TELEMATIKA, Vol. 15, No. 01, APRIL, 2018, Pp. 1 – 12
ISSN 1829-667X

PENCARIAN FILE PADA ANDROID BERBASIS SUFFIX TREE CLUSTERING DENGAN DUKUNGAN WORDNET

Adi Wibowo[(1)], Justinus Andjarwirawan[(2)], David Valentino[(3)]
Teknik Informatika, Fakultas Teknologi Industri, Universitas Kristen Petra
Jl. Siwalankerto 121-131, Surabaya 60236, Indonesia
e-mail : adiw@petra.ac.id[(1)], justin@petra.ac.id[(2)], davlnt23@gmail.com[(3)]

Abstract

Finding a specific file in Android devices is not an easy task. Not many apps can search by files' contents and find terms similarities between user's queries and files' terms. This research proposed using Suffix Tree Clustering to index files contents, and WordNet to expand user's query terms. This research used waterfall as research methodology to built a search engine prototype. There are five steps to index files, i.e. listing and parsing, preprocessing, clustering, merging of clusters, and storing cluster data into database. If a user wants to search files, the prototype will expand user's query terms using WordNet's synsets and compare them with clusters stored in a database. The results of this research show that suffix tree clustering and multithreading can be used to index files' contents, and term expansion can help users to find clusters similar with user's query terms.

Keywords : Suffix Tree Clustering, multithreading, WordNet

Abstrak

Tidak mudah menemukan file tertentu pada sebuah perangkat Android. Tidak banyak aplikasi dapat mencari file berdasarkan isinya, dan membandingkan kemiripan term antara kata kunci pengguna dengan term-term yang berasal dari isi file. Penelitian ini mengusulkan penggunaan Suffix Tree Clustering untuk mengindeks isi file, dan WordNet untuk mengembangkan term-term dari kata kunci pengguna. Penelitian ini menggunakan metode waterfall untuk membangun prototipe dari mesin pencari file. Ada lima langkah untuk mengindeks file, yaitu listing dan parsing, preprocessing, clustering, penyatuan cluster, dan penyimpanan data cluster ke database. Bila seorang pengguna ingin mencari file, prototipe akan mengembangkan kata kunci pengguna berdasarkan synset dari WordNet dan membandingkannya dengan data cluster dari database. Hasil penelitian ini menunjukkan bahwa suffi tree clustering, multithreading dapat digunakan untuk mengindeks isi file, dan pengembangan kata kunci membantu pengguna mencari cluster-cluster yang mirip dengan kata kunci tersebut.

Kata Kunci : ***Suffix Tree Clustering, multithreading, WordNet***

1. PENDAHULUAN

Setiap alat (device) yang berbasis sistem operasi tertentu tentunya memiliki filesystem. Demikian juga dengan Android yang dikembangkan dari sistem operasi Linux yang bahkan menganggap seluruh UNIX namespace sebagai struktur tree dari file (Pate, 2003). Filesystem didefinisikan sebagai metode dan struktur data yang digunakan oleh sistem operasi untuk mencatat berkas-berkas (files) pada sebuah partisi, dengan kata lain cara bagaimana berkas-berkas diatur di dalam sebuah disk (Wirzenius, Oja, Stafford, & Weeks, 2005). Pada filesystem dapat disimpan berkas terkait dengan program tertentu, atau berkas yang berisi data pengguna. Contoh berkas data pengguna adalah berkas-berkas foto, dokumen dengan format Microsoft Word, Portable Document Format (PDF), atau berkas-berkas lainnya.

Seringkali pengguna Android ingin mencari kembali berkas-berkas yang pernah tersimpan pada device-nya. Misalnya pengguna ingin menemukan kembali berkas-berkas terkait pekerjaan, atau pendidikan yang pernah ia simpan dalam perangkat Androidnya. Untuk mencari berkas-berkas tersebut pengguna menggunakan aplikasi seperti find, locate, My Files, atau ES File Explorer. Tiap aplikasi menggunakan algorituma tertentu untuk mencari berkas yang tersimpan dalam sebuah filesystem. Algoritma yang banyak digunakan di Linux adalah Levenshtein distance algorithm. Algoritma Levenshtein menghitung kemiripan atau jarak antara dua string

(Lhoussain, Hicham, & Abdellah, 2015). Jarak antar string adalah jumlah penghapusan, pengisian, atau pengubahan karakter yang diperlukan untuk mengubah satu string (source string) menjadi string yang lain (target string). Kelemahan dari algoritma Levenshtein adalah tidak mampu mengenali relasi antar kata berbeda yang memiliki makna yang sama; atau kata-kata yang adalah variasi kata pada makna yang sama, seperti *mouse* dan *mice* (Hussain, 2012). Hal ini mengakibatkan pengguna harus memasukkan kata kunci yang tepat saat melakukan pencarian berkas.

