digunakan	guna
sebagai	bagai
pemakaian	pakai
ukuran	ukur
terurut	urut

2. Tahap Pembentukkan Suffix tree

Pada tahap ini akan dilakukan pengujian terhadap hasil pembentukan *suffix tree*. Pengujian dilakukan dengan memberikan inputan beberapa kalimat. Hasil pengujian dapat dilihat dari *base cluster* yang terbentuk. Tabel 9 menunjukkan hasil uji coba pembentukkan *suffix tree*. Beberapa kalimat yang digunakan sebagai uji coba pembentukkan *suffix tree* antara lain:

1. Kucing makan keju
2. Kucing makan tikus juga
3. Tikus makan keju juga
4. Tikus makan ikan mati
5. Kucing makan ikan mati juga
6. Kucing bermain bola

Tabel 9 Hasil Pengujian Generate Suffix tree

Base Cluster	Share *Phrase*	Score	Parent
1	kucing	4	-
2	makan	5	-
3	keju	2	-
4	kucing makan	6	1
5	tikus	3	-
6	makan keju	4	2
7	tikus makan	4	5
8	ikan	2	-
9	mati	2	-
10	makan ikan	4	2
11	makan ikan	6	2
12	ikan mati	4	8
13	main	1	-
14	bola	1	-

Pada Tabel 9 menunjukkan *base cluster* yang terbentuk, *share phrase*, *score*, dan *parent*nya. *Share phrase* yang dimaksud adalah *phrase* yang mendeskripsikan *base cluster* dan *score* yang dimaksud adalah *score* dari *base cluster* tersebut.

3. Tahap Penghitungan Similiarity

Pada tahap ini akan dilakukan pengujian terhadap hasil penghitungan *similiarity* antar *base cluster*. Pengujian juga dilakukan terhadap pembentukkan *cluster*. Pengujian dilakukan dengan data *base cluster* yang didapat dari Tabel 9.

Tabel 10 Cluster yang terbentuk dari hasil uji coba

Cluster	*Base Cluster*	Score
1	1; 4; 2; 5; 7;	22
2	3; 6	6
3	8; 11; 12; 10; 9;	18
4	13; 14;	2

Pada Tabel 10 menunjukkan bahwa dari data *base cluster* yang terbentuk pada Tabel 9 dari 14

base cluster yang terbentuk akan di-*merger* menjadi 4 *cluster*.

3.1.3 Uji Coba Fungsi Pencarian Berita

Pada uji coba ini akan ditampilkan daftar berita yang berhasil ditemukan berdasarkan kata kunci yang telah dimasukkan oleh user sebagai parameter pencarian. Daftar berita yang ditampilkan terdapat dalam *cluster* yang ditemukan. Satu berita dapat menjadi anggota dari beberapa *cluster*. Hanya 3 *cluster* teratas yang ditemukan yang akan ditampilkan dan diurutkan berdasarkan *score*nya.

3.2 Uji Coba Hasil *Clustering*

Pada uji coba hasil *clustering* akan dilakukan pengujian hasil proses *clustering* dengan menggunakan analisa kuantitatif. Umumnya untuk analisa kuantitatif digunakan penghitungan *precision* dan *recall.*

Untuk uji coba hasil *clustering* dapat ditunjukkan pada Tabel 11.

Tabel 11 Uji coba Clustering dengan threshold 0,5

Jml Data	k	Jml *Base Cluster*	Jml *Cluster*	Macro Average
200	500	7058	17	80%
300	500	11208	28	81%
400	500	14732	34	83%
500	500	18653	52	81%
600	500	21045	58	80%
700	500	23167	65	81%
800	500	26491	71	83%
900	500	29146	76	84%
1100	500	32820	82	82%
1200	500	35179	87	84%

3.3 Uji Coba Waktu

Pada skenario uji coba ini akan dilakukan pengujian terhadap waktu yang dibutuhkan aplikasi untuk melakukan proses-proses utama yang meliputi proses untuk men*generate suffix tree* dan proses pencarian berita.

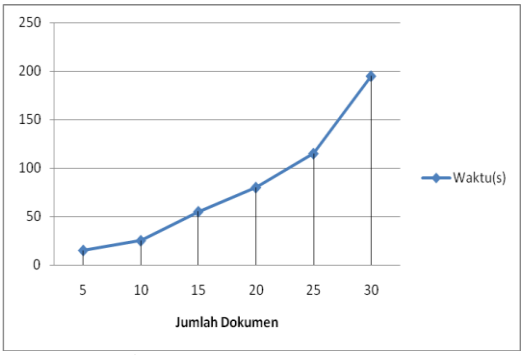
3.3.1 Uji coba Waktu *Generate Suffix tree*

Pada skenario ini akan dilakukan pengujian terhadap waktu yang dibutuhkan untuk melakukan proses *generate suffix tree*. Pengujian ini didasarkan pada jumlah dokumen yang ingin di*generate suffix tree*nya. Pada pengujian ini kita mengambil contoh 6 kali melakukan percobaan dengan jumlah dokumen yang berbeda. Percobaan ini dapat ditunjukkan pada Tabel 12.

