TUGAS AKHIR – KI141502

IMPLEMENTASI TESAURUS HIRARKIS SECARA DINAMIS PADA DATA BERITA BERBAHASA INDONESIA MENGGUNAKAN INFORMASI CO-OCCURRENCE

MIFTAHUDDIN AL ANSHORY
NRP. 5112 100 026

Dosen Pembimbing 1
Diana Purwitasari, S.Kom., M.Sc.

Dosen Pembimbing 2
Abdul Munif, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2016

(Halaman ini sengaja dikosongkan)

FINAL PROJECT – KI141502

DYNAMIC IMPLEMENTATION OF HIERARCHICAL THESAURUS ON INDONESIAN NEWS USING CO-OCCURRENCE INFORMATION

MIFTAHUDDIN AL ANSHORY
NRP. 5112 100 026

Supervisor 1
Diana Purwitasari, S.Kom., M.Sc.

Supervisor 2
Abdul Munif, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2016

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

IMPLEMENTASI TESAURUS HIRARKIS SECARA DINAMIS PADA DATA BERITA BERBAHASA INDONESIA MENGGUNAKAN INFORMASI CO-OCCURRENCE

TUGAS AKHIR

Diajukan Untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Sarjana Komputer
pada
Bidang Studi Komputasi Cerdas dan Visualisasi
Program Studi S-1 Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember

Oleh:
MIFTAHUDDIN AL ANSHORY
NRP. 5112100026

Disetujui oleh Pembimbing Tugas Akhir:

1. Diana Purwitasari, S.Kom., M.Sc. .....
   NIP. 19780410200312200[illegible] (Pembimbing 1)

2. Abdul Munif, S.Kom., M.Sc. .....
   NIP. 198608232015041004 (Pembimbing 2)

SURABAYA
JUNI 2016

(Halaman ini sengaja dikosongkan)

IMPLEMENTASI TESAURUS HIRARKIS SECARA DINAMIS PADA DATA BERITA BERBAHASA INDONESIA MENGGUNAKAN INFORMASI CO-OCCURRENCE

Nama	**: Miftahuddin Al Anshory**
NRP	**: 5112100026**
Jurusan	**: Teknik Informatika**
	Fakultas Teknologi Informasi ITS
Dosen Pembimbing I	**: Diana Purwitasari, S.Kom., M.Sc.**
Dosen Pembimbing II	**: Abdul Munif, S.Kom., M.Sc.**

ABSTRAK

Tesaurus dibangun secara dinamis merupakan salah satu penyelesaian dari masalah beban kerja yang dilakukan secara manual untuk menambah atau mengubah informasi kata. Tesaurus hirarkis bertujuan untuk menentukan kata yang memiliki arti umum (hiponim) dan kata yang memiliki arti khusus (hipernim). Tesaurus berbahasa Indonesia seperti kamus Kateglo, masih memiliki kekurangan untuk menampilkan nilai kedekatan yang dimiliki oleh masing-masing kata terhadap kata lain, sedangkan tesaurus yang dibangun diharapkan dapat menampilkan nilai kedekatan tersebut.

Tesaurus dibangun dengan cara melakukan ekstraksi kata dari artikel berita berbahasa Indonesia. Kemudian hasil ekstraksi kata tersebut dibentuk menjadi sebuah informasi co-occurrence dari masing-masing kata. Hasil dari informasi co-occurrence digunakan untuk membentuk vektor kata yang digunakan dalam penghitungan jarak antar kata menggunakan cosine similarity sebelum dilakukannya proses clustering menggunakan hierarchical clustering. Setelah terbentuk cluster,

dilakukan penghitungan Kullback-Leiber divergence pada antar kata yang terdapat pada satu cluster yang sama.

Hasil dari proses pembangunan tesaurus yang dilakukan menggunakan data sebanyak 5.313 xml artikel berita pada proses ekstraksi kata membutuhkan waktu terlama dalam pembangunan tesaurus, yaitu 1.735 menit, sedangkan yang tercepat adalah proses clustering dengan waktu 10 menit. Pencarian yang dilakukan pada tesaurus menggunakan kata "laras" dan kata "senapan" menghasilkan informasi bahwa kata "laras" merupakan kata umum (hiponim), sedangkan kata "senapan" merupakan kata khusus (hipernim). Informasi tersebut didapatkan dari nilai Kullback-Leiber divergence kata "laras" terhadap "senapan" lebih dekat dengan nilai 0, yaitu -0,026605, sedangkan kata "senapan" terhadap laras bernilai 0,075382.

Kata kunci: Tesaurus, Informasi Co-occurrence, Hirarkis, Dinamis, Hierarchical Clustering, Kullback-Leiber divergence.

DYNAMIC IMPLEMENTATION OF HIERARCHICAL THESAURUS ON INDONESIAN NEWS USING CO-OCCURRENCE INFORMATION

Name	**: Miftahuddin Al Anshory**
NRP	**: 5112100026**
Department	**: Department of Informatics Faculty of Information Technology ITS**
Supervisor I	**: Diana Purwitasari, S.Kom., M.Sc.**
Supervisor II	**: Abdul Munif, S.Kom., M.Sc.**

ABSTRACT

Thesaurus built dynamically is one of the ways to solve a problem about workload for built it manually for adding or changing word information. While hierarchical intends to define common words (hyponym) and specific words (hypernym). Indonesian Thesaurus such as Kateglo still had deficiency to show similarity value between each words, whereas thesaurus we built is expected to show the value of similarity.

Thesaurus is built by extracting words from Indonesian language news articles. Then the result from extraction process formed into a co-occurrence information of each word. Then, the results of the co-occurrence information used to form a vector of words that used in the calculation of distances between words using the cosine similarity before the clustering process using hierarchical clustering. Once formed cluster, do the calculation with the Kullback-Leiber divergence between words contained in the same cluster.

The result of thesaurus that built by using 5.313 xml news articles, process that took long time to finish was the word extraction process with 1.735 minutes, while the fasterst was the

clustering process with 10 minutes. Searches performed on the thesaurus using the word "laras" and the word "senapan" that produce the information that the word "laras" is a common word (hyponym), while the word "senapan" is a specific word (hypernym). That kind of information gained from the value of Kullback-Leiber divergence between "laras" and "senapan" is closer to 0, -0,026605, while the value between "senapan" and laras is 0,075382.

Key words: Thesaurus, Co-occurrence Information, Hierarchical, Dynamic, Hierarchical Clustering, Kullback-Leiber divergence.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

(Halaman ini sengaja dikosongkan)

DAFTAR KODE SUMBER

BAB I
PENDAHULUAN

1.1 Latar Belakang

Tesaurus merupakan acuan dalam pemrosesan bahasa natural, seperti penerapannya pada klasifikasi dokumen menggunakan *semantic contents* [1], ekspansi *query* dalam temu kembali informasi, disambiguasi arti kata [2] dan lain sebagainya. Tesaurus adalah kamus yang berisi daftar kata-kata yang dikelompokkan berdasarkan kemiripan arti (sinonim) [3].

Tesaurus seperti WordNet [4] pada umumnya dibuat secara manual, sehingga akurasi kemiripan arti kata memiliki nilai akurasi yang tinggi. Terdapat kesulitan untuk memperbaruinya ke dalam skala yang lebih besar karena proses pembaruan dilakukan secara manual sehingga membutuhkan waktu yang lama. Sedangkan pada tesaurus berbahasa Indonesia, Kateglo [5], masih terdapat kekurangan karena tesaurus tersebut hanya menampilkan keluaran berupa daftar kata, tanpa menampilkan nilai kemiripan antar kata tersebut.

Beberapa riset untuk membangun tesaurus seacara otomatis telah dilakukan. Misalnya, menggunakan metode pembangkitan tesaurus dengan algoritma *neural network* [6], teknik pengelompokan menggunakan relasi *co-occurrence* [7] [8] [9], dan beberapa riset yang lain. Tetapi untuk membagi kata baru ke dalam *node* klasifikasi baru dan konstruksi tesaurus yang statis menyebabkan perubahan seperti penambahan kata baru membutuhkan biaya yang besar.

Informasi *co-occurrence* adalah informasi nilai frekuensi kemunculan dua kata pada waktu bersamaan atau muncul dengan posisi bersebelahan pada suatu dokumen [10]. Nilai frekuensi didapatkan dengan cara menghitung berapa kali dua kata tersebut muncul dalam satu atau pada banyak dokumen. Jika terdapat pada banyak dokumen, maka nilai frekuensi akan diakumulasikan dari dokumen satu dengan dokumen yang lainnya.

Hirarki kata pada tesaurus digunakan untuk menentukan kata-kata yang merupakan hiponim atau hipernim. Semakin tinggi posisi suatu kata pada struktur hirarki, maka kata tersebut semakin umum. Selain digunakan untuk menentukan posisi kata, hirarki juga digunakan untuk menghitung nilai kemiripan menggunakan metode Kullback-Leiber *divergence* [11], metode untuk menghitung kemiripan kata menggunakan hirarki kata.

Konstruksi tersaurus secara dinamis bertujuan untuk otomatisasi penambahan kata pada tesaurus. Terdapat sebuah algoritma yang dapat digunakan untuk melakukan perubahan pada struktur hirarki pada tesaurus sehingga membuat pembaruan tesaurus tidak lagi dilakukan secara manual.

Sumber data tesaurus yang akan dibangun bersumber dari berita, karena pada berita terdapat berbagai macam kata yang dapat diekstraksi untuk diolah menjadi tesaurus. Berita selalu memiliki informasi baru tiap harinya, sehingga hal tersebut dapat berpengaruh pada konten tesaurus. Media berita yang digunakan adalah media *online* karena proses ekstraksi kata dapat dilakukan dengan mudah. Salah satu media *online* yang berupa situs portal berita yaitu situs web Kompas, situs portal berita di Indonesia, sehingga tesaurus yang akan dibuat menggunakan bahasa Indonesia.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat pada tugas akhir ini adalah sebagai berikut:

1. Bagaimana cara untuk membentuk informasi *co-occurrence* dari hasil ekstraksi kata artikel berita berbahasa Indonesia?
2. Bagaimana cara untuk membangun tesaurus hirarkis secara dinamis dengan memanfaatkan nilai hasil pembentukan informasi *co-occurrence* kata?
3. Bagaimana cara untuk melakukan evaluasi pada tesaurus yang dibangun?

1.3 Batasan Masalah

Batasan masalah dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Domain kata hanya berasal dari ekstraksi artikel berita dari situs web Kompas, sehingga tesaurus yang dibangun menggunakan bahasa Indonesia.
2. Keluaran dari tesaurus hanya berupa daftar kata dan nilai kemiripannya, tanpa menampilkan definisi dari kata tersebut.

1.4 Tujuan

Tujuan dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Membangun tesaurus untuk dapat menampilkan kata-kata mirip beserta nilai kemiripannya.
2. Membangun tesaurus yang memiliki hirarki antar kata untuk mempertimbangkan nilai kemiripan antar kata dan menentukan kata yang termasuk hiponim atau hipernim.
3. Membuat tesaurus yang dapat beradaptasi dengan penambahan kata baru.

1.5 Manfaat

Manfaat dari tugas akhir ini adalah tesaurus yang dibangun diharapkan dapat digunakan sebagai rujukan mencari nilai kemiripan dari suatu kata dan menampilkan daftar kata yang mirip dengan suatu kata tertentu. Kemudian dengan membangun tesaurus yang dinamis, tidak lagi dibutuhkan proses manual untuk melakukan penambahan kata baru pada tesaurus, sehingga dapat mengurangi beban kerja dan waktu.

1.6 Metodologi

1. Studi Literatur
   Pada tahap ini dilakukan pembelajaran dan pemahaman tentang metode-metode dan pustaka-pustaka yang digunakan dalam proses pengerjaan tugas akhir pada

tahap implementasi sistem. Metode-metode dan pustaka-pustaka tersebut adalah sebagai berikut:

a. Metode pembentukan informasi *co-occurrence*.
b. Metode penghitungan jarak antar kata, *cosine similarity*.
c. Metode pengelompokan kata, *hierarchical clustering*.
d. Metode penghitungan kemiripan, Kullback-Leiber *divergence*.
e. Kamus bahasa Indonesia, Kateglo.
f. Pustaka pembangkitan *array*, NumPy.
g. Pustaka *hierarchical clustering*, SciPy.
h. Pustaka penampilan data hasil *clustering*, *data-driven documents*.

2. Analisis dan perancangan sistem
   Pada tahap ini dilakukan analisis permasalahan yang diangkat dan perancangan sistem berdasarkan studi literatur yang telah dilakukan. Tahap ini merancang alur implementasi yang akan dilakukan, seperti merancang modul-modul yang dibuat, melakukan perancangan untuk menampilkan hasil *clustering*, perancangan basis data dan perancangan antar muka. Tesaurus yang dibangun memiliki dua modul utama, yaitu:
   a. Modul praproses data, yaitu modul yang digunakan untuk melakukan ekstraksi kata pada setiap artikel berita dan membentuk informasi *co-occurrence*.
   b. Modul pembentukan *cluster* kata, yaitu modul untuk melakukan penghitungan *cosine similarity*, mengelompokkan kata hasil ekstraksi menjadi menjadi beberapa *cluster* dan untuk menghitung nilai kemiripan antar kata pada suatu *cluster*.