Untuk menjawab masalah tersebut penelitian ini mengusulkan penggunaan algoritma Suffix Tree Clustering untuk melakukan pencarian berkas-berkas pada filesystem yang digunakan oleh Android. Penggunaan clustering sebagai alat pencarian dimaksudkan agar proses pencarian tidak hanya menghasilkan berkas yang persis sama dengan kata kunci yang dicari pengguna, tetapi juga berkas-berkas lain yang mirip yang masuk pada cluster yang sama. Hal ini memberi keuntungan bagi pengguna karena pengguna tidak perlu lagi mencari menggunakan kata kunci lain untuk mencari berkas-berkas yang mirip. Algoritma Suffix Tree Clustering dipilih karena algoritma ini adalah *description-centric* yang menekankan pada bagaimana bentuk deskripsi ditampilkan dari tiap cluster yang ditemukan (Marco & Navigli, 2013). Hal ini berbeda dengan algoritma clustering yang bertipe *data-centric* yang lebih mengutamakan problem penyelesaian clusternya dibandingkan dengan bagaimana dapat menampilkan hasil clustering yang lebih deskriptif.

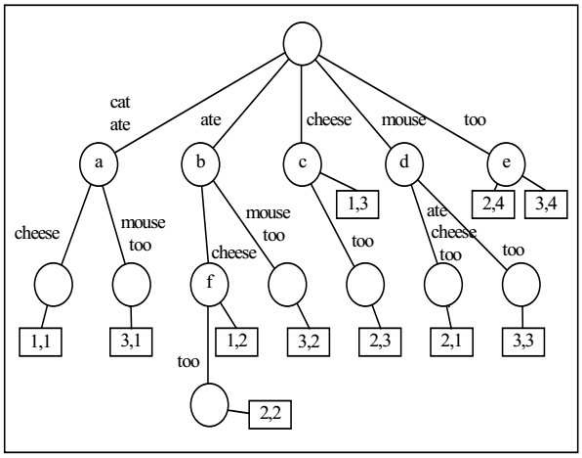
Algoritma suffix tree clustering memiliki kelemahan, yaitu membutuhkan kapasitas penyimpanan yang besar, dan tidak dapat mengenali sinonim dari tiap term yang mewakili cluster (Dang, Zhang, Lu, & Zhang, 2013). Dang dkk. menyarankan penggunaan WordNet untuk memberi informasi mengenai sinonim (synset) untuk tiap term dan dalam menyusun node dari tree juga mengunakan sinonim tersebut. AlAgha dan Nafee (2015) menyatakan bahwa dalam penggunaan WordNet untuk membantu proses clustering perlu memperhatikan diversifikasi topik yang terdapat dalam WordNet. Bila synset dalam WordNet tidak memiliki penyebaran topik-topik yang sama dengan topik-topik dari sumber data dari proses clustering, maka penggunaan WordNet tidak akan banyak berpengaruh. AlAgha dan Nafee juga menyarankan bahwa karena WordNet tidak memilki data spesifik tentang tingkat kedekatan antar kata, maka penggunaan Is-A, hiponim, dan hipernim dapat dipertimbangkan untuk mendapatkan informasi kedekatan antar kata.

Pada penelitian ini karena kapasitas komputasi dan penyimpanan (*storage*) dari perangkat Android sangat terbatas, maka untuk membantu proses pencarian, hanya kata kunci dari pengguna yang akan dikembangkan (*term expansion*) menggunakan WordNet. Tree yang dibuat melalui algoritma Suffix Tree tidak akan dipengaruhi oleh synset dari WordNet. Hal ini dilakukan agar dalam proses pencarian masih dapat mencari menggunakan term-term yang memiliki sinonim yang sama dengan kata kunci yang dimasukkan pengguna, tetapi tree yang dibuat tetap dapat berukuran lebih kecil.

2. STUDI LITERATUR

2.1. Suffix Tree Clustering

Suffix Tree Clustering (STC) adalah proses pembuatan cluster dengan cara mengidentifikasi frasa-frasa yang banyak (*common*) terdapat di sekumpulan dokumen (Janruang, Guha, 2011). Frasa didefinisikan sebagai urutan teratur dari satu atau lebih kata. STC memiliki tiga langkah, yaitu: (1) pembersihan dokumen, (2) idenfitifasi *base cluster* menggunakan suffix tree, dan (3) menggabungkan beberapa *base cluster* menjadi *cluster* baru. Pada proses pembersihan dokumen dilakukan pemotongan kalimat menjadi token, lalu token-token yang adalah stopword dihilangkan, menghilangkan tanda baca, dan mengubah huruf kapital menjadi huruf kecil. Pada proses pembentukan *base cluster*, setiap kalimat dari tiap dokumen yang telah dibersihkan pada tahap 1 disusun menjadi tree dimulai dari token terakhir dari kalimat tersebut dan bergerak mundur ke arah token yang lebih awal. Untuk tiga dokumen dengan kalimat masing-masing berupa “cat ate cheese”, “mouse ate cheese too”, dan “cat ate mouse too”, proses kedua menghasilkan tree seperti terlihat pada Gambar 1.

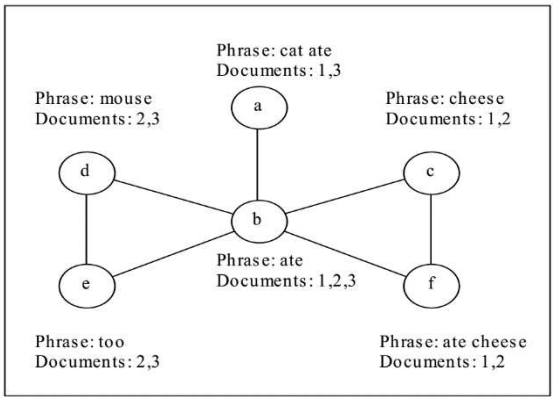


Gambar 1. Proses Pembentukan Suffix Tree (Zamir & Etzioni, 1998).

