Indexing dan Searching Document Menggunakan Metode Semantic Suffix Tree Clustering Berbasis Android

David Valentino[1], Adi Wibowo[2], Justinus Andjarwirawan[3]
Program Studi Teknik Informatika
Fakultas Teknologi Industri, UK Petra
Jln. Siwalankerto 121-131 Surabaya 60236
Telp. (031)-2983455, Fax. (031)-8417658
davidvalen95@gmail.com[1], adiw@petra.ac.id[2], justin@petra.ac.id[3]

ABSTRAK

Perangkat smartphone berbasis android sudah menjadi bagian hidup bagi penggunanya di zaman modern seperti saat ini. Pengguna menggunakan smartphone untuk mendukung aktivitas harian seperti diantaranya adalah membaca dan menyimpan dokumen elektronik Pdf, Word dan lain-lain. Pengguna smartphone seringkali lupa terhadap lokasi penyimpanan dokumen elektronik.

Penelitian ini dilakukan untuk membantu pengguna smartphone android mencari dokumen yang mencerminkan keyword pengguna secara semantic atau kata asli di dalam smartphone pengguna. Hasil pencarian dokumen adalah dokumen yang mencerminkan input pengguna secara semantic atau kata asli. Berbagai metode diuji untuk membuat kebutuhan waktu proses cluster menggunakan metode suffix tree hingga pencarian dokumen secara semantik seminimal mungkin

Hasil penelitian menunjukan bahwa pengguna dapat mencari dokumen yang dimaksud di dalam smartphone pengguna. Konsumsi waktu rata-rata untuk proses cluster terhadap 100 dokumen yang mengandung kurang lebih 1000 kata adalah 686.7 detik. Pengguna dapat mencari dokumen yang mencerminkan maksud pengguna setelah proses cluster dilakukan. Konsumsi waktu rata-rata untuk pencarian dokumen yang mencerminkan maksud pengguna adalah kurang dari 2 detik. Kesimpulan performa aplikasi dalam kategori optimal dalam segi waktu karena aplikasi telah dioptimasi dengan menggunakan thread dan hasil pencarian yang diberikan mampu memberikan hasil secara semantic.

Kata Kunci: *Clustering, suffix tree, semantic,* pencarian dokumen, android

ABSTRACT

Anroid smartphone device has been involved in user's way of living in this modern era. Smartphone device is used in user's daily activity such as reading and storing electronic document in Pdf, Word and other file formats. User might and frequently forgot elctronic document's directory in the smartphone.

This research aims to help user to find documents that reflect user's keyword semantically or literally. Documents that reflect user keyword semantically or literaly will be shown. Various method is tested to minimize time use in clustering using suffix tree to semantic searching processes.

This research finds that user could find documents in the smartphone that reflect user's keyword. Average time use for clustering about 100 documents containing 1000 word for each document is 686.7 seconds. User is able to search for document right after clustering process is done. Average time use for document searching is less than 2 seconds. Hence, thread implementation for processes decrease time consume greatly and the search result displayed to the user represents document content semantically.

Keywords: *Clustering, suffix tree, semantic, document searching,* android

1. INTRODUCTION

Perangkat smartphone berbasis android sudah menjadi bagian hidup bagi penggunanya di zaman modern seperti saat ini. Pengguna menggunakan sartphone untuk mendukung aktivitas harian seperti diantaranya adalah membaca dan menyimpan dokumen elektronik Pdf, Word dan lain-lain. Manusia cenderung lupa terhadap lokasi penyimpanan sebuah dokumen elektronik terutama jika dokumen elektronik tersimpan cukup lama di dalam smartphone, pengguna tidak membuka dokumen elektronik tersebut dalam waktu yang relatif lama dan faktor lainnya Kemungkinan menyimpan sebuah dokumen elektronik tanpa nama yang jelas sering terjadi pada pengguna smartphone. File dokumen elektronik diberi nama secara asal seperti "file baru.doc", "coba.pdf", "13-02-33.pdf", "new.doc" dan lainnya untuk alasan praktis. Diharapkan dengan menggunakan metode semantic suffix tree clustering (SSTC) untuk pencarian dokumen di android dapat menyelesaikan latar belakang masalah tersebut.