3. Implementasi sistem

Pada tahap ini dilakukan pembangunan tesaurus menggunakan informasi *co-occurrence*, metode-metode dan pustaka-pustaka yang telah dipelajari menjadi sistem yang dapat digunakan. Bahasa pemrograman yang digunakan adalah PHP dan Python dengan menggunakan basis data MySQL untuk menyimpan data tesaurus dan Apache sebagai *web server*.

4. Pengujian dan evaluasi
   Pada tahap ini dilakukan pengujian dan evaluasi terhadap tesaurus yang telah dibuat dengan beberapa skenario pengujian untuk mengetahui hasil keluaran dan kebenaran yang diberikan oleh sistem.

5. Penyusunan buku tugas akhir
   Pada tahap ini dilakukan penyusunan buku tugas akhir sebagai dokumentasi pengerjaan tugas akhir dari keseluruhan proses pengerjaan. Mencakup tinjauan pustaka pengerjaan tugas akhir, analisis dan perancangan, implementasi, uji coba dan evaluasi, kesimpulan dan saran terhadap sistem yang telah dibangun.

1.7 Sistematika Penulisan Laporan Tugas Akhir

Buku tugas akhir ini disusun dengan tujuan untuk memberikan gambaran tentang pengerjaan tugas akhir yang telah dilakukan. Buku tugas akhir ini terbagi menjadi enam bab, yaitu:

Bab I Pendahuluan

Bab yang berisi mengenai latar belakang, rumusan masalah, batasan masalah, tujuan dan manfaat pembuatan tugas akhir, serta metodologi dan sistematika penulisan buku tugas akhir.

Bab II Tinjauan Pustaka

Bab yang berisi penjelasan metode-metode dan pustaka-pustaka yang digunakan sebagai dasar dan penunjang pengerjaan tugas akhir. Metode-metode yang digunakan meliputi beberapa algoritma, yaitu algoritma pemrosesan teks, algoritma pengelompokan kata dan algoritma penghitungan kemiripan. Sedangkan pustaka yang digunakan yaitu pustaka pembangkitan *array*, pustaka pengelompokan kata dan pustaka untuk menampilkan hasil pengelompokan kata.

Bab III Analisis dan Perancangan Sistem

Bab yang berisi tentang analisis masalah yang diangkat, mendefinisikan alur proses implementasi, merancang modul-modul, algoritma, tahapan sistem hingga terbentuk suatu rancangan sistem yang siap dibangun.

Bab IV Implementasi Sistem

Bab yang berisi implementasi perancangan sistem yang telah dibuat berupa algoritma dan tahapan pada bab perancangan sehingga menjadi sistem yang dilakukan uji coba dan evaluasi.

Bab V Uji Coba dan Evaluasi

Bab yang berisi skenario uji coba, hasil uji coba, analisis dan evaluasi yang dilakukan terhadap sistem sehingga dapat diketahui performa dan kebenaran yang dihasilkan oleh sistem.

Bab VI Kesimpulan dan Saran

Bab yang berisi kesimpulan dari hasil uji coba terhadap sistem yang dibuat dan saran untuk pengembangan sistem pada tugas akhir ini untuk kedepannya.

BAB II
TINJAUAN PUSTAKA

Bab ini membahas metode dan pustaka yang digunakan dalam implementasi tesaurus. Penjelasan ini bertujuan untuk memberikan gambaran secara umum terhadap tesaurus yang dibangun dan menjadi acuan dalam proses pembangunan dan pengembangan tesaurus.

2.1 Algoritma Pemrosesan Teks

Algoritma pemrosesan teks digunakan untuk melakukan ekstraksi kata pada kumpulan artikel berita dan pembentukan informasi *co-occurrence* dari kata hasil ekstraksi. Algoritma ini bisa disebut sebagai tahapan praproses data.

2.1.1 Tokenisasi

Tokenisasi adalah suatu proses untuk membagi suatu teks berupa kalimat atau paragraf menjadi unit-unit kecil berupa kumpulan kata atau token [12]. Sebagai contoh pada kalimat "Mengurangi penerbitan obligasi yang dilakukan oleh pemerintah" menghasilkan tujuh token, yaitu: "Mengurangi", "penerbitan", "obligasi", "yang", "dilakukan", "oleh", "pemerintah". Pada proses tokenisasi, yang menjadi acuan pemisah antar kata adalah tanda baca dan spasi.

2.1.2 Penghapusan *Stopword*

Setelah dilakukan proses tokenisasi pada artikel berita, proses yang dilakukan selanjutnya yaitu penghapusan *stopword*. *Stopword* adalah kata-kata sering muncul pada suatu dokumen yang tidak memberikan informasi penting, seperti kata penghubung dan kata ganti orang [12]. Penghapusan *stopword* ini bertujuan agar kata-kata yang digunakan hanya kata-kata yang memiliki arti penting dan memberikan suatu informasi.

2.1.3 *Stemming*

S*temming* adalah proses untuk mengembalikan bentuk kata menjadi bentuk dasarnya [12]. Membuang awalan, sisipan atau akhiran hingga menjadi kata dasar yang sesuai dengan bahasa Indonesia yang baik dan benar. Proses ini menggunakan pencocokan kata-kata artikel dengan kata pada Kateglo [5]. Jika suatu kata terdapat pada Kateglo, berarti kata tersebut terdapat pada kaidah bahasa Indonesia.

2.1.4 Pembentukan Informasi *Co-occurrence*

Proses yang terakhir adalah pembentukan informasi *co-occurrence* kata. Elemen penyusun informasi *co-occurrence* adalah kata yang muncul terlebih dahulu, *registered word*, misalnya kata *X*, kata *co-occurrence Y* kata yang selanjutnya muncul dan frekuensi (*f*) dari kemunculan dua kata tersebut. Jika pasangan kata tersebut muncul pada banyak artikel, maka frekuensi akan diakumulasikan. Sebagai contoh, jika terdapat tiga kata berikut sesuai dengan urutan kemunculannya "tanam", "modal" dan "perusahaan", maka informasi *co-occurrence* yang dapat dibentuk dari kata-kata tersebut sesuai dengan persamaan (*X*, *Y*, *f*) adalah (tanam, modal, 1) dan (modal, perusahaan, 1). Proses ini dilakukan hingga artikel diproses secara keseluruhan dan masing-masing kata pada kumpulan artikel berita tersebut mempunyai informasi *co-occurrence*.

2.2 Algoritma Pengelompokan Kata

Sebelum melakukan pengelompokan kata, dilakukan pembentukan vektor atribut yaitu vektor kata dan vektor *node* dengan menggunakan informasi *co-occurrence*. Vektor kata terdiri dari kata *co-occurrence* dan frekuensi *co-occurrence*. Sedangkan vektor *node* berisi kumpulan vektor kata dari *registered word* yang besangkutan dengan *node* tersebut. Vektor *node* juga dapat disebut sebagai *cluster*. Contoh dari vektor kata adalah sebagai berikut:

$$WV_X = \begin{bmatrix} k_{X_1}[f_{X_1}] \\ k_{X_2}[f_{X_2}] \\ k_{X_3}[f_{X_3}] \\ \vdots \\ k_{X_n}[f_{X_n}] \end{bmatrix} \quad (2.1)$$

WV_X (*Word Vector X*) adalah vektor kata berupa matriks dari kata *X* yang memiliki kata *co-occurrence k* sebanyak *n* dan frekuensi *co-occurrence f* sebanyak *n*. Sedangkan contoh vektor *node* adalah sebagai berikut:

$$NV_i = \begin{bmatrix} WV_1 \\ WV_2 \\ WV_3 \\ \vdots \\ WV_m \end{bmatrix} \quad (2.2)$$

NV_i adalah vektor *node* dengan indeks *i* berupa matriks yang berisi vektor kata *WV* sebanyak *m*.

Proses *clustering* dilakukan terhadap kata-kata hasil ekstraksi berdasarkan nilai dari penghitungan jarak antar kata. Metode untuk menghitung nilai jarak antar kata yang telah berbentuk vektor adalah *cosine similarity* [13]. Sedangkan metode yang digunakan untuk melakukan *clustering* adalah metode *hierarchical clustering* [12] dengan pendekatan *agglomerative*. Pengelompokan dilakukan dengan cara membandingkan nilai jarak antar vektor kata dari penghitungan *cosine similarity*, lalu menggabungkan dua vektor kata jika terdapat nilai jarak terdekat hingga semua menjadi sebuah kelompok besar atau hingga pada kondisi tertentu yang diinginkan.

Rumus penghitungan *cosine similarity* dari dua vektor kata ditunjukkan pada persamaan berikut:

$$D(WV_A, WV_B) = \frac{\sum_{i=1}^{t}(f_{A_i} \cdot f_{B_i})}{\sqrt{\sum_{i=1}^{t} f_{A_i}{}^2}\sqrt{\sum_{i=1}^{t} f_{B_i}{}^2}} \quad (2.3)$$

Distance atau jarak antara vektor kata *A*, WV_A dan vektor kata *B*, WV_B dihitung dengan menggunakan nilai *dot product* dan *euclidean distance* dari vektor kata A dan vektor kata B. Variabel t menunjukkan jumlah kesamaan kata *co-occurrence* antara kata *A* dan kata *B*. Sedangkan f_{A_i} dan f_{B_i} adalah nilai frekuensi *co-occurrence* indeks ke *i* dari masing-masing kata *A* dan kata *B*. Jika pada WV_A, kata *A* memiliki *n* kata *co-occurrence*, maka penghitungan jarak dilakukan sebanyak *n* kali antara kata *A* dengan masing-masing kata *co-occurrence*.

Hasil dari penghitungan *cosine similarity* digunakan sebagai nilai masukan untuk melakukan pengelompokan atau *clustering*. *Clustering* dilakukan dengan menggabungkan vektor kata yang memiliki jarak terdekat hingga semua menjadi sebuah kelompok besar atau hingga pada kondisi tertentu yang diinginkan. Hasil dari *clustering* hirarkis digambarkan dalam *dendogram* [12], yaitu grafik yang menunjukkan penggabungan vektor kata berdasarkan urutan penggabungan. Sebagai contoh, terdapat matriks yang menunjukkan nilai jarak antara vektor kata WV_A, hingga vektor kata WV_F.

	WV_A	WV_B	WV_C	WV_D	WV_E	WV_F
WV_A	0					
WV_B	0,71	0				
WV_C	5,66	4,95	0			
WV_D	3,61	2,92	2,24	0		
WV_E	4,24	3,54	1,41	1,00	0	
WV_F	3,20	2,50	2,50	0,50	1,12	0

Jarak terdekat berdasarkan matriks di atas adalah jarak antara vektor kata WV_D dan vektor kata WV_F yaitu 0,5. Kedua vektor kata tersebut digabungkan dengan hasil sebagai berikut:

	WV_A	WV_B	WV_C	WV_D,WV_F	WV_E
WV_A	0				
WV_B	0,71	0			
WV_C	5,66	4,95	0		
WV_D,WV_F	**3,20**	**2,50**	**2,24**	0	
WV_E	4,24	3,54	1,41	**1,00**	0

Setelah dilakukan penggabungan antara vektor kata WV_D dan vektor kata WV_F, terjadi perubahan nilai pada jarak antara vektor kata hasil penggabungan dengan vektor kata lain. Cara mencari nilai perubahan jarak tersebut adalah sebagai berikut:

$$\begin{aligned} D\big((WV_D, WV_F), WV_A\big) &= \min\big(D(WV_D, WV_A), D(WV_F, WV_A)\big) \\ D\big((WV_D, WV_F), WV_A\big) &= \min(3{,}61, 3{,}20) \\ D\big((WV_D, WV_F), WV_A\big) &= 3{,}20 \end{aligned} \quad (2.4)$$

Mencari jarak minimal antara vektor kata yang digabung dengan vektor kata yang besangkutan, yaitu jarak minimal dari vektor kata WV_D terhadap WV_A dan WV_F terhadap WV_A. Sehingga jarak minimal tersebut yang akan menjadi jarak baru terhadap *cluster-cluster* yang lain.

Berdasarkan contoh di atas, vektor kata WV_D digabung dengan vektor kata WV_F karena memiliki jarak terdekat. *Clustering* dilakukan hingga menjadi satu kelompok besar atau pada kondisi tertentu yang diinginkan untuk jumlah *cluster* yang terbentuk. *Cluster* direpresentasikan dalam vektor *node* dengan jumlah vektor kata sebanyak *m* berdasarkan hasil *clustering*.