Suffix dari kalimat yang diproses akan dicatat pada edge. Node menunjukkan penggabungan dari seluruh edge yang berawal dari root dan berakhir pada node tersebut. Pada setiap leaf node akan memiliki catatan yang menunjukkan node tersebut muncul pada dokumen nomor berapa, dan pada suffix berapa. Tiap node yang memiliki dua atau lebih dokumen disebut sebagai base cluster. Pada Gambar 1 terdapat enam base cluster yang ditunjukkan dengan huruf a hingga f.

Proses ketiga menggabungkan beberapa base cluster yang mirip ke dalam satu cluster. Proses menggabungkan cluster dilakukan dengan menghitung nilai kemiripan antara cluster yang saling overlap. Persamaan nilai kemiripan antar cluster ditunjukkan oleh Pers. (1). Bm dan Bn adalah dua base cluster yang dinilai kemiripannya. adalah jumlah dokumen yang ada di kedua cluster, |Bm| dan |Bn| adalah jumlah dokumen di setiap base cluster. Hasil penggabungan base cluster untuk tiga dokumen di atas dapat dilihat pada Gambar 2.

$$\frac{|Bm \cap Bn|}{|Bm|} > 0.5 \; And \frac{|Bm \cap Bn|}{|Bn|} > 0.5 \qquad (1)$$



Gambar 2. Hasil Penggabungan Base Cluster (Zamir & Etzioni, 1998).

Pada penelitian ini digunakan persamaan untuk menyatukan base cluster yang diajukan oleh Janruang dan Guha (2011) yang ditunjukkan pada Pers. (2). Pada persamaan ini kemiripan antar cluster dihitung apakah kata-kata dalam satu cluster termasuk dalam cluster lainnya.

$$ClusSim\ (Ca, Cb) = 1\ if\ |Ca\ \cap Cb| = |Ca|, Ca\ is\ deleted \quad (2)$$
$$or\ |Ca\ \cap Cb| = |Cb|, Cb\ is\ deleted$$
$$ClusSim\ (Ca, Cb) = 0\ otherwise$$

Tabel 1. Contoh WordNet.

Kata-kata yang berelasi dengan "Computer"	
Jenis Relasi	***Kata-kata yang berelasi***
Synonym	Computing machine, computing device, data processor
Hypernym	machine
Hyponym	Node, client, guest, server, host, turing machine, website.
Meronym	Bus, data converter, monitor, peripheral, disk cache

2.2. WordNet

WordNet adalah sebuah pusat data yang berisi term-term dalam bahasa Inggris dan relasi-relasi antar term tersebut. Term dapat berupa kata benda, kata kerja, kata sifat, atau kata keterangan. Term-term dikelompokkan ke dalam kumpulan-kumpulan sinonim (sets of cognitive synonyms / synsets). Tiap synset akan mengekspresikan konsep-konsep term yang berbeda (Princeton University, 2010). Relasi antar term digunakan untuk menggambarkan relasi antar synset, yaitu:

1. Hypernym. Kata yang maknanya menunjukkan kelas yang lebih tinggi. Binatang adalah hypernym dari kucing.
2. Hyponym. Kata yang maknanya termasuk dalam makna kata lainnya. Hyponym adalah lawan dari hypernym. Kucing adalah hyponym dari binatang.
3. Holonym. Kata yang menunjukkan makna keseluruhan (lebih utuh) dibanding kata lain yang maknanya adalah bagian (part) dari holonym. Contoh: bangunan adalah holonym dari jendela.
4. Meronym. Kata yang menunjukkan bagian dari kata yang lebih besar. Jendela adalah meronym dari bangunan.
5. Coordinate term. Sebuah kata adalah coordinate term dari kata lainnya bila kedua kata memiliki hypernym yang sama. Kucing adalah coordinate term dari harimau.

Contoh dari WordNet synsets ditunjukkan di Tabel 1.

3. METODE PENELITIAN

Metode yang digunakan adalah melalui pembuatan prototype menggunakan metode waterfall. Metode waterfall terbagi atas langkah Planning, Analysis, Design, Implementation, dan Deployment. Pada langkah planning dilaksanakan proses studi pustaka mengenai suffix tree clustering, WordNet, dan pengembangan aplikasi berbasis Android. Pada langkah analysis dan design dilakukan pembuatan flowchart proses indexing dan searching dari suffix tree clustering, meliputi proses listing dan parsing, clustering, penyimpanan ke database, dan pencarian kembali cluster yang sesuai dengan kata kunci yang dimasukkan pengguna menggunakan WordNet. Pada tahap ini juga dilakukan perencanaan user interface dari prototype yang memudahkan pengguna mengindeks dan mencari file. Tahap implementation menerapkan flowchart dari setiap proses di atas ke dalam bahasa Java. Pada tahap terakhir dilakukan proses pengujian dan deployment hingga prototype dapat digunakan tanpa ditemukan bug yang menghalangi fungsi prototype.