2. TINJAUAN PUSTAKA

Semantic similiarity adalah untuk melakukan identifikasi konsep yang memiliki kesamaan "karakter". Manusia dapat melakukan interpretasi terhadap relasi antar konsep meskipun tidak dapat mendefinisikan relasi tersebut secara formal. Sebagai contoh, seorang anak kecil dapat mengerti bahwa antara "apel" dan "jeruk" lebih memiliki persamaan hubungan dibandingkan dengan kata-kata "apel" dan "pasta gigi". [3]

WordNet adalah sebuah database thesaurus untuk bahasa Inggris yang digunakan dan dimanfaatkan untuk meningkatkan kualitas proses text clustering. WordNet memetakan semua kata dasar, kata imbuhan, keterkaitan kata terhadap kata dasar tersebut. WordNet versi 2.1, memiliki 207,016 pasangan dari term-sense, 155,327 terms, dan 117,597 sense. [5]

WordNet adalah sebuah produk riset dari Princeton University mengenai pembuatan model dari lexical knowledge untuk bahasa Inggris. Kata-kata yang memiliki beberapa sense (makna) akan berada di berbagai synset [3]

Suffix Tree Clustering(STC) adalah sebuah algoritma *clustering* berbentuk pohon struktur *data* dalam menghadapi masalah *string* dasar. yang membutuhkan waktu berbanding lurus dengan *data* yang di proses. STC melakukan identifikasi

hubungan antar dokumen berdasarkan untaian kata di dalam masing-masing dokumen. STC dapat dibentuk secara *incremental* berarti *tree* akan berkembang seiring tambahan *input* baru tanpa merombak *suffix tree* dari semula. Pengertian dari suffix tree pada poin di bawah ini[7]

- Suffix tree dari String S adalah sebuah compact trie.
- Trie adalah sebuah tipe tree yang memiliki cabang / branch yang dimungkinkan untuk terbentuk dari setiap node sebanyak alphabet yang ada dalam sebuah string
- Suffix tree memiliki root yang memiliki cabang yang terarah dan teridentifikasi
- Internal node memiliki minimal 2 anak
- Setiap edge harus memiliki label string (teridentifikasi sebagai trie)
- Tidak ada edge dalam node yang sama yang memiliki awalan kata yang sama (teridentifikasi sebagai compact)
- Node yang tidak memili percabangan lagi disebut leaf
- Suffix-node harus mengandung sufix dari S untuk setiap sufix dari S.

Suffix Tree Clustering pertama kali ditemukan oleh Weiner pada tahun 1973 kemudian dikembangkan oleh Ukkonen pada tahun 1995. Algoritma Ukkonen membaca string dari kiri-ke kanan tanpa pengulangan. Algoritma Weiner membaca string dari belakang dokumen. Algoritma Ukkonen lebih efisien dalam hal waktu dan memory dibandingkan dengan Algoritma Weiner

Semantic Suffix Tree Clustering (SSTC) akan menghasilkan cluster lebih spesifik, label yang mudah teridentifikasi dibandingkan dengan STC biasa. [2]

Terdapat 3 tahap dalam mendapatkan hasil clustering yang baik. Tahap pertama 'membersihkan' dokumen (preprocessing) tahap kedua melakukan clustering dan tahap ketiga merapikan cluster yang ada [6].

Tahap preprocessing terbagi dalam 4 tahap proses yaitu *tokenizing, filtering* dan *stemming*. [1]

Tahap kedua melakukan proses suffix tree clustering dan pemberian bobot pada setiap cluster.

Tahap ketiga adalah menggabungkan cluster yang sejenis (*semantic*). Berbagai dokumen dimungkinkan memiliki kesamaan lebih dari 1 frasa. Dengan demikian bisa didapatkan jika dokumen tersebut adalah identik dan sejenis jika memenuhi rumus. dimana Ca adalah kata dalam *cluster* a dan Cb adalah kata dalam *cluster* b. [2]