2.3 Algoritma Penghitungan Kemiripan

Proses selanjutnya adalah menghitung kemiripan antar kata yang terdapat pada *cluster-cluster* hasil dari proses *clustering* untuk memperoleh nilai kemiripan kata dan posisi hirarki kata. Penghitungan tersebut dilakukan menggunakan metode Kullback-Leiber *divergence*, yaitu menghitung kemiripan antar kata dengan mempertimbangkan hirarki antar kata tersebut [11]. Metode ini tidak memiliki bentuk simetri, artinya jarak antara kata *A* dan kata *B* tidak sama dengan jarak kata *B* dan kata *A*. Sehingga dapat ditentukan kata yang memiliki posisi lebih tinggi, kata yang lebih rendah dan nilai kemiripan dari dua kata tersebut.

Kullback-Leiber *divergence* menghitung kemiripan dengan memanfaatkan nilai ekspektasi *entropy loss* antara kata *A* dan kata *B* menggunakan distribusi probabilitas. Nilai probabilitas P_{A_i} dari kata *A* dengan indeks *i* dan P_{B_i} dari kata *B* dengan indeks *i* ditunjukkan dengan persamaan berikut:

$$P_{A_i} = \frac{f_{A_i}}{\sum_{i=1}^{j} f_{A_i}} \tag{2.5}$$

$$P_{B_i} = \frac{f_{B_i}}{\sum_{i=1}^{j} f_{B_i}} \tag{2.6}$$

Nilai t adalah jumlah kesamaan kata *co-occurrence* dari kata *A* dan kata *B*. Nilai *j* adalah banyaknya kata *co-occurrence*. Sedangkan f_{A_i} dan f_{B_i} adalah nilai frekuensi *co-occurrence* indeks ke *i* dari masing-masing kata *A* dan kata *B*. Kata *A* dan kata *B* telah direpresentasikan dalam bentuk vektor kata WV_A dan WV_B. Sehingga nilai Kullback-Leiber *divergence*, $KL(WV_A, WV_B)$ yaitu:

$$KL(WV_A, WV_B) = \sum_{i=1}^{t} P_{A_i} \log \frac{P_{A_i}}{P_{B_i}} \tag{2.7}$$

Didefinisikan beberapa pengecualian sebagai berikut:

$$P_{A_i} \log \frac{P_{A_i}}{P_{B_i}} = \begin{cases} 0 & \text{jika } P_{A_i} = 0 \\ P_{A_i}(\log P_{A_i}) & \text{jika } P_{B_i} = 0 \end{cases} \quad (2.8)$$

Jika hasil dari penghitungan $KL(WV_A, WV_B)$ bernilai kecil atau mendekati 0, berarti kata *A* adalah kata yang lebih umum dari kata *B*. Sebaliknya, jika hasil penghitungan bernilai besar atau menjauhi 0, maka kata *A* adalah kata yang lebih khusus dari kata *B*.

2.4 Kamus Bahasa Kateglo

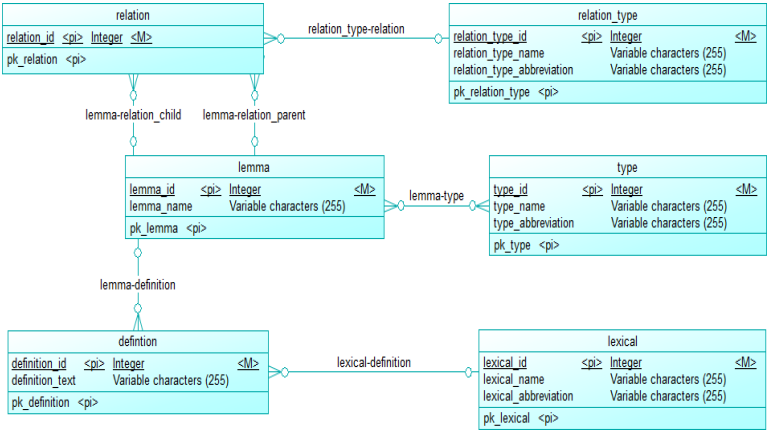
Kateglo adalah sebuah kamus, tesaurus dan glosarium bahasa Indonesia yang dikembangkan oleh Ivan Lanin, Romi Hardiyanto dan Arthur Purnama [14]. Kateglo digunakan untuk mencari definisi, antonim, sinonim, kata bentukan, glosarium terkait suatu kata atau frasa. Setiap kata yang terdapat pada Kateglo memiliki leksikal, yaitu kelas kata yang digunakan untuk penggolongan kata dalam satuan bahasa berdasarkan kategori bentuk, fungsi, dan makna dalam sistem gramatikal. Contoh dari kata dan leksikal kata dijelaskan sebagai berikut:

1. Adjektiva
   Adjektiva (kata sifat) adalah kata yang digunakan untuk memberi sifat, menambah suatu makna pada suatu kata benda atau kata ganti. Contoh dari kata adjektiva adalah keras, jauh, dan kaya.
2. Adverbia
   Adverbia (kata keterangan) adalah kata yang digunakan untuk memberikan informasi lebih banyak tentang kata kerja, kata keterangan yang lain, atau keseluruhan kalimat. Contoh dari kata adverbia adalah cuman, amat, sangat dan konon.
3. Nomina
   Nomina (kata benda) adalah kata atau kelompok kata yang menyatakan suatu nama. Contoh dari kata nomina adalah mesin, mikrofon, minyak dan minuman.
4. Numeralia

Numeralia adalah kata atau frasa yang menunjukkan bilangan atau kuantitas. Contoh dari kata numeralia adalah berenam, bertiga, delapan dan miliar.

5. Pronomina
   Pronomina (kata ganti) adalah kata yang digunakan sebagai kata benda atau frase kata benda. Contoh dari kata pronomina adalah kau, kaulah, kamu dan saya.
6. Verba
   Verba (kata kerja) adalah kata yang digunakan untuk menyatakan suatu perbuatan, kejadian, peristiwa, eksistensi, pengalaman, keadaan antara dua benda. Contoh dari kata verba adalah antre, alur, gagal dan goreng.

Pemanfaatan Kateglo pada tugas akhir ini adalah penggunaannya dalam proses *stemming*. Setiap kata pada artikel berita dicocokan dan diambil leksikal katanya, jika kata tersebut terdapat pada Kateglo, berarti kata tersebut sesuai dengan kaidah bahasa Indonesia. Pemanfaatan leksikal kata pada pembangunan tesaurus adalah untuk membatasi kata yang diproses tesaurus. Penjelasan lebih lanjut akan dijelaskan pada Bab III.



Gambar 2.4.1 *Conceptual Data Model* Kateglo

2.5 Pustaka Pembangkitan *Array*

NumPy adalah pustaka bahasa pemrograman Python yang digunakan untuk mempermudah perhitungan matematika. Salah satu fungsionalitas yang dimiliki oleh NumPy adalah pembangkitan *N-dimensional array* [15], fungsi tersebut adalah ***NumPy.zeros()*** dan ***NumPy.ones()***. ***NumPy.zeros()*** berfungsi untuk melakukan pembangkitan *N-dimensional array* berisi angka 0, sedangkan ***NumPy.ones()*** berfungsi untuk melakukan pembangkitan *N-dimensional array* berisi angka 1.

Pustaka ini menjadi salah satu pustaka utama dalam pembangunan tesaurus pada tugas akhir ini karena kemampuannya untuk melakukan pembangkitan *array* $N \times N$ elemen mengacu pada banyaknya kata yang digunakan dalam proses *clustering*. *N-dimensional array* berfungsi sebagai matriks jarak hasil penghitungan *cosine similarity* yang digunakan untuk melakukan *hierarchical clustering*.

2.6 Pustaka *Hierarchical Clustering*

Pustaka *hierarchical clustering* yang digunakan pada bahasa pemrograman Python adalah SciPy. Pustaka tersebut memiliki modul untuk melakukan *hierarchical clustering*, yaitu modul ***scipy.cluster.hierarchy***.

SciPy adalah sebuah pustaka bahasa pemrograman Python yang dapat digunakan untuk penghitungan *scientific* [16], contohnya adalah penghitungan integral, differensial numerik, interpolasi dan lain sebagainya.

Fungsi pada modul ***scipy.cluster.hierarchy*** yang digunakan untuk melakukan *hierarchical clustering* pendekatan *agglomerative* adalah:

```
linkage(Y, method='single', metric='euclidean')
```

Penjelasan beberapa parameter fungsi ***linkage()*** adalah sebagai berikut:

1. *Y* adalah variabel yang menampung hasil pembangkitan matriks jarak penghitungan *cosine similarity*.
2. *Method* adalah parameter untuk menentukan metode penghitungan jarak antar *cluster* (*linkage*) yang digunakan. Terdapat beberapa metode *linkage*, yaitu *single*, *complete*, *average*, *weighted* dan *centroid*. Metode yang akan digunakan pada tugas akhir ini adalah metode *single*, dengan persamaan berikut:

$$d(u, v) = \min(dist(u[i], v[j])) \quad (2.9)$$

   Menghitung jarak (*d*) minimal untuk semua titik *i* pada *cluster u* dan semua titik *j* pada *cluster v*. metode ini juga dikenal sebagai metode *Nearest Point Algorithm*.
3. *Metric* adalah parameter yang digunakan untuk membentuk matriks jarak menjadi lebih ringkas atau *condensed matrix*. Parameter *metric* akan memanggil fungsi penghitungan jarak, ***pdist()***. Metode ***pdist()*** yang digunakan mengacu pada masukan pada parameter *metric*. Terdapat beberapa metode untuk penghitungan fungsi ***pdist()***, yaitu *euclidean*, *cosine*, *jaccard*, *correlation* dan *dice*. Metode yang digunakan pada tugas akhir ini adalah *euclidean*, menghitung jarak antara *m* titik menggunakan *euclidean distance* sebagai matriks jarak antara titik-titik tersebut.

Hasil keluaran dari pustaka *hierarchical clustering* adalah matriks berisi titik-titik yang digabungkan dan jarak di antara titik-titik tersebut.

Proses *clustering* yang dilakukan selanjutnya adalah menggunakan pustaka SciPy untuk melakukan pembagian *cluster* sehingga terbentuk beberapa *cluster* dari satu *cluster* besar. Fungsi yang digunakan membagi *cluster* adalah ***cut_tree()***. Parameter-parameter yang perlu digunakan pada fungsi ***cut_tree()*** adalah sebagai berikut:

1. *Z* adalah variabel yang menampung matriks hasil *clustering*.
2. *n_clusters* adalah parameter opsional yang digunakan untuk menentukan jumlah *cluster* yang ingin dibentuk.
3. *height* adalah parameter opsional yang digunakan untuk menentukan batas pembagian cluster berdasarkan nilai jarak yang ada pada variabel *Z*.

Jika parameter *n_clusters* atau *height* tidak digunakan, maka hasil keluaran dari fungsi ***cut_tree()*** adalah sebuah *array* yang mempelihatkan keanggotaan *cluster* pada setiap langkah *agglomerative*. Sedangkan jika salah satu parameter tersebut digunakan, maka hasil keluaran fungsi ***cut_tree()*** memperlihatkan keanggotaan *cluster* sesuai dengan parameter yang digunakan.

2.7 Pustaka Data-Driven Documents

Data-Driven Documents atau D3.js adalah pustaka bahasa JavaSript untuk memanipulasi dokumen berdasarkan data pada tampilan situs web. D3.js memungkinkan untuk mengikat suatu data pada Document Object Model (DOM) dan menerapkan transformasi data pada dokumen. Contohnya adalah menghasilkan sebuah tabel pada suatu tampilan HTML dari kumpulan angka atau menggunakan data yang sama untuk membuat grafik batang interaktif dengan berbagai macam transisi pada interaksi yang dilakukan pengguna [17].

Tugas akhir ini menggunakan salah satu contoh penerapan D3.js yaitu Zoomable Circle Packing. Zoomable Circle Packing menampilkan data kelompok-kelompok berupa lingkaran yang di dalamnya terdapat detail anggota masing-masing kelompok.

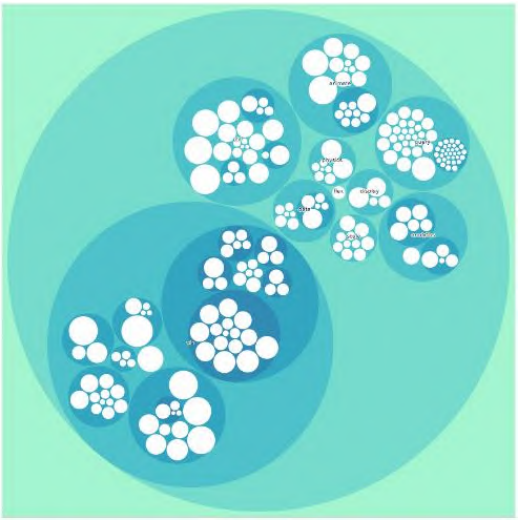
Data yang ditampilkan berasal dari berkas JSON yang berisi tiga parameter yaitu *name*, *children* dan *size*. Parameter *name* berisi nama dari kelompok atau nama anggota kelompok. Jika parameter *name* tersebut memiliki *child*, maka *name* berisi nama kelompok, jika tidak maka *name* berisi nama anggota

kelompok. *Children* adalah parameter untuk menentukan suatu *name* merupakan nama kelompok atau nama anggota dari kelompok. Sedangkan *size* adalah ukuran dari lingkaran yang ingin dibuat.