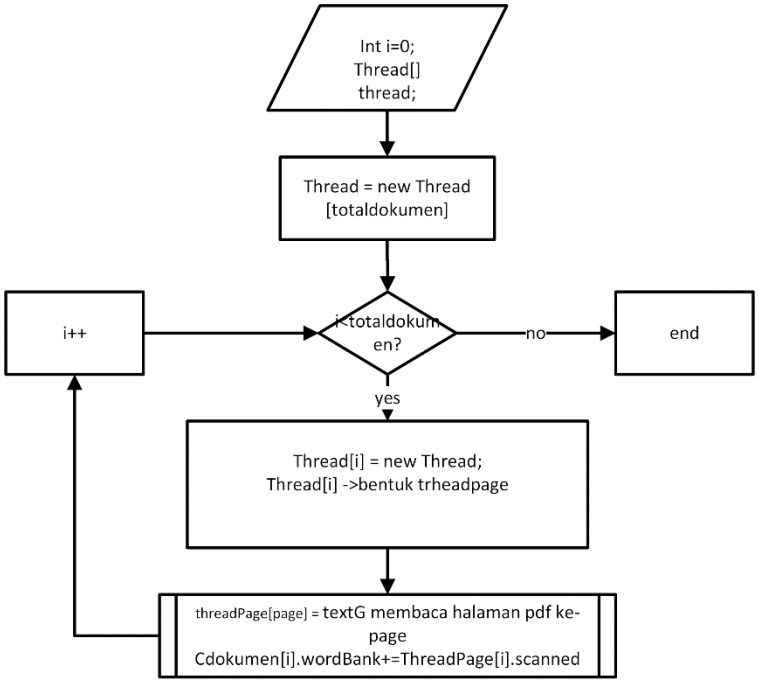
4. HASIL DAN PEMBAHASAN

Cara kerja sistem terbagi atas dua bagian besar, yaitu proses indexing, dan proses searching. Proses indexing adalah proses menemukan seluruh file dengna format PDF dan TXT pada device storage, dan kemudian membentuk base cluster-base cluster sesuai Suffix Tree Clustering. Proses indexing hanya dijalankan bila ada file PDF atau TXT baru yang disimpan dalam storage. Proses searching menerima kata kunci dari pengguna, lalu membandingkan kata kunci tersebut dengan base cluster-base cluster yang ditemukan saat proses indexing kamudian menampilkan base cluster yang sama.

4.1. Proses Indexing

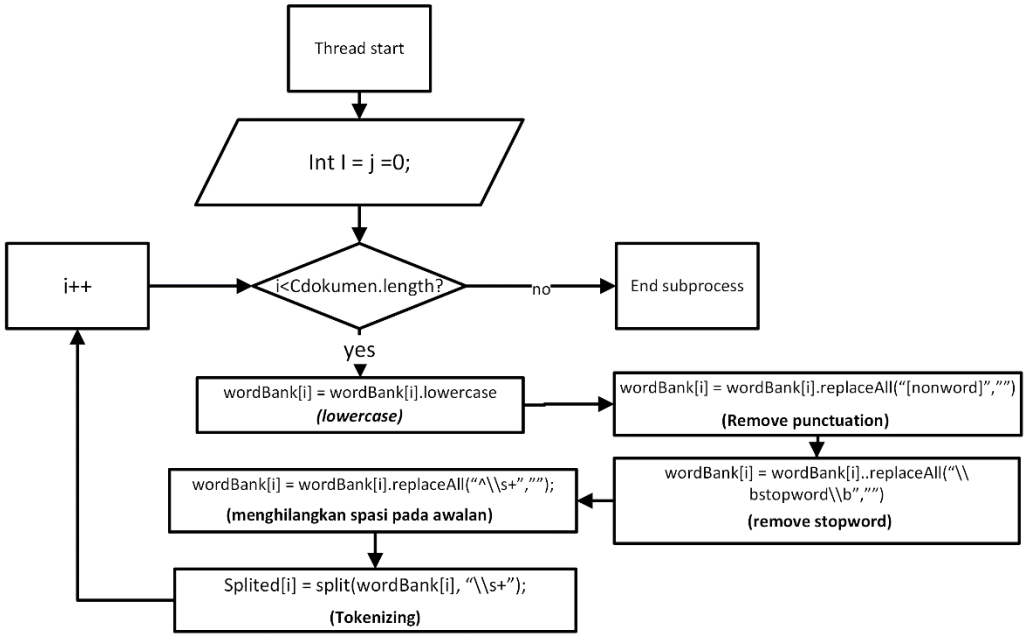
Proses indexing terdiri atas beberapa langkah, yaitu:

1. Listing dan Parsing
   Proses pencarian file PDF dan TXT pada storage dilakukan menggunakan multi-thread. Setiap thread akan melakukan pencarian rekursi dimulai dari root folder. Setiap nama file yang didapat akan disimpan dalam variabel list. Setiap file tersebut kemudian diproses menggunakan library iText untuk mendapatkan jumlah halaman dalam setiap file PDF. Setiap halaman akan diparse menggunakan iText juga melalui proses multi threading. Proses pembacaan tiap halaman PDF ditunjukkan pada Gambar 3.