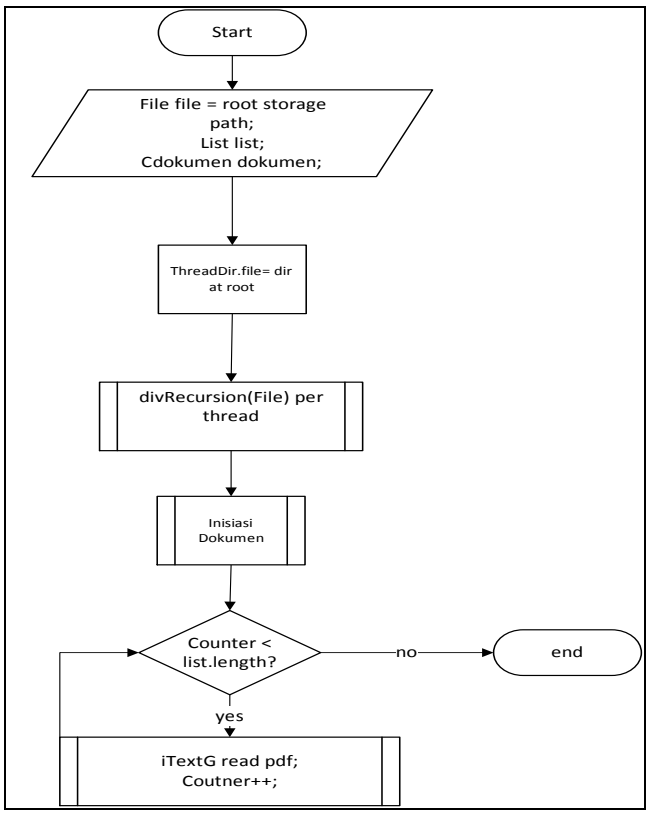
$$ClusSim\,(Ca, Cb) = 1\ if\ |Ca \cap Cb| = |Ca|, Ca\ is\ deleted$$
$$or\ |Ca \cap Cb| = |Cb|, Cb\ is\ deleted$$
$$ClusSim(Ca, Cb) = 0\ otherwise \quad (1)$$

3. METODE

Terdapat 3 proses Utama untuk melakukan cluster. Proses pertama adalah membaca atau parsing isi dokumen dengan ekstensi ".pdf" dan ekstensi ".TRY" di dalam smartphone. Proses kedua adalah melakukan STC dari data string yang telah didapat pada proses pertama. Proses ketiga adalah penyimpanan hasil STC di dalam smartphone.

Terdapat 3 proses untuk melak ukan pencarian. Proses pertama adalah preprocesing input. Proses kedua adalah search semanticaly antara input dengan hasil SSTC. Proses ketiga adalah menampilkan hasil searching.

Untuk memindai seluruh dokumen menjadi satu kesatuan maka membutuhkan proses dokumen listing dan parsing file dokumen yang dapat dilihat pada Gambar 1.



Gambar 1. Flowchart Parsing Dokumen

Proses preprocessing menghilangkan kata yang tidak merubah kalimat meskipun dihilangkan [4], merubah format kapital menjadi non-kapital, menghilangkan seluruh tanda baca (selain angka dan huruf a-z) dan tokenizing berdasarkan whitespace spasi. Preprocessing menggunakan regular expression untuk proses match string dan replace

Proses *sorting* dokumen dibutuhkan untuk mendukung binary search dan pemberian nilai index dokumen.

Nilai index dokumen sebagai alias adalah fitur yang paling penting dalam pemberian nama *edge* pada pembentukan *suffix tree*. Labeling *edge* tidak menyimpan *string* konten dokumen melainkan index alias dengan tujuan penghematan penggunaan memory pada aplikasi.

Pembentukan suffix tree memperhatikan beberapa peraturan penting sebagai berikut

1. Peraturan iterasi baru, nilai finger (penunjuk string dokumen yang akan di scan) dan nilai end ditambah 1.
2. Peraturan penunjuk kata aktif, jika terjadi perubahan penunjuk kata aktif maka, iterasi saat itu berakhir dan memulai iterasi baru
3. Peraturan percabangan. Jika setelah kata aktif tidak sama dengan finger maka terjadi percabangan dari kata aktif. Active berubah menjadi root kemudian memeriksa(menjelajah) setiap list word dari root.
4. Peraturan listword, penambahan dan shift listword.
   a. Listword sebagai penampung untaian kata finger yang belum terbentuk di tree

b. Memenuhi nomer 2 maka list.add(finger)

c. Memenuhi nomer 3 maka list.shiftleft()

5. Peraturan suffix link. Setiap node baru yang terbentuk memiliki suffix link root. Jika pada iterasi yang sama terjadi pembentukan node baru, maka suffix link node baru sebelumnya pada iterasi yang sama berubah menjadi node baru yang terbentuk.