Data yang ditampilkan menggunakan D3.js pada tugas akhir ini adalah data hasil pengelompokan kata, sehingga dapat dilihat kelompok-kelompok kata yang terbentuk. Pembangkitan berkas JSON dilakukan menggunakan data hasil dari pengelompokan kata dengan bahasa pemrograman Python.


```
{
  "name": "cluster",
  "children": [
    {
      "children": [
        {
          "name": "defisit",
          "size": 500
        },
        {
          "name": "neraca",
          "size": 500
        },
        {
          "name": "pendapatan",
          "size": 500
        }
      ]
    },
    {
      "children": [
        {
          "name": "sektor",
          "size": 500
        },
        {
          "name": "jasa",
          "size": 500
        },
        {
          "name": "tekstil",
          "size": 500
        }
      ]
    }
  ]
}
```


Gambar 2.7.1 Contoh Berkas JSON



Gambar 2.7.2 Zoomable Circle Packing

Gambar 2.7.3 Perbesaran Zoomable Circle Packing

(Halaman ini sengaja dikosongkan)

BAB III
ANALISIS DAN PERANCANGAN SISTEM

Bab ini membahas analisis dan perancangan tesaurus yang akan dibangun. Analisis sistem membahas permasalahan yang diangkat pada tugas akhir dan gambaran secara umum tesaurus yang dibangun. Sedangkan perancangan sistem menjelaskan perancangan dua proses utama pembangunan tesaurus, yaitu praproses data dan pembentukan *cluster* kata tesaurus. Selain perancangan dua proses utama, juga terdapat perancangan basis data serta perancangan antar muka.

3.1 Analisis Sistem

Analisis sistem terbagi menjadi dua bagian, yaitu analisis permasalahan yang diangkat pada tugas akhir dan gambaran umum sistem yang dibangun.

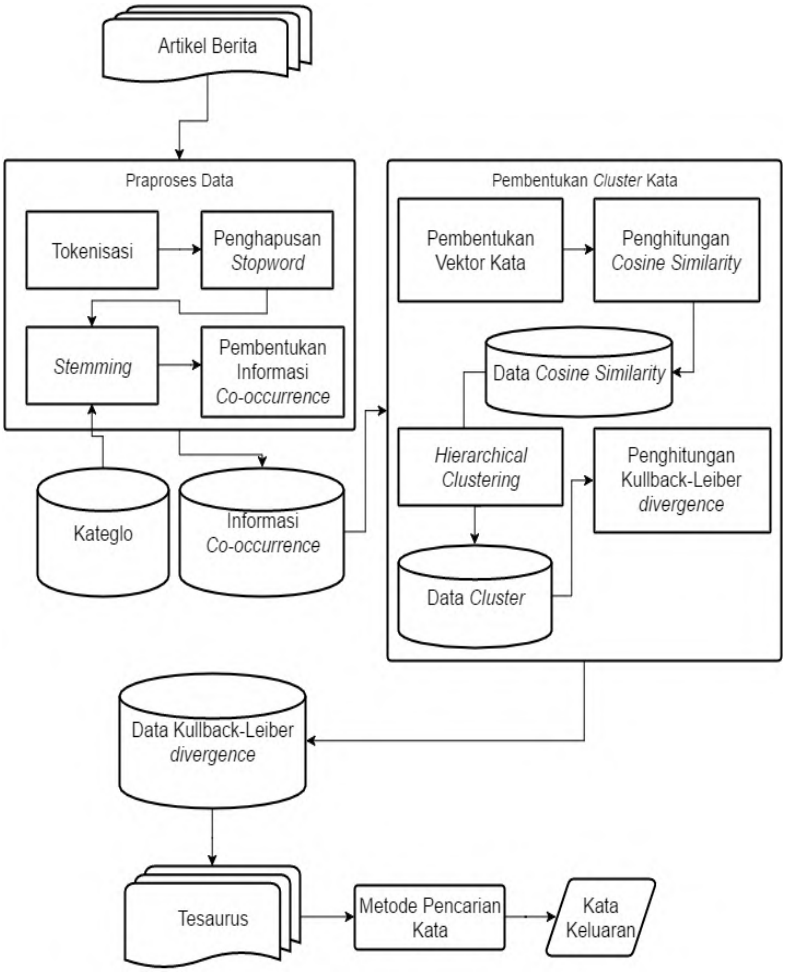
3.1.1 Analisis Permasalahan

Permasalahan yang diangkat pada tugas akhir ini adalah tesaurus seperti WordNet [4] dan Kateglo [5] umumnya dibuat secara manual, sehingga memiliki akurasi kemiripan arti kata yang tinggi, tetapi membutuhkan waktu yang cukup lama untuk melakukan penambahan kata. Kekurangan berikutnya yaitu hanya menampilkan keluaran berupa daftar kata tanpa menampilkan nilai kemiripan kata-kata tersebut. Hal tersebut menyebabkan tidak dapat diketahui seberapa mirip antara kata masukan dan kata keluaran.

Permasalahan tersebut dapat diatasi dengan membangun tesaurus yang dapat memperbarui kata-kata yang ada dan dapat menampilkan nilai kemiripan kata-kata tersebut.

3.1.2 Deskripsi Umum Sistem

Sistem atau tesaurus yang dibangun pada tugas akhir ini terbagi menjadi dua proses utama, yaitu praproses data dan pembentukan *cluster* kata. Gambar 3.1.1 adalah bentuk dari gambaran umum sistem.



Gambar 3.1.1 Gambar Deskripsi Umum Sistem

Praproses data berfungsi untuk melakukan ekstraksi kata dari kumpulan artikel berita yang ada dan membentuk informasi *co-occurrence* dari hasil ekstraksi tersebut. Sedangkan pembentukan *cluster* kata berfungsi untuk menghitung nilai jarak atau *cosine similarity* antar kata yang selanjutnya dikelompokkan

menjadi *cluster-cluster*. Kemudian menghitung nilai Kullback-Leiber *divergence* antar kata anggota masing-masing *cluster*.

3.2 Perancangan Sistem

Perancangan sistem menjelaskan perancangan dua proses utama sistem, yaitu praproses data dan pembentukan *cluster* kata tesaurus serta menjelaskan rancangan basis data, rancangan antar muka sistem dan rancangan metode pencarian kata.

3.2.1 Praproses Data

Praproses data adalah proses awal yang dilakukan untuk melakukan ekstraksi kata-kata dari kumpulan artikel berita yang ada dan membentuk informasi *co-occurrence*. Empat tahapan praproses data yang dilakukan adalah sebagai berikut:

3.2.1.1. Tokenisasi

Proses tokenisasi dilakukan pada setiap artikel berita yang ada. Tokenisasi adalah suatu proses untuk membagi suatu teks berupa kalimat atau paragraf menjadi unit-unit kecil berupa kumpulan kata atau token [12].

Sebelum melakukan tokenisasi, kalimat atau paragraf yang ada pada artikel berita disimpan pada suatu variabel. Tokenisasi dilakukan dengan menggunakan fungsi pada bahasa pemrograman PHP, yaitu ***preg_split()***. Fungsi ***preg_split()*** digunakan untuk memisahkan kata pada suatu kalimat berdasarkan tanda baca dan spasi. Hal tersebut menyebabkan ***preg_split()*** memiliki dua parameter, yaitu tanda baca atau spasi sebagai parameter pertama dan kalimat atau paragraf sebagai parameter kedua. Rancangan penggunaan fungsi ***preg_split()*** adalah sebagai berikut:

```
preg_split("/[\s,?.!()-\/\[\]:=“”;#%]+/", STRING)
```

Kata-kata yang dihasilkan dari proses tokenisasi disimpan dalam variabel berbentuk *array* yang selanjutnya akan digunakan pada proses penghapusan *stopword*.

Tabel 3.2.1 Penjelasan Fungsi *preg_split()*

Pola	Penjelasan
/	Awal dan akhir dari pola tanda baca yang akan diproses menggunakan fungsi ***preg_split()***
[…]	Tempat untuk mengelompokkan kumpulan karakter, tanda baca atau spasi
\s	Semua karakter spasi atau karakter tab
,?.!()-\/\[\]:=“”;#%	Tanda baca yang akan dijadikan pemisah kata jika terdapat pada kalimat atau paragraf
+	Menandakan beberapa atau salah satu karakter dari kelompok karakter terdapat pada STRING
STRING	Parameter kedua yang berisi variabel yang digunakan untuk menyimpan kalimat atau paragraf pada artikel berita

3.2.1.2. Penghapusan *Stopword*

Proses selanjutnya adalah melakukan penghapusan *stopword* yang terdapat pada kumpulan kata-kata hasil tokenisasi. Terdapat suatu berkas kumpulan *stopword* yang sering muncul pada suatu artikel atau dokumen. Berkas kumpulan *stopword* tersebut didapatkan dari penelitian yang dilakukan oleh F. Z. Tala [18] tentang efek *stemming* untuk temu kembali informasi pada bahasa Indonesia.

Kemudian, yang dilakukan pada isi dari berkas kumpulan *stopword* tersebut adalah membandingkannya dengan variabel *array* yang menyimpan hasil tokenisasi. Jika kata yang tersimpan pada variabel *array* terdapat pada berkas kumpulan *stopword*, kata tersebut dihapus dari variabel *array* sehingga pada proses

pembentukan informasi *co-occurrence*, kata tersebut tidak akan menjadi kata terdaftar ataupun kata *co-occurrence*.

Tujuan dari proses ini adalah agar kata yang digunakan pada proses *stemming* dan pembentukan informasi *co-occurrence* merupakan kata-kata yang memiliki informasi penting, bukan kata-kata yang sering muncul pada artikel berita.

3.2.1.3. *Stemming*

Stemming adalah proses untuk mengembalikan bentuk dari suatu kata pada bentuk dasar kata tersebut [12]. Setelah *stopword* dihapus dari variabel *array*, dilakukan proses *stemming* terhadap kata-kata yang masih tersimpan di dalamnya.

Proses *stemming* yang akan dilakukan sedikit berbeda dengan *stemming* pada umumnya. Proses *stemming* akan dilakukan dengan membandingkan kata-kata pada variabel *array* dengan kata pada Kateglo [5]. Kata-kata pada variabel *array* tidak sepenuhnya diubah ke dalam bentuk dasarnya, melainkan hanya mencari bentuk dasar dari kata yang bersangkutan. Sehingga ketika disimpan ke dalam basis data, kata dalam variabel *array* akan mempunyai kata dalam bentuk dasar yang disimpan dalam *field* yang berbeda.

Selain bentuk kata dasar, dari Kateglo juga didapatkan jenis kata dan leksikal kata. Terdapat dua jenis kata, yaitu kata dasar dan kata turunan, dan terdapat tujuh leksikal kata, yaitu adjektiva, adverbia, lain-lain, nomina, numeralia, pronominal dan verba.

Langkah-langkah yang dilakukan untuk mendapatkan bentuk dasar, jenis kata dan leksikal kata dari kata-kata pada variabel *array* adalah sebagai berikut:

1. Lakukan pencarian menggunakan kata dari variabel *array* dengan kueri beroperator *like* pada MySQL untuk melakukan pengecekan.
2. Jika kata tersebut terdapat pada Kateglo, ambil *identifier* (*id*) Kateglo dari kata yang bersangkutan.

3. Jika kata tersebut tidak ada atau tidak mempunyai identitas pada Kateglo, abaikan terlebih dahulu.
4. Gunakan identitas Kateglo yang didapatkan untuk melakukan pencarian dengan kueri beroperator *like* untuk mendapatkan jenis kata dan leksikal kata.
5. Lakukan kembali pencarian menggunakan identitas Kateglo untuk mendapatkan jenis kata dan leksikal kata.

Setelah seluruh kata pada varibel *array* melalui proses *stemming*, simpan kata-kata tersebut ke dalam basis data untuk digunakan pada proses pembentukan informasi *co-occurrence*.

3.2.1.4. Pembentukan Informasi *Co-occurrence*

Setelah seluruh kata pada artikel berita tersimpan pada basis data, dilakukan proses pembentukan informasi *co-occurrence*. Informasi *co-occurrence* dibentuk dengan memasangkan kata dengan kata yang muncul selanjutnya. Kata yang muncul selanjutnya disebut kata *co-occurrence*, sedangkan kata pasangannya disebut kata terdaftar atau *registered word*.

Langkah-langkah yang dilakukan untuk membentuk informasi *co-occurrence* dari kata-kata yang terdapat pada artikel berita adalah sebagai berikut:

1. Lakukan proses tokenisasi pada artikel berita seperti yang telah dilakukan pada subbab 3.2.1.1.
2. Simpan hasil tokenisasi pada suatu variabel *array*.
3. Lakukan kueri beroperator *like* pada basis data tempat penyimpanan kata menggunakan kata dari variabel *array* sebagai parameter pencarian.
4. Kueri pencarian tersebut bertujuan untuk mencari *identifier* (*id*) kata dari kata pada variabel *array*.
5. Jika kata-kata dalam variabel *array* tidak ditemukan identitas katanya, maka kata tersebut dihapus dari variabel *array*. Sedangkan yang ada, tetap disimpan dalam variabel *array* dengan diwakilkan oleh identitas kata dari kata tersebut.