Gambar 3. Proses Membaca Dokumen

2. Preprocessing
   Pada proses ini sistem akan melakukan penyiapan atas tiap term yang didapatkan dari langkah pertama. Langkah preprocessing yang dilakukan adalah tokenizing, membuang stopword, dan case transformation. Pada langkah ini tidak dilakukan proses stemming dengan dua alasan, pertama, bahwa proses stemming pada ratusan ribu term yang didapatkan pada tahap preprocessing akan membutuhkan waktu yang sangat lama karena processor pada device android tidak sekuat processor pada perangkat komputer seperti server, atau workstation. Alasan kedua adalah bahwa term-term yang mengalami proses stemming tidak dapat dikembalikan ke bentuk aslinya padahal synset pada WordNet masih menggunakan term dalam bentuk asli bukan yang telah di-stemming. Karena itu term-term tidak di-stemming agar masih dapat dibandingkan juga dengan setiap synset dari WordNet. Proses preprocessing ditunjukkan pada Gambar 4.



Gambar 4. Preprocessing setiap Dokumen

3. Clustering

Pada proses clustering menggunakan Suffix Tree Clustering digunakan beberapa fungsi dan kondisi yang menggambarkan proses pembuatan cluster. Flowchart pembuatan cluster ditunjukkan pada Gambar 5 dan Gambar 6.

Fungsi-fungsi dan kondisi yang digunakan adalah:

Start : proses memberi nilai ID berdasarkan terminasi yang terjadi
P1 : proses peraturan nomer 1 untuk memulai iterasi baru
P2 : proses pembentukan edge baru dari active node
P3 : proses perpindahan kata aktif
P5 : proses perpindahan kata aktif
P7 : proses pembentukan node baru dan inisiasi nilai end baru dari edge tersebut
P8 : proses pembentukan edge dari nilai percabangan
P9 : proses pementukan suffix link
P10 : proses mencatat node terakhir
P11 : proses pembentukan edge dengan nilai finger
P14 : proses perpindahan active edge untuk kata aktif
P15 : proses perpindahan node ke activeEdge.getChildNode()
P16 : proses mencatat posisi proses terakhir
P17 : proses penanda proses STC selesai
P18 : proses mencatat posisi proses terakhir
Np1 : proses peraturan nomer 3 dan 4c
Np2 : proses pembentukan edge baru dari node
Np3 : proses setelah peraturan 3 dan cek peraturan 2
Np4 : proses mencatat posisi proses terakhir
K1 : kondisi jika finger telah mencapai akhir dokumen
K2 : kondisi jika ada kata belum terbentuk
K3 : kondisi jika active node memiliki edge dengan awalan finger
K5 : kondisi jika aktif adalah root
K6 : kondisi jika kata selanjutnya pada kata aktif adalah finger
K8 : kondisi jika ative edge punya child node
K9 : kondisi jika iterasi berjalan lebih dari satu kali
K11 : kondisi jika listword kosong menandakan tidak ada finger yang belum terbentuk
K12 : kondisi jika active node memiliki awalan kata finger

K13 : kondisi jika active edge memiliki childnode
K14 : kondisi jika jumlah proses melebihi threshold stackoverflow
K15 : kondisi jika jumlah proses melebihi threshold stackoverflow
Nk1 : kondisi jika activeNode memiliki kata finger
Nk2 : kondisi jika setelah perpindahan active edge, listword menyentuh index terakhir dari edge baru.
Nk3 : kondisi jika jumlah proses melebihi threshold stackoverflow.

Beberapa masalah yang perlu diantisipasi dalam proses pembuatan cluster adalah, pertama, percabangan terjadi pada edge yang memiliki child node. Masalah ini diselesaikan dengan pemisahan edge dengan childnode dan menggabungkan dengan node baru seperti pada proses k8 dan p6relbreaker. Masalah kedua, kata aktif berada pada akhir dari edge tersebut karena memiliki child node. Masalah ini diselesaikan dengan perpindahan active node dari active node saat ini menjadi activeEdge.getChildNode. Proses perpindahan node terjadi pada proses p4 dan p15.

Pada akhir tahap clustering ini dilakukan proses identifikasi base cluster. Proses identifikasi base cluster tidak dilakukan melalui proses penelusuran pada tiap node dari tree yang telah dibuat di atas. Hal ini disebabkan bahwa proses penelurusan pada tiap node pada perangkat Android akan membutuhkan waktu yang cukup panjang. Pada tahap ini proses merging dilakukan langsung pada daftar node yang diperoleh tanpa melakukan penelurusan ulang pada tree yang telah terbentuk. Hasil base cluster yang terbentuk dengan metode ini memiliki jumlah base cluster dan anggota dokumen yang sama dengan bila melalui penelusuran ulang tree, tetapi frase base cluster yang sedikit berbeda. Perbedaan ini ditunjukkan pada Tabel 2.