Merging cluster atau merging cluster adalah fitur yang sangat penting untuk menghemat memory yang digunakan oleh aplikasi jika diimplementasikan secara tepat. Proses merging dapat menghemat space untuk data pada database, menghemat waktu dalam proses iterasi pencarian, menghemat waktu dalam proses iterasi fetch data dari tabel database

Rumus jumlah maksimal cluster yang terbentuk dengan menggunakan metode dokumen sejenis adalah sebagai berikut dimana n adalah jumlah dokumen.

$$s(n) = \sum_{i=1}^{n} n\,C\,i \qquad (2)$$

Pemberian nilai identik antara cluster dengan input user agar sesuai dengan kebutuhan pengguna yang hemat memory dan waktu adalah sebagai berikut dimana Km adalah kata asli pada cluster dan Bu adalah synset dari input user.

$$|Km \cap Bu| > 0 \qquad (3)$$

Proses mendapatkan cluster yang identik dengan pengguna dapat dilakukan setelah mendapatkan synset dari input user

Proses pencarian dengan query "where like '%kata%'" memiliki kekurangan yaitu memberikan hasil dengan syarat asalkan terdapat substring kata dalam sebuah string kata secara utuh. Proses filter berdasarkan regex diperlukan untuk mengatasi masalah ini. String match regular expression digunakan untuk mendapatkan cluster yang hanya mengandung synset input user.

String *replace* regular expression digunakan untuk merubah tampilan kata identik menjadi berwarna.

4. IMPLEMENTASI

Kebutuhan aplikasi terhadap tujuan aplikasi membutuhkan banyak class pendukung. Class pendukung memetakan kebutuhan algoritma program menjadi sebuah objek.

Cara penggunaan objek CSTC berada pada iterasi hingga status done. Dapat dilihat pada segmen 1. Proses looping untuk objek CSTC dibutuhkan untuk menghindari terjadinya *stack overflow* akibat dari pemanggilan fungsi secara rekursiv untuk melakukan proses selanjutnya.

Pengujian pertama penggunaan aplikasi dilakukan terhadap total konsumsi waktu dari proses parsing dokumen hingga input database, total konsumsi waktu proses pencarian dan interface aplikasi. Seluruh pengujian pada subbab ini menggunakan variabel 101 dokumen yakni 49 file ekstensi pdf dan 52 file ekstensi TRY yang tersebar pada direktori smartphone.

Jumlah *cluster* yang terbentuk pada variabel ini adalah 4699 buah *cluster* yang dapat dilihat pada Gambar 2. Detail pengujian proses *clustering* terhadap waktu dalam satuan detik dapat dilihat pada Tabel 1. Pada Tabel 1. Menanpilkan 10 kalil percobaan dengan untuk mendapatkan rata-rata dan standar deviasi

Segmen 1. Objek CSTC

```
CSTC classStc = new CSTC(_dokumen);

        int lastDokumen = classStc.getDokumenLastIndex();

        while(!classStc._done){
            if(classStc.getGoingTo().equals("start")) {
                classStc.start();
            }else if( classStc.getGoingTo().equals("p6relBreaker")){
                classStc.k6();
            }else if(classStc.getGoingTo().equals("k11")){
                classStc.k11();
            }
            setText(_textClusteringStatus,""+classStc.getFinger()+"/"+lastDokumen);
        }
        setText(_textClusteringStatus,"Done "+ processTime());
        setText(_textRetrievingCluster,"Processing");
```

Tabel 1. Jumlah Proses Clustering

No Pengujian	Listing Dokumen	Parsing Dokumen	Preproces	Tree	retrieve	Merging	Insert DB	Total
1	3	121	3	544	1	19	4	695
2	4	101	3	546	1	17	4	676
3	5	101	3	552	1	19	4	685
4	5	103	3	550	0	19	4	684
5	5	107	3	547	0	18	4	684
6	4	128	3	559	1	19	3	717
7	3	104	3	535	1	18	4	668
8	4	128	2	541	1	18	4	698
9	4	105	3	545	0	19	3	679
10	4	105	3	545	1	19	4	681
Avg	4.1	110.3	2.9	546.4	0.7	18.5	3.8	686.7
Min	3	101	2	535	0	17	3	668
Max	5	128	3	559	1	19	4	717

Pengujian dilakukan sebanyak 10 kali. Hasil pengujian dapat dikatakan konsisten antara hasil pengujian yang satu dengan yang lainnya. Nilai sample standard deviation terhadap total waktu proses adalah 13.69 detik. Nilai Population standard deviation terhadap total waktu proses adalah 12.99. Nilai rata-rata total waktu proses adalah 686.7 detik.