6. Setelah semua kata diproses, dibentuklah informasi *co-occurrence* dengan memasangkan kata menggunakan indeks pada *array*. Misal nilai indeks pada *array* adalah *i*, maka pasangan katanya adalah kata dengan indeks *i* dan indeks *i+1*. Jika kata tersebut bernilai indeks terakhir dari variabel *array*, kata tersebut tidak dipasangkan dengan kata apapun.
7. Contoh dari bentuk informasi *co-occurrence* dari urutan kata berikut "defisit", "sektor", "jasa", "sektor", "faktor", "pendorong", "defisit", "neraca", "berjalan" adalah (defisit, sektor, 1), (sektor, jasa, 1), (jasa, sektor, 1), (sektor, faktor, 1), (faktor, pendorong, 1), (pendorong, defisit, 1), (defisit, neraca, 1) dan (neraca, berjalan, 1).
8. Jika terdapat pasangan kata yang sama pada satu dokumen atau pada dokumen yang berbeda, frekuensi kemunculan dari pasangan kata tersebut diakumulasikan.

Setelah semua kata-kata pada kumpulan artikel berita terbentuk informasi *co-occurrence*, selanjutnya disimpan ke dalam basis data untuk digunakan pada proses selanjutnya.

3.2.2 Pembentukan *Cluster* Kata

Pembentukan *cluster* kata dilakukan setelah melakukan penghitungan jarak antar vektor kata menggunakan *cosine similarity*. Vektor kata dibentuk dengan memanfaatkan informasi *co-occurrence* kata-kata dari kumpulan artikel berita. Algoritma *clustering* yang digunakan adalah *hierarchical clustering* dengan menggunakan pendekatan *agglomerative*.

Setelah dilakukan pembentukan *cluster* kata, dilakukan penghitungan similaritas antar vektor kata masing-masing *cluster* menggunakan Kullback-Leiber *divergence* untuk mengetahui similaritas antar vektor kata pada tiap *cluster*.

Berikut akan dijelaskan proses pembentukan vektor kata, penghitungan *cosine similarity*, proses *clustering* dan penghitungan Kullback-Leiber *divergence*.

3.2.2.1. Pembentukan Vektor Kata

Vektor kata (*WV*) dibentuk dengan melakukan *query* untuk mengelompokkan kata-kata yang memiliki *registered word* yang sama dan kata *co-occurrence* masing-masing kata ke dalam *array* yang sama. Berikut adalah contoh bentuk vektor kata berdasarkan persamaan (2.1) dari hasil pembentukan informasi *co-occurrence*.

Informasi *co-occurrence* yang terbentuk adalah (defisit, sektor, 1), (sektor, jasa, 1), (jasa, sektor, 1), (sektor, faktor, 1), (faktor, pendorong, 1), (pendorong, defisit, 1), (defisit, neraca, 1) dan (neraca, berjalan, 1). Maka salah satu vektor kata yang dapat dibentuk adalah:

$$WV_{sektor} = \begin{bmatrix} jasa[1] \\ faktor[1] \end{bmatrix}$$

Contoh di atas adalah contoh vektor kata dari kata "sektor" yang memiliki dua kata *co-occurrence*, yaitu "jasa" dan "faktor". Angka-angka di samping kata *co-occurrence* merupakan frekuensi dari pasangan kata tersebut muncul pada kumpulan artikel berita.

Bentuk yang dapat merepresentasikan vektor kata dalam bahasa pemrograman PHP adalah *array* dua dimensi, *array[i][j]*, *registered word* menjadi *key* atau indeks *i* dan kata *co-occurrence* menjadi indeks *j* dengan frekuensi menjadi *value* dari indeks *j*. Sedangkan banyaknya indeks *i* mengacu pada jumlah *registered word* dan indeks *j* mengacu pada jumlah kata *co-occurrence*. Proses pembentukan vektor kata ini dilakukan hingga seluruh kata yang terdapat pada basis vektor kata.

3.2.2.2. Penghitungan *Cosine Similarity*

Proses selanjutnya adalah penghitungan jarak antar vektor kata menggunakan *cosine similarity* [13]. Jarak vektor kata yang dihitung adalah jarak antara vektor kata *registered word* dengan vektor kata dari kata *co-occurrence* yang bersangkutan. Misalnya penghitungan *cosine similarity* antara vektor kata "miliar" sebagai

registered word dengan vektor kata "investasi", "luar", "angka" dan "lantas" didapatkan empat nilai penghitungan nilai *cosine similarity*. Berikut adalah contoh dua vektor kata, "neraca" dan "defisit":

$$WV_{neraca} = \begin{bmatrix} berjalan[3] \\ pendapatan[2] \\ defisit[1] \\ primer[1] \end{bmatrix}$$

$$WV_{defisit} = \begin{bmatrix} neraca[5] \\ sektor[1] \\ pendapatan[2] \\ primer[1] \end{bmatrix}$$

Contoh penghitungan *cosine similarity* antara dua vektor kata tersebut berdasarkan persamaan (2.3) adalah sebagai berikut:

$$D(WV_{neraca}, WV_{defisit}) = \frac{\sum_{i=1}^{t}(f_{neraca_i} \cdot f_{defisit_i})}{\sqrt{\sum_{i=1}^{t} f_{neraca_i}{}^2}\sqrt{\sum_{i=1}^{t} f_{defisit_i}{}^2}}$$

$$D(WV_{neraca}, WV_{defisit}) = \frac{(2 \times 2) + (1 \times 1)}{\sqrt{3^2 + 2^2 + 1^2 + 1^2} \times \sqrt{5^2 + 1^2 + 2^2 + 1^2}}$$

$$D(WV_{neraca}, WV_{defisit}) = \frac{5}{3{,}873 \times 5{,}568} = \frac{5}{21{,}56486} = 0{,}23185$$

Jadi nilai hasil penghitungan jarak menggunakan *cosine similarity* antara vektor kata "neraca" dan vektor kata "defisit" adalah **0,23185**. Jika terdapat suatu kasus antara dua vektor kata tersebut tidak memiliki nilai *dot product* antar kata, maka dilakukan pencarian apakah *registered word* juga menjadi kata *co-occurrence* dari kata tersebut. Sebagai contoh adalah dua vektor kata berikut. Keduanya tidak memiliki kesamaan dalam kata *co-occurrence* yang digunakan untuk menghitung *dot product* dari kedua vektor kata.

$$WV_{miliar} = \begin{bmatrix} investasi[2] \\ luar[2] \\ angka[1] \\ lantas[1] \end{bmatrix}$$

$$WV_{investasi} = \begin{bmatrix} asing[9] \\ langsung[3] \\ miliar[1] \\ pasar[1] \end{bmatrix}$$

Sehingga, langkah-langkah yang dilakukan untuk mendapatkan nilai penghitungan *cosine similarity* antara vektor kata "miliar" dan vektor kata "investasi" adalah sebagai berikut:

1. Lakukan pencarian menggunakan *registered word*, "miliar", pada vektor kata "investasi".
2. Jika ditemukan kata *co-occurrence* pada vektor kata "investasi" yang sama dengan kata "miliar", gunakan nilai frekuensi dari kata *co-occurrence* tersebut sebagai pengganti nilai *dot product* pada proses penghitungan *cosine similarity*.
3. Lakukan penghitungan *cosine similarity*.

Setelah melakukan penghitungan *cosine similarity* terhadap semua vektor kata yang terbentuk, langkah-langkah yang dilakukan selanjutnya adalah sebagai berikut:

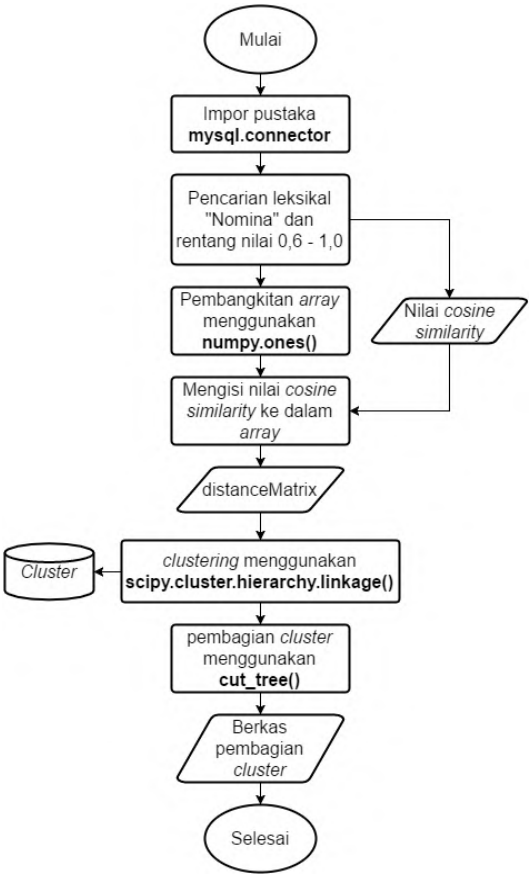
1. Setelah mendapatkan hasil penghitungan dari semua vektor kata yang terbentuk, kurangi angka 1 dengan nilai hasil penghitungan untuk memperoleh nilai kebalikan dari penghitungan *cosine similarity*.
2. Hal tersebut bertujuan untuk membalik nilai hasil penghitungan, dengan anggapan bahwa semakin kecil nilai hasil penghitungan *cosine similarity* awal (sebelum dibalik), jarak vektor kata tersebut semakin jauh. Sedangkan pada proses selanjutnya, *clustering*, semakin

kecil nilai *cosine similarity*, jarak vektor kata semakin dekat.

3. Simpan hasil penghitungan ke dalam basis data.

Selanjutnya, hasil penghitungan digunakan pada proses pengelompokan kata atau *clustering* menggunakan *hierarchical clustering* dengan pendekatan *agglomerative*.

3.2.2.3. *Hierarchical Clustering*



Gambar 3.2.1 Diagram Alur *Hierarchical Clustering*

Proses setelah melakukan penghitungan *cosine similarity* adalah proses mengelompokkan hasil penghitungan tersebut menggunakan algoritma *hierarchical clustering* dengan pendekatan *agglomerative*. Berbeda dengan proses-proses sebelumnya yang menggunakan bahasa pemrograman PHP, proses *clustering* menggunakan bahasa pemrograman Python, digunakan pustaka NumPy untuk melakukan pembangkitan *array* dan pustaka SciPy untuk melakukan proses *clustering*. Penjelasan penggunaan bahasa Python akan dijelaskan pada Bab IV, subbab 4.4.3, Implementasi *Hierarchical Clustering*.

Berikut ini adalah langkah-langkah untuk melakukan proses *clustering* menggunakan *hierarchical clustering* pada Python menggunakan pustaka Scipy:

1. Impor pustaka Python yang digunakan untuk menghubungkan berkas Python dengan basis data MySQL, ***mysql.connector***.
2. Pencarian pada basis data untuk mendapatkan nilai *cosine similarity* pasangan kata dengan leksikal "Nomina" dan rentang nilai *cosine similarity* antara 0,6 hingga 1,0.
3. Lakukan pembangkitan *array* yang berisi angka 1 menggunakan fungsi dari pustaka NumPy, yaitu ***numpy.ones()*** dengan ukuran sebanyak *distinct* kata yang dihasilkan pada *query* sebelumnya.
4. Gunakan *distinct* kata sebagai koordinat untuk melakukan penempatan nilai penghitungan *cosine similarity* pada *array* yang telah dibangkitkan.
5. Setelah semua nilai dimasukkan ke dalam *array* berupa *distance matrix*, masukkan *array* tersebut ke dalam fungsi pada pustaka Scipy untuk melakukan *clustering*, ***scipy.cluster.hierarchy.linkage()***.
6. *Array* berupa *distance matrix* diubah menjadi lebih ringkas atau ke dalam bentuk *condensed matrix* oleh pustaka tersebut menggunakan fungsi ***pdist()***. Contoh dari penghitungan menggunakan fungsi ***pdist()*** adalah sebagai berikut:

$$x = array([0,10],[10,10],[20,20])$$
$$pdist(x) = array(10,22{,}36067,14{,}14213)$$

Elemen pertama dari *array* hasil fungsi ***pdist()*** merupakan hasil penghitungan antara elemen pertama dan kedua dari *array x*, elemen kedua hasil penghitungan antara elemen pertama dan ketiga *array x*, sedangkan elemen ketiga merupakan hasil penghitungan antara elemen kedua dan ketiga dari *array x*. Sehingga dapat diketahui bahwa bentuk *condensed matrix* merupakan bentuk perubahan dari koordinat titik menjadi jarak antar titik yang terdapat pada suatu *array*.