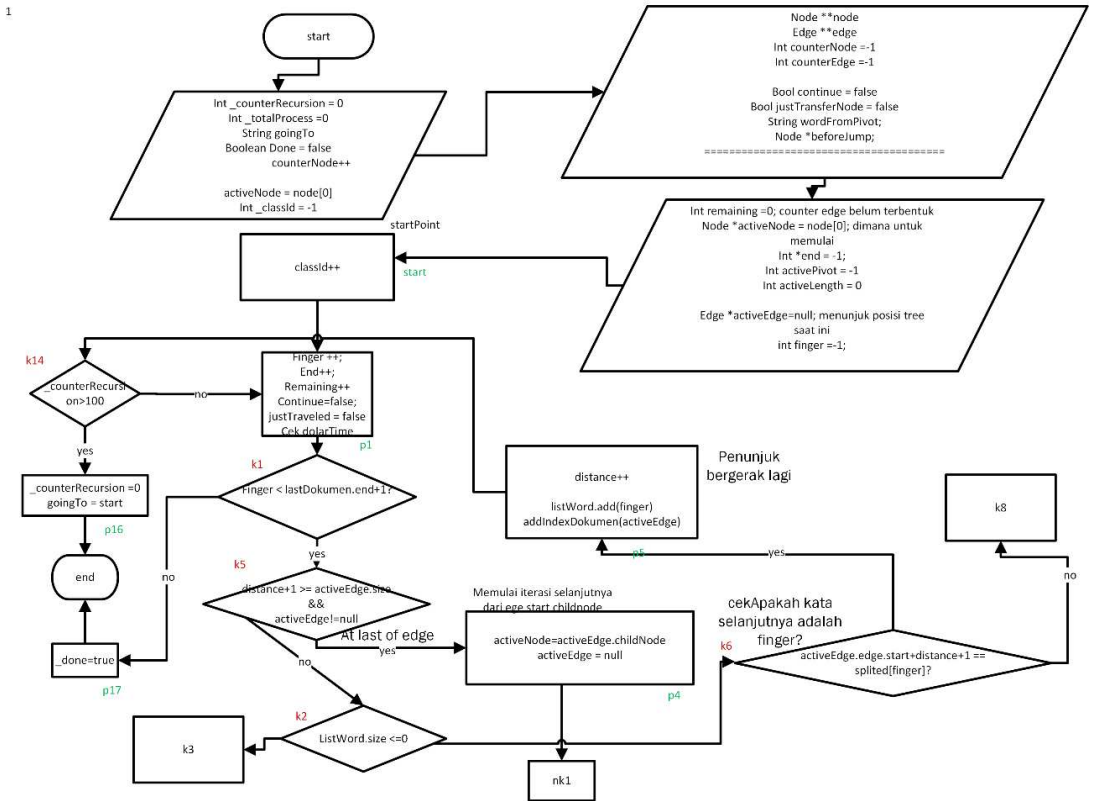
Tabel 2. Perbedaan Dua Metode Pembentukan Base Cluster

Node	***Cluster* Melalui Penelusuran Ulang Tree**	***Cluster* Tanpa Melalui Penelusuran**	**Dokumen**
A	*Cat ate*	*Cat ate*	1,3
B	*Ate*	*Ate*	1,2,3
C	*Cheese*	*Cheese*	1,2
D	*Mouse*	*Mouse*	2,3
E	*Too*	*Too*	2,3
F	*Ate cheese*	*cheese*	1,2

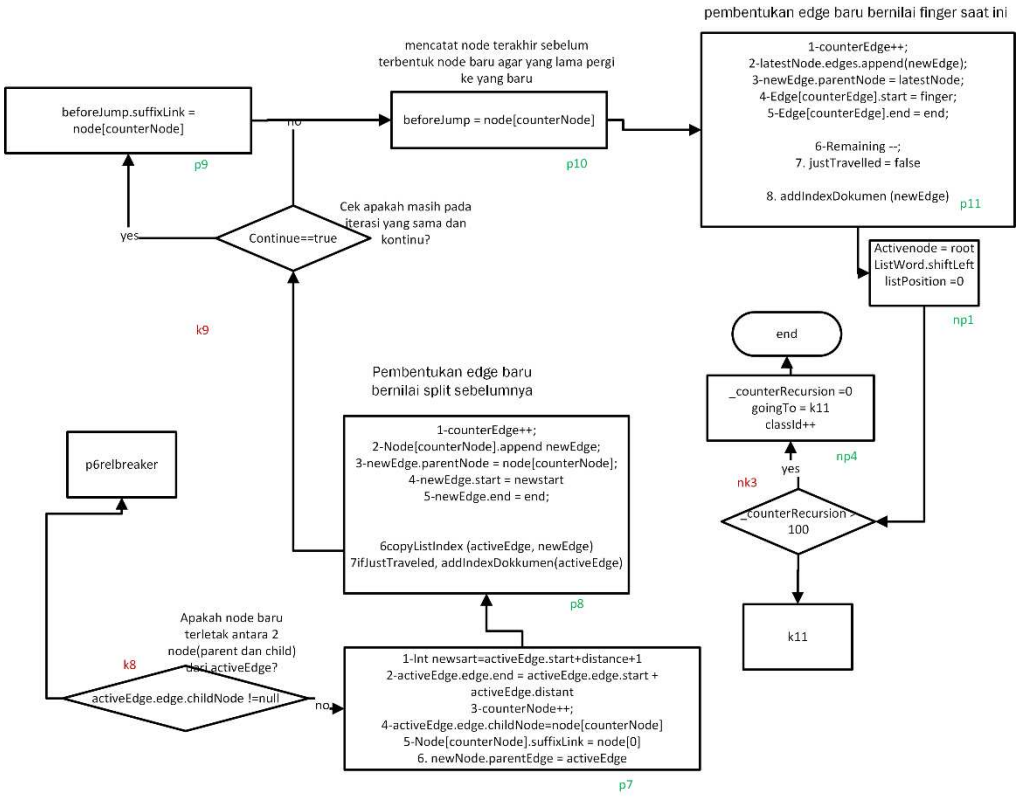
4. Cluster Merging
Pada proses cluster merging dilakukan penyatuan base cluster-base cluster yang dokumen anggota clusternya persis sama. Proses penyatuan base cluster menggunakan algoritma yang diajukan oleh oleh Janruang dan Guha pada Pers. (2). Jumlah maksimum cluster yang didapatkan setelah proses merging ditunjukkan pada Pers. (3).