Nila confidence interval waktu yang dibutuhkan jika menggunakan distribusi normal dapat dilihat pada Tabel 2.

Tabel 2. Confidence Interval

Confidence Interval	Range
68.3%, σ	673.00450195624 - 700.39549804376
90%, 1.645σ	664.17090571801 - 709.22909428199
95%, 1.960σ	659.85682383423 - 713.54317616577
99.9999%, 4.892σ	619.70162356992 - 753.69837643008
99.999%, 4.417σ	626.20698514071 - 747.19301485929

Jumlah waktu yang dibutuhkan untuk pencarian dapat dilihat pada Tabel 3. Nilai waktu dipengaruhi oleh variabel jumlah data yang dikembalikan oleh database sqlite.

Tabel 3. Jumlah Waktu Proses Pencarian

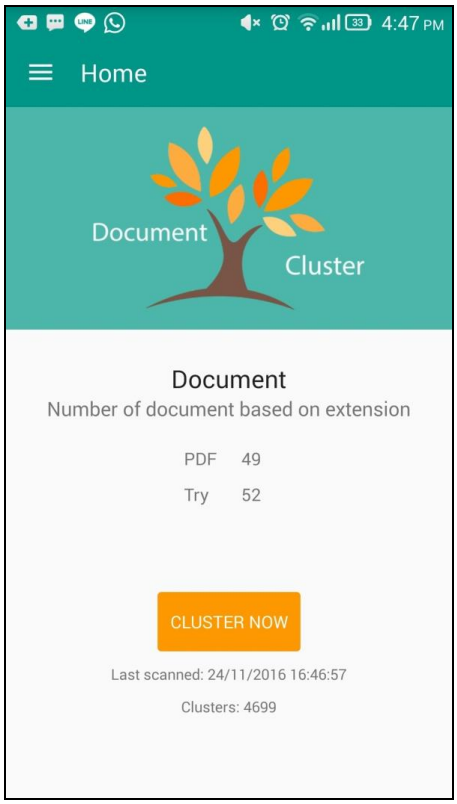
no	*Input*	n *Cluster*	t	t/n
1	god	5	0.439	0.0878
2	war	9	0.844	0.093778
3	movie	8	1.111	0.138875
4	food	3	0.225	0.075
5	cartoon	2	0.302	0.151
6	armor	3	0.589	0.196333
7	dangerous	2	0.206	0.103
8	bite	5	1.451	0.2902
9	race,big, fast god	26	2.571	0.098885
10	god,poison,war,movie	24	2.694	0.11225
			Avg	0.134712
			Min	0.075
			Max	0.2902

Pengujian kedua menguji aplikasi per-10 dokumen terhadap waktu dilakukan dengan menggunakan variabel file ekstensi TRY sebagai file ascii yang mengandung 1000 kata. pengujian hanya menggunakan file ekstensi TRY bertujuan untuk mendapatkan konsumsi waktu tanpa ada faktor library pembaca file PDF. Pengujian dapat dilihat pada Tabel 4.