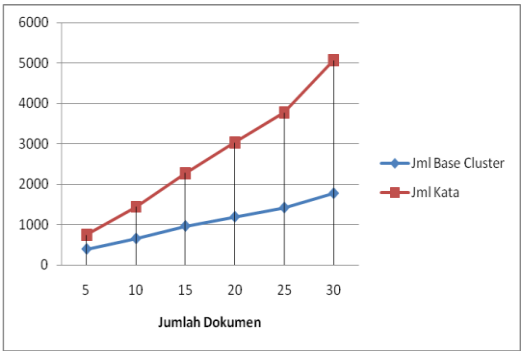
Tabel 12 Hasil Proses Generate Suffix tree

Jml Dok	*Base Cluster*	Jml Kata	Waktu
5	391	739	15
10	652	1434	25
15	960	2270	55
20	1188	3020	80
25	1415	3763	115
30	1769	5065	195

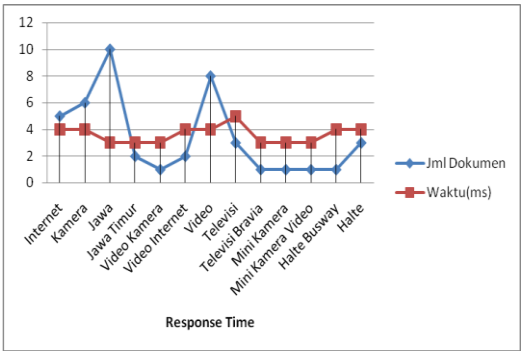
Gambar 3 menunjukkan grafik pengaruh jumlah dokumen terhadap waktu untuk *generate suffix tree* yaitu semakin banyak jumlah dokumen maka semakin lama waktu yang dibutuhkan untuk men-*generate suffix tree*. Sedangkan Gambar 4 menunjukkan grafik pengaruh jumlah dokumen terhadap jumlah *base cluster* dan jumlah kata dimana semakin besar jumlah dokumen maka semakin banyak jumlah kata dan jumlah *base cluster* yang diperoleh.



Gambar 3 Grafik pengaruh jumlah dokumen terhadap waktu untuk generate suffix tree



Gambar 4 Grafik pengaruh jumlah dokumen terhadap jumlah base cluster dan jumlah kata



Gambar 5 Grafik Response time

3.3.2 Uji coba *Response time*

Pada skenario uji coba ini akan dilakukan pengujian waktu yang dibutuhkan untuk merespon *query* sebagai parameter inputan dari user untuk pencarian berita. Pada uji coba ini juga diuji jumlah dokumen yang ditemukan untuk setiap *query* yang diinputkan seperti terlihat pada Tabel 13 dan grafik *response time* dapat dilihat pada Gambar 5.

Tabel 13 Response time

Query	Jml Dokumen	Waktu
Internet	5	4
Kamera	6	4
Jawa	10	3
Jawa Timur	2	3
Video Kamera	1	3
Video Internet	2	4
Video	8	4
Televisi	3	5
Televisi	1	3
Mini Kamera	1	3
Mini Kamera	1	3
Halte Busway	1	4
Halte	3	4

4. KESIMPULAN

Setelah melalui tahap implementasi dan uji coba, maka dapat ditarik kesimpulan sebagai berikut:

1. Algoritma *Suffix tree Clustering* dapat diterapkan untuk *clustering* dokumen berita Berbahasa Indonesia.
2. Untuk melakukan *clustering* dokumen yang didasarkan pada multiword *phrase* digunakan struktur data *suffix tree*.
3. Untuk pembentukkan *suffix tree* membutuhkan waktu yang lama karena selain tergantung pada jumlah dokumen yang dikoleksi juga tergantung pada jumlah kata untuk setiap dokumen yang ingin diklasifikasikan.
4. Algoritma untuk proses *stemming* dalam Bahasa Indonesia perlu dikembangkan lagi untuk mendapatkan kata dasar yang sesuai dengan maksud dari kalimat.
5. Hasil *clustering* algoritma ini memiliki tingkat rata-rata *precision* yang sangat tinggi yaitu 80% untuk jumlah data diatas 200 dengan nilai threshold 0,5 dan nilai k = 500.
6. *Cluster* yang terbentuk dapat dideskripsikan dengan share *phrase*nya.

5. PUSTAKA

Yiming Yang, Jaime G.Carbonell, Rulf D. Brown, Thomas Pierce, Brian T. Achibald, Xin Liu, 1999, ***Learning Approaches for Detecting and Tracking News Events***. IEEE Intelligent Systems, Language Techlonolies Institute, Carbegie Mellon University

J. Allan et al, 1998, ***Topic Detection and Tracking Pilot Study : Final Report*** Proc. DARPA Broadcast News Transcription & Understanding Workshop, Morgan Kaufman, San Francisco, pp194-218

Novan. S, Ari, 2001, ***Implementasi Aplikasi Information Retrieval Untuk Pendeteksian dan Klasifikasi Berita Kejadian Berbahasa Indonesia Berbasis Web***, Tugas Akhir, Jurusan Teknik Informatika Fakultas Teknologi Informasi ITS Surabaya

William B.Frakes, Richardo Baeza-Yates, 1992, ***Information Retrieval Data Structures And Algorithm***, Prentice-Hall International Edition

Zamir, O. And Etzioni, O., 1998, ***Web Document Clustering: A Feasibility Demonstration***. In Proceedings of the 19th International ACM SIGIR Conference on Research and Development of Information Retrieval (SIGIR '98), 46-54

RIWAYAT PENULIS

Agus Zainal Arifin lahir di kota Surabaya pada tanggal 9 Agustus 1972. Penulis menamatkan pendidikan S1 di Teknik Informatika ITS (1995), S2 di Universitas Indonesia (2001), dan S3 di Hiroshima University (2007) dalam bidang Information Engineering. Saat ini bekerja sebagai Dosen di Teknik Informatika ITS.