7. Hasil keluaran dari fungsi ***linkage()*** adalah *array* dari satu *cluster* besar dengan elemen berupa *cluster* yang tergabung, jarak antara kedua *cluster* tersebut dan jumlah anggota yang menyusun *cluster*.
8. Setelah terbentuk menjadi satu *cluster* besar, *cluster* tersebut akan dibagi menjadi beberapa *cluster* menggunakan fungsi ***cut_tree()*** dengan memanfaatkan nilai *height* dari *array* hasil keluaran fungsi *clustering*. Penentuan penggunaan nilai *height* tergantung dari persebaran nilai jarak antar *cluster* dari hasil proses fungsi ***linkage*** yang akan dijelaskan pada Bab V.
9. Simpan hasil pembagian *cluster* ke dalam suatu berkas teks berupa kelompok kata-kata anggota tiap *cluster* untuk digunakan pada proses penghitungan Kullback-Leiber *divergence*.

Selain menyimpan hasil *clustering* ke dalam sebuah berkas teks, simpan juga hasil tersebut ke dalam basis data untuk mempersiapkan jika diperlukan pada proses yang lain.

3.2.2.4. Penghitungan Kullback-Leiber *divergence*

Proses yang dilakukan setelah terbentuknya *cluster-cluster* kata adalah proses melakukan penghitungan similaritas

antar kata anggota *cluster* menggunakan Kullback-Leiber *divergence*. Proses penghitungan Kullback-Leiber *divergence* tetap menggunakan bahasa pemrograman Python.

Penghitungan Kullback-Leiber *divergence* memanfaatkan vektor kata yang didalamnya termasuk kata *co-occurrence* dan nilai frekuensi *co-occurrence*. Sebelum melakukan penghitungan Kullback-Leiber *divergence*, diperlukan nilai distribusi probabilitas dari setiap pasangan kata pada suatu *cluster*.

Contoh penghitungan nilai Kullback-Leiber *divergence* dari pasangan vektor kata yang terdapat dalam satu *cluster* antara vektor kata "neraca" (WV_{neraca}) dan vektor kata "defisit" ($WV_{defisit}$) berdasarkan persamaan (2.5), (2.6) dan (2.7) adalah sebagai berikut:

$$WV_{neraca} = \begin{bmatrix} berjalan[3] \\ pendapatan[2] \\ defisit[1] \\ primer[1] \end{bmatrix}$$

$$WV_{defisit} = \begin{bmatrix} neraca[5] \\ sektor[1] \\ pendapatan[2] \\ primer[1] \end{bmatrix}$$

Dua vektor kata tersebut memiliki dua kata *co-occurrence* yang sama yaitu "pendapatan" dan "primer". Sehingga terdapat dua nilai distribusi probabilitas untuk masing-masing vektor kata yaitu $P_{neraca_{pendapatan}}$, $P_{neraca_{primer}}$ untuk WV_{neraca} dan $P_{defisit_{pendapatan}}$, $P_{defisit_{primer}}$ untuk $WV_{defisit}$. Nilai j bergantung pada jumlah kata *co-occurrence* yang dimiliki masing-masing vektor kata. Berikut adalah nilai untuk masing-masing distribusi probablitas.

$$P_{neraca_{pendapatan}} = \frac{f_{neraca_{pendapatan}}}{\sum_{i=1}^{j} f_{neraca}} = \frac{2}{7} = 0{,}2857$$

$$P_{neraca_{primer}} = \frac{f_{neraca_{primer}}}{\sum_{i=1}^{j} f_{neraca}} = \frac{1}{7} = 0{,}1428$$

$$P_{defisit_{pendapatan}} = \frac{f_{defisit_{pendapatan}}}{\sum_{i=1}^{j} f_{defisit}} = \frac{2}{9} = 0{,}2222$$

$$P_{defisit_{primer}} = \frac{f_{defisit_{primer}}}{\sum_{i=1}^{j} f_{defisit}} = \frac{1}{9} = 0{,}1111$$

$P_{neraca_{pendapatan}}$ merupakan nilai distribusi probabilitas untuk kata *co-occurrence* "pendapatan" pada WV_{neraca}. $f_{neraca_{pendapatan}}$ bernilai 2 didapatkan dari frekuensi kata *co-occurrence* "pendapatan" pada WV_{neraca}. Sedangkan nilai 7 didapatkan dari jumlah frekuensi kata *co-occurrence* dari WV_{neraca}. Setelah melakukan penghitungan distribusi probabilitas, nilai hasil penghitungan tersebut dimasukkan ke dalam persamaan Kullback-Leiber *divergence* untuk mengetahui nilai dari pasangan vektor kata tersebut. Nilai t pada contoh penghitungan ini adalah 2 karena jumlah kata *co-occurrence* yang sama antara kedua vektor kata tersebut adalah 2.

$$KL(WV_{neraca}, WV_{defisit}) = \sum_{i=1}^{t} P_{neraca_i} \log \frac{P_{neraca_i}}{P_{defisit_i}}$$

$$P_{neraca_{pendapatan}} \log \frac{P_{neraca_{pendapatan}}}{P_{defisit_{pendapatan}}} + P_{neraca_{primer}} \log \frac{P_{neraca_{primer}}}{P_{defisit_{primer}}}$$

$$\left(0{,}2857 \times \log \frac{0{,}2857}{0{,}2222}\right) + \left(0{,}1428 \times \log \frac{0{,}1428}{0{,}1111}\right)$$

$$(0{,}2857 \times \log 1{,}2869) + (0{,}1428 \times \log 1{,}2853)$$

$$(0{,}2857 \times 0{,}1095) + (0{,}1428 \times 0{,}1090)$$

$$0{,}0313 + 0{,}0156 = 0{,}0469$$

Berdasarkan hasil penghitungan Kullback-Leiber *divergence* nilai yang dihasilkan antara kata "neraca" dan "defisit" adalah **0,0469**.

Sedangkan untuk rancangan implementasi penghitungan Kullback-Leiber *divergence* pada tesaurus yang dibangun adalah sebagai berikut:

1. Tesaurus membaca terlebih dahulu berkas teks yang berisi kata-kata anggota masing-masing *cluster* yang terbentuk dari proses *clustering*.
2. Melakukan pencarian kata-kata *co-occurrence* tiap kata anggota *cluster* sehingga didapatkan kata *co-occurrence* dan nilai frekuensinya.
3. Membandingkan kata *co-occurrence* pasangan kata.
4. Jika terdapat kata *co-occurrence* yang sama, maka dilakukan penghitungan distribusi probabilitas. Jika tidak terdapat kata *co-occurrence* yang sama, maka secara otomatis, nilai distribusi probabilitasnya adalah 0.
5. Selanjutnya adalah melakukan penghitungan Kullback-Leiber *divergence* menggunakan nilai distribusi probabilitas dari pasangan kata tersebut.
6. Setelah antar anggota *cluster* memiliki nilai Kullback-Leiber *divergence*, simpan hasil penghitungan tersebut ke dalam basis data.

Penghitungan Kullback-Leiber *divergence* dapat dilakukan jika terdapat lebih dari satu kata dalam satu *cluster*. Jika *cluster* hanya beranggotakan satu kata, maka penghitungan Kullback-Leiber *divergence* tidak dapat dilakukan.

3.2.3 Perancangan Basis Data

Setiap proses yang dilakukan pada tugas akhir ini menghasilkan suatu keluaran seperti kata pada proses ekstraksi kata artikel berita, hasil penghitungan *cosine similarity*, hasil *clustering* dan hasil penghitungan Kullback-Leiber *divergence*. Hasil penghitungan *cosine similarity* digunakan untuk melakukan proses pembentukan *cluster* kata atau *clustering* sehingga hal tersebut menyebabkan diperlukan tempat untuk menyimpan nilai keluaran-keluaran sehingga dapat digunakan pada proses yang lain.

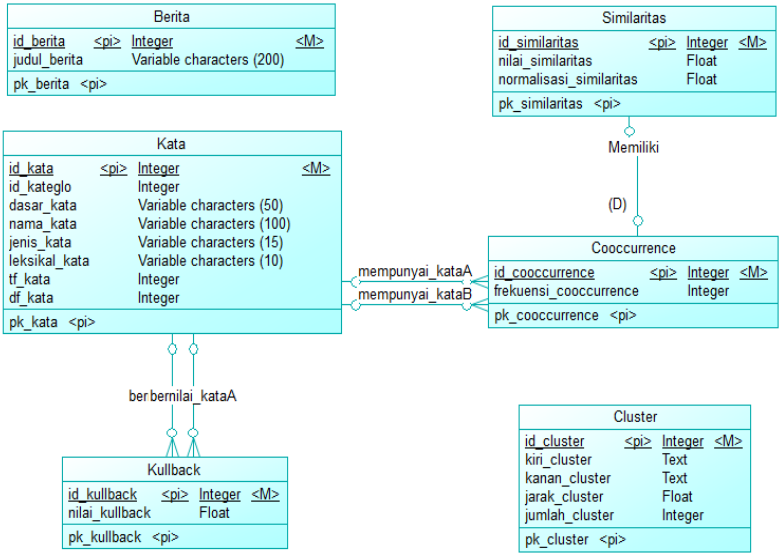
Perancangan *Conceptual Data Model* (CDM) dan *Physical Data Model* (PDM) basis data untuk menyimpan hasil keluaran penghitungan maupun hasil ekstraksi kata yang akan digunakan adalah sebagai berikut:

Terdapat enam tabel yang akan digunakan selama proses pembangunan tesaurus, yaitu tabel berita, cluster, cooccurrence, kata, kullback dan similaritas. Masing-masing kegunaan tabel adalah sebagai berikut:

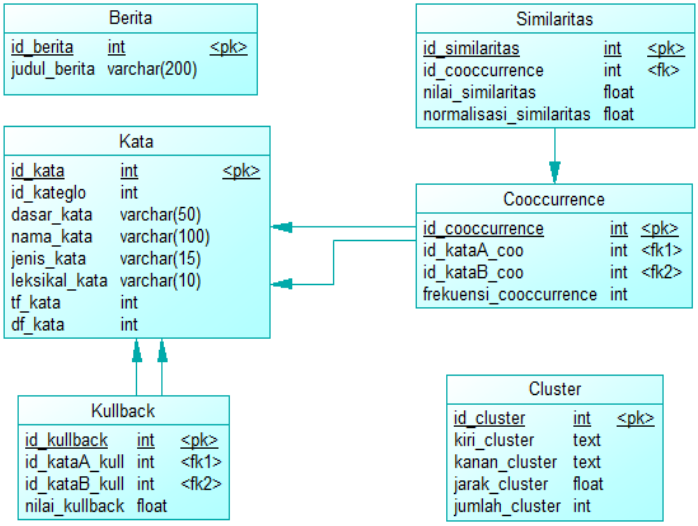
1. Tabel berita digunakan untuk menyimpan judul berita yang telah diproses. Memiliki dua atribut, yaitu id_berita dan judul_berita.
2. Tabel cluster digunakan untuk menyimpan hasil *clustering*. Memiliki lima atribut, yaitu id_cluster, kiri_cluster, kanan_cluster, jarak_cluster dan jumlah_cluster. Atribut kiri_cluster berguna untuk menyimpan hasil penggabungan *hierarchical clustering* tiap iterasi pada *child* atau cabang sebelah kiri, sedangkan kanan_cluster menyimpan *child* sebelah kanan.
3. Tabel cooccurrence digunakan untuk menyimpan hasil pembentukan informasi *co-occurrence* sehingga dapat digunakan ketika proses penghitungan *cosine similarity*. Memiliki empat atribut, yaitu id_cooccurrence, id_kataA_coo dan id_kataB_coo yang merupakan *foreign key* dari tabel kata serta atribut frekuensi_cooccurrence.
4. Tabel kata digunakan untuk menyimpan kata hasil ekstraksi. Memiliki delapan atribut, yaitu id_kata,

id_kateglo, dasar_kata, nama_kata, jenis_kata, leksikal_kata, tf_kata dan df_kata. Atribut id_kateglo digunakan untuk menyimpan *identifier* dari Kateglo jika kata tersebut ada pada basis data Kateglo.

5. Tabel kullback digunakan untuk menyimpan nilai hasil penghitungan Kullback-Leiber *divergence*. Memiliki empat atribut, yaitu id_kullback id_kataA_kull dan id_kataB_kull yang merupakan *foreign key* dari tabel kata serta atribut nilai_kullback.
6. Tabel similaritas digunakan untuk menyimpan hasil penghitungan *cosine similarity* dan hasil normalisasi nilai *cosine similarity*. Memiliki empat atribut, yaitu id_similaritas, id_cooccurrenceSim yang merupakan *foreign key* dari tabel co-occurrence, nilai_similaritas dan normalisasi_similaritas.



Gambar 3.2.2 *Conceptual Data Model* Tesaurus



Gambar 3.2.3 *Physical Data Model* Tesaurus

3.2.4 Perancangan Antar Muka

Antar muka dibuat sebagai perantara tesaurus dan pengguna untuk memberikan kemudahan pada pengguna menggunakan tesaurus yang telah dibangun. Sehingga diperlukan tiga antar muka, yaitu antar muka pencarian kata, antar muka hasil pencarian kata dan antar muka visualisasi hasil *clustering* menggunakan pustaka Data-Driven Documents atau D3.js.