$$s(n) = \sum_{i=1}^{n} n\ C\ i \qquad (3)$$

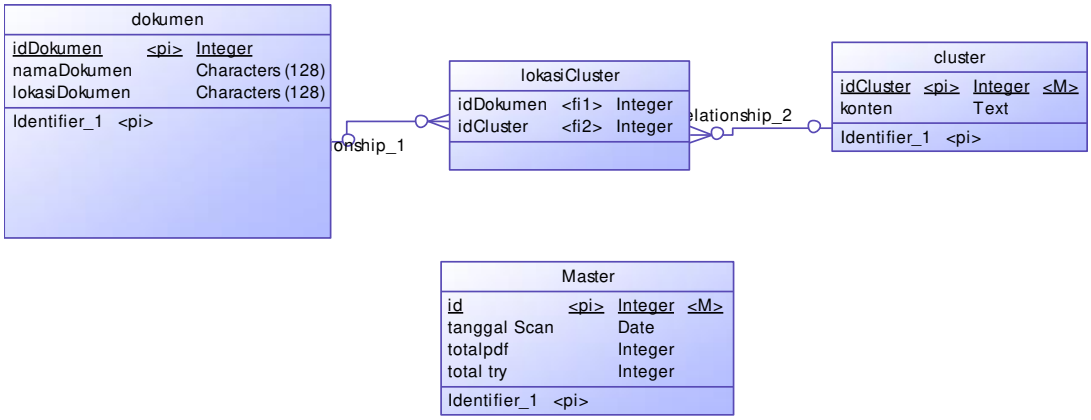
5. Penyimpanan ke Database
Hasil base cluster-base cluster yang teridentifikasi, dokumen anggota cluster, dan frase tiap base cluster disimpan pada database yang ditunjukkan pada Gambar 7.



Gambar 5. Proses Pembuatan Cluster



Gambar 6. Proses Pembuatan Cluster (lanjutan)



Gambar 7. Struktur Database Cluster

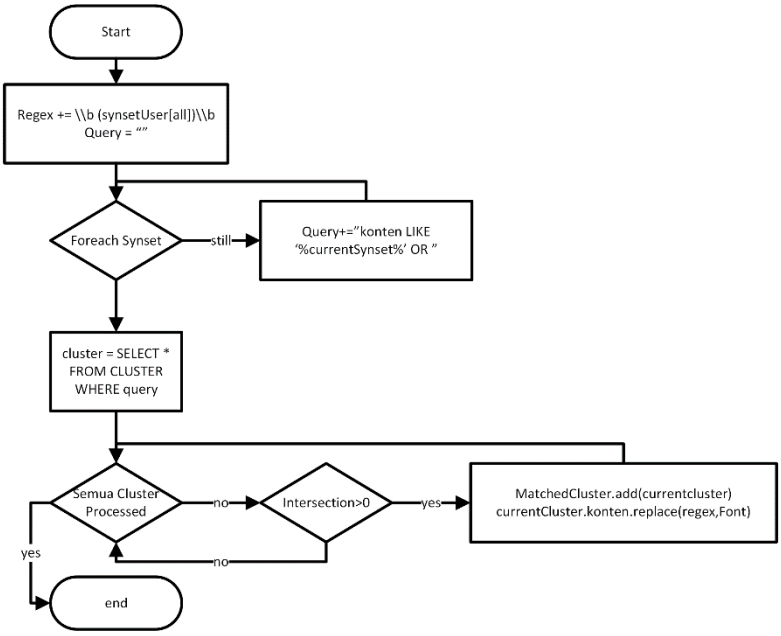
4.2. Proses Searching

Proses searching adalah proses membandingkan kata kunci yang dimasukkan pengguna dengan frase dari base cluster-base cluster yang tersimpan pada database. Base cluster yang frasenya mirip dengan kata kunci pengguna akan ditampilkan sebagai hasil pencarian file. Sebelum dibandingkan kata kunci dari pengguna akan mengalami proses keyword expansion menggunakan WordNet. Proses ekspansi ini dimaksudkan agar proses searching juga dapat menemukan base cluster-base cluster lainnya yang mirip walaupun frase dari base cluster tidak secara langsung mengandung kata kunci yang dicari pengguna.

Bila Km adalah term-term dari frase base cluster, dan Bu adalah synset dari kata kunci yang didapatkan dari WordNet, maka perbandingan frase base cluster dan synset kata kunci pengguna ditunjukkan pada Pers. (4).

$$|Km \cap Bu| > 0 \qquad (4)$$

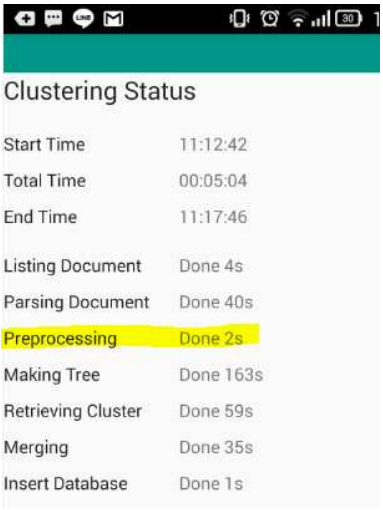
Proses perbandingan antara frase base cluster dan synset kata kunci pengguna dilakukan menggunakan sqlite seperti flowchart yang ditunjukkan pada Gambar 8.