Tabel 4. Pengujian Per-10 Dokumen

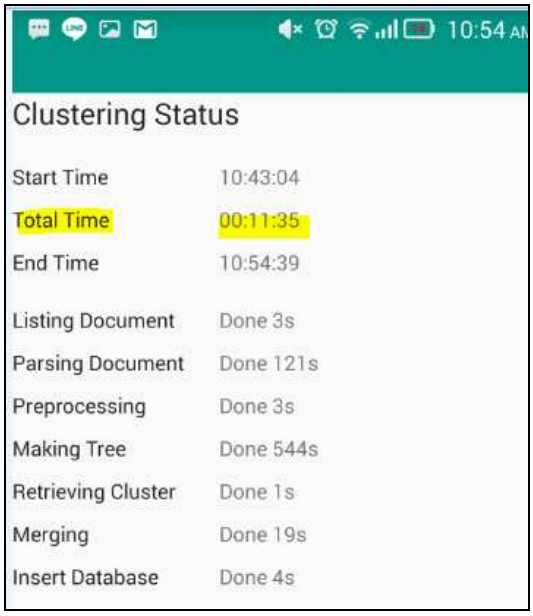
N File	Listing Dokumen	Parsing Dokumen	Preprocessing	Making Tree	Retrieving Cluster	Merging	Insert DB	Total t	t/N
10	1	1	0	10	0	1	0	13	1.3
20	2	0	1	28	1	1	0	33	1.65
30	6	0	1	55	0	2	1	65	2.166667
40	2	0	1	91	0	3	2	99	2.475
50	2	1	2	141	0	4	2	152	3.04
60	4	0	3	192	0	6	2	207	3.45
70	2	1	2	264	0	9	2	280	4
80	1	1	2	334	1	10	3	352	4.4
90	3	1	3	418	1	13	4	443	4.922222
100	2	0	3	495	1	15	4	520	5.2
110	3	1	3	564	2	13	5	591	5.372727
120	3	1	3	675	1	22	5	710	5.916667
130	2	2	5	785	1	25	5	825	6.346154
140	2	0	4	826	1	28	5	866	6.185714
150	2	1	3	942	1	36	5	990	6.6

Halaman utama pada aplikasi menampilkan informasi *cluster* dan *file* tombol untuk proses cluster. Pengguna akan melihat sebuah logo, kuantitas dokumen, tombol "*CLUSTER* NOW" dan keterangan proses *cluster* pada halaman utama. *Last scanned* mencatat waktu terakhir proses *cluster* dilakukan . tampilan halaman utama dapat dilihat pada Gambar 2.



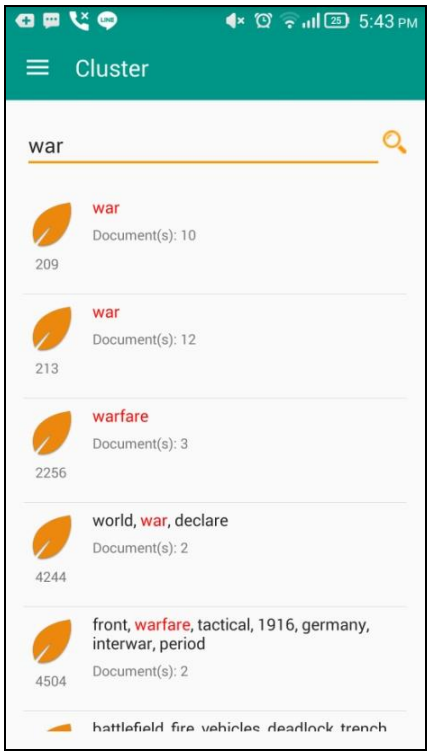
Gambar 2 Halaman Utama Aplikasi

Halaman *status cluster* pada Gambar 3. akan muncul ketika pengguna menekan tombol "*Cluster* Now". Kolom sebelah kiri adalah nama proses dan kolom sebelah kanan adalah status proses.



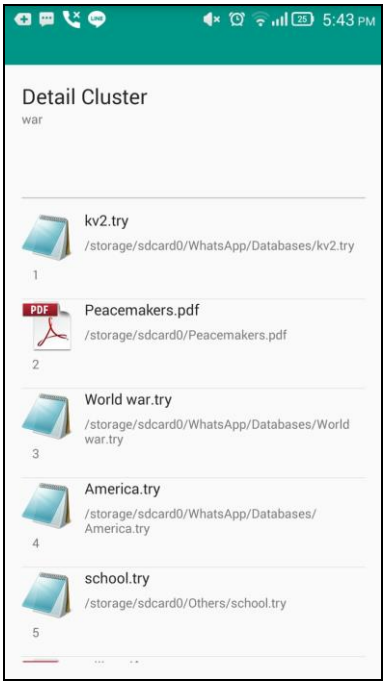
Gambar 3. Halaman Status Proses *Cluster*