3.2.4.1. Antar Muka Pencarian Kata

Antar muka ini adalah antar muka awal dari tesaurus saat tesaurus pertama kali dijalankan oleh pengguna. Hal yang dapat dilakukan oleh pengguna pada halaman ini adalah melakukan pencarian kata dan menuju halaman visualisasi *clustering*.

Terdapat kotak pencarian dan tombol pencarian yang dapat digunakan untuk memasukkan kata pada tesaurus dan menggunakan kata tersebut sebagai parameter pencarian sehingga

tesaurus dapat menampilkan kata-kata yang memiliki nilai kedekatan tertinggi dengan kata masukan tersebut.

Selain kotak pencarian, juga terdapat tautan untuk menuju antar muka lain yaitu antar muka visualisasi hasil *clustering*. Penjelasan detail tentang antar muka pencarian kata dapat dilihat pada Tabel 3.2.2.

Gambar 3.2.4 Antar Muka Pencarian Kata

Tabel 3.2.2 Tabel Antar Muka Pencarian Kata

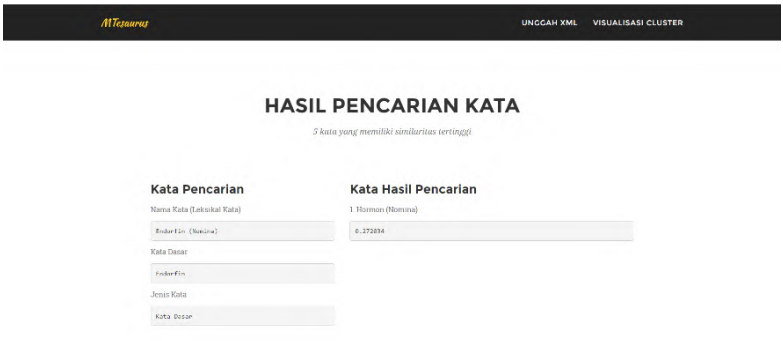
No.	Nama	Jenis	Fungsi	Masukan / Keluaran
1	label_tesaurus	Label	Menampilkan nama dari tesaurus	-
2	label_pencarian	Label	Menampilkan label formulir pencarian	-
3	form_pencarian	Form	Memasukkan kata pencarian untuk diproses oleh tesaurus	Teks kata pencarian
4	btn_cari	Button	Mengirim kata pencarian ke	-

			dalam tesaurus	
5	link_cluster	Link	Tautan untuk menampilkan antar muka visualisasi hasil *clustering*	-

3.2.4.2. Antar Muka Hasil Pencarian Kata

Antar muka ini adalah antar muka yang digunakan untuk menampilkan hasil keluaran dari pencarian kata yang dilakukan sebelumnya oleh pengguna. Hal yang dapat dilakukan oleh pengguna pada halaman ini adalah melihat hasil keluaran tesaurus, kata-kata dan nilai kemiripannya serta melihat keterangan yang diberikan jika kata masukan tidak terdapat pada tesaurus.

Pada tampilan antar muka hasil pencarian kata terdapat label yang menunjukkan kata masukan, kata keluaran dan nilai kedekatan di antara kata-kata tersebut. Jika suatu kata masukan tidak ada dalam tesaurus, maka keluaran yang diberikan oleh tesaurus adalah pemberitahuan bahwa kata pencarian tidak ada. Penjelasan detail tentang antar muka hasil pencarian kata dapat dilihat pada Tabel 3.2.3.



Gambar 3.2.5 Hasil Pencarian Kata

Gambar 3.2.6 Tidak Terdapat Hasil Pencarian Kata

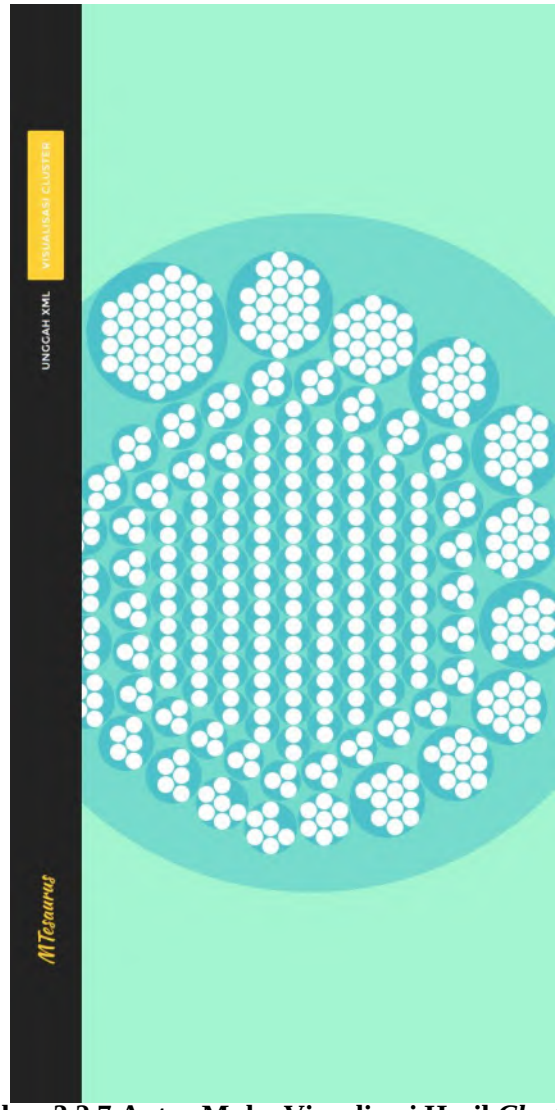
Tabel 3.2.3 Tabel Antar Muka Hasil Pencarian Kata

No.	Nama	Jenis	Fungsi	Masukan / Keluaran
1	label_masukan	Label	Menampikan nama kata masukan	-
2	label_keluaran	Label	Menampikan nama kata keluaran tesaurus	-
3	label_nilai	Label	Menampikan nilai keluaran yang diberikan tesaurus	-
4	label_notifikasi	Label	Menampikan notifikasi dari hasil pencarian	-

3.2.4.3. Antar Muka Visualisasi Hasil *Clustering*

Antar muka ini adalah antar muka yang digunakan untuk menampilkan hasil dari proses *clustering* yang telah dilakukan. Hal yang dapat dilakukan oleh pengguna pada halaman ini adalah

melihat hasil visualisasi *clustering* dan melihat kata-kata anggota tiap *cluster* yang terbentuk.



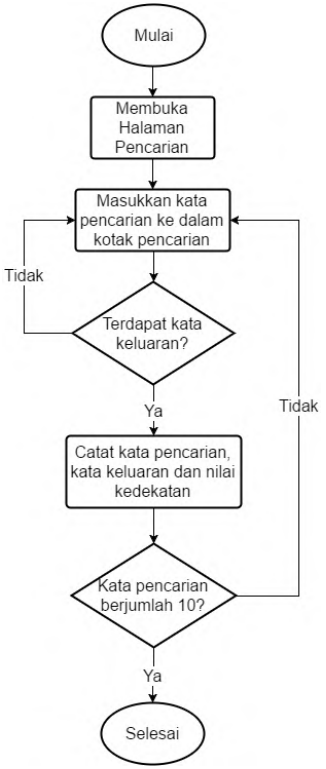
Gambar 3.2.7 Antar Muka Visualisasi Hasil *Clustering*

Gambar 3.2.8 Perbesaran Lingkaran Hasil *Clustering*

Visualisasi *clustering* yang dilakukan menggunakan pustaka D3.js yaitu Zoomable Circle Packing dengan menggunakan pembangkitan berkas JSON dari data hasil proses *hierarchical clustering*.

Pada tampilan antar muka visualisasi hasil *clustering*, tampilan berupa lingkaran-lingkaran berwarna hijau menunjukkan kelompok kata dan lingkaran berwarna putih menunjukkan kata anggota dari kelompok tersebut. Lingkaran yang berwarna hijau dapat diperbesar sehingga dapat terlihat nama dari kata anggota kelompok atau lingkaran tersebut.

3.2.5 Perancangan Metode Pencarian Kata



Gambar 3.2.9 Diagram Alur Metode Pencarian Kata

Metode pencarian kata adalah metode yang digunakan untuk mengetahui hasil keluaran dari tesaurus yang telah dibangun. Hasil keluaran tersebut berupa daftar kata dan nilai kedekatannya dengan kata masukan atau kata yang digunakan untuk melakukan kueri pencarian.

Metode ini diterapkan setelah tesaurus selesai dibangun dan antar muka selesai dibuat serta digunakan dalam skenario uji coba pada tesaurus. Metode ini juga memfasilitasi pengguna untuk melakukan pencarian kata terhadap tesaurus.

Metode pencarian kata memiliki tiga proses, yaitu:

1. Membuka halaman pencarian, untuk membuka antar muka pencarian kata tesaurus.
2. Masukkan kata pencarian ke dalam kotak pencarian, sehingga kata masukan dapat diproses dalam tesaurus dan dapat menghasilkan keluaran berupada daftar kata dan nilai kedekatannya.
3. Mencatat kata pencarian, kata keluaran dan nilai kedekatan.

BAB IV
IMPLEMENTASI

Bab ini membahas proses implementasi yang dilakukan berdasarkan hasil perancangan tesaurus yang dibangun pada Bab III. Penjelasan proses implementasi terbagi menjadi enam bagian, yaitu lingkungan implementasi, penggunaan Kateglo dan pustaka bahasa pemrograman, implementasi praproses data, implementasi pembentukan *cluster* kata, implementasi penampilan hasil *clustering* dan implementasi basis data.

4.1 Lingkungan Implementasi

Lingkungan implementasi adalah lingkungan di mana tesaurus dibangun. Lingkungan implementasi dibagi menjadi dua yaitu perangkat keras dan perangkat lunak.

4.1.1 Perangkat Keras

Lingkungan implementasi perangkat keras dari tugas akhir ini adalah sebagai berikut:

- Tipe : ASUS ROG GL552VW
- Prosesor : Inter® Core(TM) i7-6700HQ CPU (8 CPUs) @ 2.6GHz
- Memori (RAM) : 8192 MB

4.1.2 Perangkat Lunak

Lingkungan implementasi perangkat lunak dari tugas akhir ini adalah sebagai berikut:

- Sistem operasi : Windows 10 Enterprise 64-bit
- Bahasa Pemrograman : PHP 5.6.3, Python 2.7.11
- *Text Editor* : Sublime Text 3, IDLE 2.7.11
- Kerangka Kerja : CodeIgniter 3.0.3
- *Web Server* : Apache 2.4.4
- Basis Data : MySQL 5.5.32
- Kakas Bantu Basis Data : HeidiSQL 9.3.0.4984 (64-bit)

4.2 Penggunaan Kateglo, Pustaka dan Kerangka Kerja

Penggunaan Kateglo dan pustaka-pustaka bahasa pemrograman Python merupakan salah satu hal penting dalam proses implementasi tesaurus. Hal tersebut menjadi penting karena dapat membantu untuk menyediakan fungsi-fungsi yang dapat digunakan dalam penerapan perancangan tesaurus yang telah dibuat pada Bab III.

4.2.1 Kerangka Kerja CodeIgniter

Kerangka kerja CodeIgniter (CI) terbagi menjadi tiga bagian utama, yaitu *model*, *view* dan *controller*. Tiga bagian tersebut memiliki fungsi masing-masing yang berbeda satu dengan bagian yang lain. Hal tersebut ditujukan agar sistem yang dibangun dapat tersusun rapi karena terdapat pada fungsi masing-masing.