Gambar 8. Flowchart Pencarian Base Cluster yang Mirip dengan Kata Kunci Pengguna

4.3. Hasil Implementasi dan Pengujian

Pengujian performa terhadap waktu dilakukan pada proses indexing meliputi proses listing, preprocessing, hingga penyimpanan datatabase sqlite di Android. Pengujian dilakukan atas 53 dokumen, dengan jumlah term lebih kurang 52.000 kata. Seperti terlihat pada Gambar 9, preprocessing diselesaikan dalam 2 detik. Waktu ini relatif singkat dibandingkan proses pembuatan suffix tree yang membutuhkan waktu 2 menit 43 detik (163 detik).



Gambar 9. Konsumsi Waktu Proses *Preprocessing* Metode Run Aplikasi

Pada proses merging dilakukan pengujian lebih lanjut menggunakan 5 dokumen dengan total 2924 term. Jumlah cluster yang ditemukan pada proses pembuatan cluster adalah 528 cluster. Setelah dilakukan proses merging, maka jumlah cluster yang ditemukan adalah 27 cluster. Proses sebelum dan setelah merging ditunjukkan pada Gambar 10a, dan 10b.

a. Sebelum b. Setelah

Gambar 10. Kondisi Cluster Sebelum dan Setelah Proses Merging

Tabel 3. Hasil Pengujian Proses Searching

No	*Kata Kunci Pengguna*	Jumlah *Cluster yang ditemukan (n)*	Lama Waktu (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 untuk searching menggunakan jumlah dokumen yang lebih besar, yaitu 49 dokumen PDF, dan 52 dokumen dengan format text. Sumber dokumen diambil dari Wikipedia yang kemudian disimpan ke dalam storage dari device Android. Jumlah cluster yang terbentuk adalah 4699 cluster. Waktu yang dibutuhkan dalam rata-rata 10 kali proses indexing untuk jumlah dokumen di atas adalah 686,7 detik. Hasil pengujian proses searching dengan kata kunci yang berbeda-beda ditunjukkan pada Tabel 3.

Hasil tampilan untuk pengujian searching dengan kata kunci “cartoon” ditunjukkan pada Gambar 11. Pada gambar tersebut terlihat bahwa terdapat 2 cluster yang sesuai dengan synset dari kata “cartoon”.

Gambar 11. Hasil Pencarian File pada Android Menggunakan Kata Kunci “Cartoon”

KESIMPULAN

Dari hasil pembuatan dan pengujian prototype dapat disimpulkan bahwa Suffix Tree Clustering dapat menemukan berkas-berkas yang serupa pada perangkat Android berdasarkan isi berkas-berkas tersebut. Penggunaan WordNet sebagai *term expansion* juga dapat membantu dalam

proses penemuan kembali cluster yang berhubungan dengan kata kunci yang dimasukkan pengguna walaupun frase yang mewakili cluster tersebut tidak secara langsung memiliki kata kunci pengguna. Yang diuji adalah berkas-berkas dengan konten berupa PDF stream, dan file dengan konten teks.

DAFTAR PUSTAKA

AlAgha, I., & Nafee, R. 2015. *Investigating the Efficiency of WordNet as Background Knowledge for Document Clustering*. Journal of Engineering Research and Technology *2*(2), pp. 152-158.

Dang, Q., Zhang, J., Lu, Y., & Zhang, K. (2013). *WordNet-Based Suffix Tree Clustering Algorithm*. Proceedings of the 2013 International Conference on Information Science and Computer Applications (ISCA 2013), pp. 66-74. doi:10.2991/isca-13.2013.12

Hussain, A. 2012. *Textual Similarity*. Tesis, Informatics and Mathematical Modelling: Technical University of Denmark.

Janruang, J. & Guha, S. 2011. *Semantic Suffix Tree Clustering*. Makalah disajikan dalam First IRAST International Conference on Data Engineering and Internet Technology (DEIT)

Lhoussain, A.S., Hicham, G., & Abdellah, Y. 2015. *Adaptating The Levenshtein Distance to Contextual Spelling Correction*. International Journal of Computer Science and Applications. 12(1). pp. 127–133.

Marco, A. D., & Navigli, R. 2013. *Clustering and Diversifying Web Search Results with Graph-Based Word Sense Induction*. Computational Linguistics 39(3). pp. 709-754. doi:10.1162/coli_a_00148

Pate, S.D. 2003. *UNIX Filesystems: Evolution, Design, and Implementation*. Indianapolis: Wiley Publishing Inc.

Princeton University. 2010. About WordNet. http://wordnet.princeton.edu diakses 8 Juni 2017.

Wirzenius, L., Oja, J., Stafford, S., & Weeks, A. 2005. Linux System Administrators Guide: Chapter 5. Using Disks and Other Storage Media. http://www.tldp.org/LDP/sag/html/filesystems.html diakses 8 Juni 2017

Zamir, O., & Etzioni, O. 1998. *Web Document Clustering: A Feasibility Demonstration*. Makalah disajikan dalam 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.