Hasil pencarian pada aplikasi *Document Cluster* menemukan *synset* kata 'war' dan 'warfare' pada 9 *cluster* dalam 1 detik. Hasil pencarian dapat dilihat pada Gambar 4. Jumlah dokumen yang terdapat pada *cluster* pertama dengan konten kata war adalah 10 dokumen, jumlah dokumen pada *cluster* ke dua adalah 12 dan seterusnya



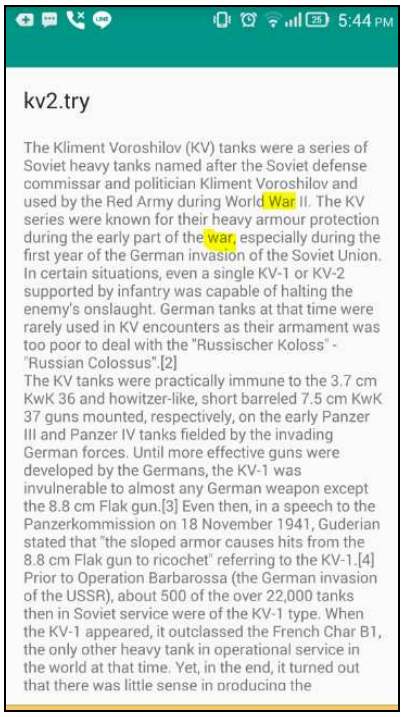
Gambar 4. Halaman Hasil Pencarian

Daftar dokumen pada *cluster* kedua akan muncul jika pengguna menekan *cluster* kedua. Kesimpulan *cluster* kedua adalah dokumen 'kv2.TRY', 'Peacemakrs.PDF', 'World war.TRY', 'America.TRY', 'school.TRY' dan selanjutnya adalah dokumen mengenai 'war'. Detail *cluster* berisi daftar dokumen dapat dilihat pada Gambar 5.



Gambar 5 Halaman Detil Cluster

Dokumen akan ditampilkan pada layar smartphone jika pengguna menekan dokumen. Dokumen pertama ditampilkan pada layar ketika pengguna menekan dokumen pertama. Isi dari dokumen pertama dapat dilihat pada Gambar 6.



Gambar 6. Halaman Teks Isi Dokumen

5. KESIMPULAN

Hasil penelitian menunjukan bahwa pengguna dapat mencari dokumen yang dimaksud di dalam smartphone pengguna. Konsumsi waktu rata-rata untuk proses *cluster* terhadap 100 dokumen yang mengandung kurang lebih 1000 kata adalah 686.7 detik. Pengguna dapat mencari dokumen yang mencerminkan maksud pengguna setelah proses *cluster* dilakukan. Konsumsi waktu rata-rata untuk pencarian dokumen yang mencerminkan maksud pengguna adalah kurang dari 2 detik. Kesimpulan performa aplikasi dalam kategori optimal dalam segi waktu karena aplikasi telah dioptimasi dengan menggunakan *thread* dan hasil pencarian yang diberikan mampu memberikan hasil secara *semantic*.

6. REFERENCES

[1] Al-Anazi S., AlMahmoud H. & Al-Turaiki I. 2016. Finding similar documents using different clustering technique. *Procedia Computer Science*, *82 28–34*

[2] Janruan, J. & Guha. S. 2011. Semantic Suffix Tree Clustering. *IEEE*, 978-1-4244-8581-9/11, 35-40.

[3] Liu, H., Bao, H. & Xu, D. 2011. Concept vector for semantic similarity and relatedness based on WordNet structur. The Journal of Systems and Software 85 (2012) 370– 381

[4] *Stopwords.* Retrieved from Ranksnl website: http://www.ranks.nl/stopwords

[5] Wei, T., Lu, Y., Chang, H,. Zhou, Q., & Bao, X,. 2014. A semantic approach for text clustering using WordNet and lexical chains. *Expert Systems with Applications, 42 (2015) 2264–2275*

[6] Zamir, O. & Etzioni, O. 1998. Web Document Clustering: A Feasibility Demonstration. *ACM*, 1-58113-015-5 8/98

[7] Zhuang, Y. & Chen, Y. 2015. Improving *Suffix Tree Clustering* Algorithm for Web *Document*. *Atlantis Pres*