Model berfungsi untuk menjadi penghubung antara sistem dan basis data yang digunakan untuk melakukan fungsi yang pencarian, penambahan, penghapusan dan pembaruan data. *View* berfungsi untuk mengatur tampilan yang ingin ditampilkan pada pengguna, berbentuk halaman HTML. Sedangkan *controller* berfungsi untuk melakukan pemrosesan pada data dan menjadi penghubung antara *model* dan *view*. Berikut adalah contoh dari implementasi *model*, *view*, *controller*.

```
<?php if ( ! defined('BASEPATH')) exit('No direct script
access allowed');
class Stemming extends CI_Model {
...
}
```

Kode Sumber 4.2.1 Contoh Impementasi *Model*

```
<?php if ( ! defined('BASEPATH')) exit('No direct script
access allowed');
class Praproses extends CI_Controller {
...
}
```

Kode Sumber 4.2.2 Contoh Impementasi *Controller*

```
<!DOCTYPE html>
<html lang="en">
   <head></head>
   <body>
      ...
   </body>
</html>
```

Kode Sumber 4.2.3 Contoh Impementasi *View*

4.2.2 Kateglo

```
function getIdKateglo($kata)
{
   $this->db->select('lemma_id');
   $this->db->from('lemma');
   $this->db->like('lemma_name', $kata, 'none');
   $query = $this->db->get();
   return $query->row('lemma_id');
}

function getLeksikalKata($id)
{
   $data['lemma.lemma id'] = $id;
   $this->db->select('lexical_name');
   $this->db->from('lemma');
   $this->db->join('definition', 'lemma.lemma_id =
   definition.definition_lemma_id');
   $this->db->join('lexical',
   'definition.definition lexical id = lexical.lexical id');
   $this->db->where($data);
   $query = $this->db->get();
   return $query->row('lexical_name');
}

function getJenisKata($id)
{
   $data['lemma.lemma_id'] = $id;
   $this->db->select('type name');
   $this->db->from('lemma');
   $this->db->join('lemma type', 'lemma.lemma id =
   lemma type.lemma id');
   $this->db->join('type', 'lemma_type.type_id =
   type.type_id');
   $this->db->where($data);
   $query = $this->db->get();
   return $query->row('type_name');
}
```

Kode Sumber 4.2.4 *Model* Pengambilan Data Kateglo

Kode Sumber 4.2.4 diimplementasikan pada bagian *model* CI dengan menggunakan koneksi pada basis data Kateglo. Terdapat tiga fungsi, yaitu ***getIdKateglo()***, ***getLeksikalKata()*** dan ***getJenisKata()***. Pencarian *id*, jenis kata dan leksikal kata tersebut pada setiap fungsi memanfaatkan kueri SELECT, FROM, JOIN dan WHERE.

getIdKateglo() berfungsi untuk mendapatkan *id* pada basis data Kateglo menggunakan parameter kata artikel berita. Jika kata tersebut memiliki padanan kata yang sama pada basis data Kateglo, maka *id* dari padanan kata tersebut menjadi hasil kembalian dari fungsi ini.

getLeksikalKata() berfungsi untuk mendapatkan leksikal kata pada basis data Kateglo menggunakan parameter *id* Kateglo. Kembalian dari fungsi ini adalah leksikal dari suatu kata berdasarkan *id* Kateglo yang dimasukkan. Jika kata tersebut tidak memiliki leksikal, maka kembalian fungsi adalah *NULL*.

Sedangkan ***getJenisKata()*** berfungsi untuk mendapatkan jenis kata pada basis data Kateglo menggunakan parameter *id* Kateglo. Jika kata dari *id* Kateglo tersebut memiliki jenis kata, maka kembalian fungsi adalah jenis kata, jika tidak kembalian fungsi adalah *NULL*.

4.2.3 MySQL Connector

MySQL Connector pada Kode Sumber 4.2.5 digunakan untuk menghubungkan antara basis data MySQL pada bahasa pemrograman Python. Parameter yang digunakan pada fungsi ***mysql.connector.connect()*** adalah *user*, *password*, *host* dan *database*. *User* dan *password* digunakan untuk memasukkan nama pengguna dan kata sandi yang digunakan pada basis data. Parameter *host* digunakan untuk memasukkan alamat dari basis data MySQL yang akan dihubungkan dengan bahasa pemrograman Python, jika terdapat pada jaringan lokal atau *localhost*, masukkan alamat IP dari *localhost* yaitu 127.0.0.1. Sedangkan parameter *database* digunakan untuk memasukkan nama basis data yang akan digunakan.

```
...
import mysql.connector
from mysql.connector import errorcode
...
try:
   connect = mysql.connector.connect(user='root',
   password='', host='127.0.0.1', database='ta')
   ...
except mysql.connector.Error as err:
   if err.errno == errorcode.ER_ACCESS_DENIED_ERROR:
      print("Something is wrong with your user name or
      password")
   elif err.errno == errorcode.ER_BAD_DB_ERROR:
      print("Database does not exist")
   else:
      print(err)
connect.close()
```

Kode Sumber 4.2.5 Penggunaan MySQL Connector

4.2.4 NumPy

Kode Sumber 4.2.6 merupakan cara untuk implementasi penggunaan NumPy pada tesaurus. Sebelum mengimpor pustaka NumPy, pastikan pustaka NumPy sudah terpasang pada lingkungan implementasi. Proses penggunaan NumPy adalah setelah penghitungan nilai *cosine similarity* selesai.

Fungsi ***np.ones()*** digunakan untuk melakukan pembangkitan *array* atau matriks dengan jumlah elemen sesuai dengan banyaknya kata pada variabel *listMerge*, yaitu variabel yang digunakan untuk menampung kata-kata hasil dari proses penghitungan *cosine similarity*. Contohnya, jika pada variabel *listMerge* terdapat 10 kata hasil penghitungan *cosine similarity*, berarti jumlah elemen pada matriks yang dibangkitkan adalah $10 \times 10 = 100$ elemen.

```
...
import numpy as np
...
countList = len(listMerge)
distanceMatrix = np.ones((countList, countList))
...
```

Kode Sumber 4.2.6 Penggunaan NumPy

4.2.5 SciPy

```
...
import scipy.cluster.hierarchy as cls
import scipy.cluster.hierarchy_new as cls_new
...
cluster = cls.linkage(distanceMatrix, method='single',
metric='euclidean')
cut_tree = cls_new.cut_tree(cluster,height=1.393)
...
```

Kode Sumber 4.2.7 Penggunaan SciPy

Terdapat dua versi modul ***scipy.cluster.hierarchy*** pada pustaka SciPy yang digunakan untuk melakukan implementasi tesaurus. Perbedaan dari kedua modul tersebut adalah pada fungsi ***cut_tree()*** dan parameter yang digunakan pada fungsi ***linkage()***. Modul ***cls*** tidak memiliki fungsi ***cut_tree()*** sedangkan ***cls_new***, parameter yang digunakan untuk ***linkage()*** berbeda dengan modul ***cls*** yang menggunakan matriks jarak, metode penghitungan jarak antar *cluster* dan pembentukan matriks jarak menjadi *condensed matrix*. Sehingga hal tersebut menyebabkan harus mengimpor dua modul ***scipy.cluster.hierarchy***.

Kode Sumber 4.2.7 merupakan cara untuk melakukan implementasi penggunaan SciPy modul ***scipy.cluster.hierarchy*** pada tesaurus yang dibangun. Sebelum mengimpor pustaka SciPy, pastikan pustaka SciPy sudah terpasang pada lingkungan implementasi.

Informasi yang didapatkan dari implementasi fungsi ***linkage()*** di atas adalah variabel yang menampung matriks jarak adalah *distanceMatrix*, metode yang digunakan untuk penghitungan jarak antar *cluster* adalah *single* dan parameter *metric* atau cara penghitungan untuk membentuk *condensed matrix* adalah *euclidean distance*.

Sedangkan pada fungsi ***cut_tree()*** yang digunakan adalah parameter *cluster*, hasil keluaran dari fungsi ***linkage()*** dan parameter *height*, yaitu jarak hasil penggabungan *cluster*. Satu *cluster* besar yang terbentuk, dibagi menjadi *cluster-cluster* yang lebih kecil berdasarkan parameter *height*.

4.2.6 Data-Driven Documents

```
...
<script src="assets/js/d3.js"></script>
<script src="assets/js/d3.min.js"></script>
...
<script>
...
d3.json("json/data.json", function(error, root) {
...
}
</script>
```

Kode Sumber 4.2.8 Penggunaan Data-Driven Documents

Implementasi Data-Driven Documents atau D3.js adalah dengan mengunduh berkas *d3.js* dan *d3.min.js* pada situs resmi D3.js [17] dan mengimpornya ke dalam berkas antar muka visualisasi hasil *clustering*. Berkas *data.json* adalah berkas hasil pembangkitan JSON menggunakan hasil dari proses *clustering* dan penggunaan fungsi ***cut_tree()***.

4.3 Implementasi Praproses Data

Tahapan implementasi dari hasil proses perancangan praproses data yang telah dilakukan pada Bab III. Implementasi yang dilakukan adalah implementasi proses tokenisasi kata pada artikel berita, penghapusan *stopword*, *stemming* dan pembentukan informasi *co-occurrence* menggunakan bahasa pemrograman PHP dan kerangka kerja CodeIgniter (CI).

4.3.1 Tokenisasi

```
public function praproses()
{
   ...
   foreach ($files as $key => $value)
   {
      $docXML = $value;
      $fileXML = simplexml_load_file($dir.$docXML);
      ...
   }
   ...
}
```

Kode Sumber 4.3.1 Proses Pengambilan Data xml

```
public function tokenisasiJudul($file, $stopword)
{
   $judulBerita = strtolower($file->judul);
   $judul = preg_split("/[\s,?.!()-\/\[\]:=“”;#%]+/",
   $judulBerita);
   ...
}
```

Kode Sumber 4.3.2 Proses Tokenisasi Judul Berita

```
public function tokenisasiIsi($file, $stopword)
{
   $kalimatBerita = (array)$file->isi->paragraf;
   $isiBerita = strtolower(implode(".", $kalimatBerita));
   $berita = preg_split("/[\s,?.!()-\/\[\]:=“”;#%]+/",
   $isiBerita);
   ...
}
```

Kode Sumber 4.3.3 Proses Tokenisasi Isi Berita

Proses tokenisasi diimplementasikan pada bagian *controller* CI. Pada Kode Sumber 4.3.1 terdapat suatu fungsi bahasa pemrograman PHP, yaitu fungsi ***simplexml_load_file()*** digunakan untuk memuat berkas xml untuk diproses. Parameter yang dikirim pada fungsi tersebut adalah nama dari berkas xml.

Setelah berkas xml dimuat, dilakukan tokenisasi dengan menggunakan ***preg_split()*** pada fungsi ***tokenisasiJudul()*** dan fungsi ***tokenisasiIsi()***. Perbedaan antara kedua fungsi tersebut adalah proses pengambilan data dari berkas xml yang dimuat, sedangkan parameter yang dikirim sama, yaitu variabel penampung berkas xml dan variabel penampung *stopword*. Hasil fungsi ***preg_slit()*** dimasukkan ke dalam suatu variabel *array* untuk menampung sementara.

4.3.2 Penghapusan *Stopword*

```
public function stopword()
{
   $string = read_file('./stopword/stopword.txt');
   $stopword = preg_split("/[\s,]+/", $string);
   return $stopword;
}
```

Kode Sumber 4.3.4 Proses Membaca Berkas *Stopword*

```
public function tokenisasiJudul($file, $stopword)
{
   ...
   $kata = array();
   foreach ($judul as $key => $value)
   {
      if (!in_array($value, $stopword))
      {
         $kata[] = $value;
      }
   }
   return $kata;
}
```

Kode Sumber 4.3.5 Proses Penghapusan *Stopword* pada Judul

```
public function tokenisasiIsi($file, $stopword)
{
   ...
   $kata = array();
   foreach ($berita as $key => $value)
   {
      if (!in_array($value, $stopword))
      {
         $kata[] = $value;
      }
   }
   return $kata;
}
```

Kode Sumber 4.3.6 Proses Penghapusan *Stopword* pada Isi

Pada Kode Sumber 4.3.4 terdapat fungsi melakukan proses membaca isi berkas *stopword* dan memasukkannya ke dalam suatu variabel *array* sebagai kembalian fungsi untuk digunakan pada proses penghapusan *stopword*.

Setelah dilakukan tokenisasi menggunakan ***preg_split()***, kumpulan kata hasil tokenisasi judul atau isi yang terdapat pada variabel *array* penampung, dibandingkan dengan kumpulan *stopword*. Jika kata hasil tokenisasi tersebut tidak berada pada kumpulan *stopword* maka kata tersebut disimpan pada suatu variabel *array* dan kemudian menjadi kembalian dari fungsi ***tokenisasiJudul()*** dan ***tokenisasiIsi()***. Proses tokenisasi dan penghapusan *stopword* berada pada fungsi yang sama yaitu fungsi ***tokenisasiJudul()*** dan fungsi ***tokenisasiIsi()***.

4.3.3 *Stemming*

```
public function insertKata($kumpulanKata)
{
   foreach ($kumpulanKata as $key => $value)
   {
      $kata = $key;
      $id_kateglo = $this->stemming->getIdKateglo($kata);
      ...
      if ($id_kateglo)
      {
         $leksikal = $this->stemming-
         >getLeksikalKata($id_kateglo);
         $jenis = $this->stemming-
         >getJenisKata($id kateglo);
         ...
      }
      else
      {
         $id kateglo = NULL;
         $leksikal kata = NULL;
         $jenis kata = NULL;
      }
      ...
   }
}
```

Kode Sumber 4.3.7 Proses *Stemming*, Pengambilan Jenis dan Leksikal Kata

Proses *stemming* berada pada fungsi ***insertKata()***, karena setelah proses *stemming* selesai, kata langsung dimasukkan ke dalam basis data. Proses ini memanfaatkan bentuk bagian *model* dari proses implementasi Kateglo, yaitu memanfaatkan fungsi ***getIdKateglo()***, ***getLeksikalKata()*** dan ***getJenisKata()***. Parameter yang dikirim pada fungsi ***insertKata()*** adalah variabel *array* hasil proses tokenisasi judul dan tokenisasi isi yang telah digabung menggunakan fungsi PHP, ***array_merge()***.

Jika kembalian dari fungsi ***getIdKateglo()*** tidak sama dengan *NULL*, kembalian berupa *id* kata tersebut selanjutnya digunakan sebagai parameter untuk mencari leksikal kata dan jenis kata. Jika kembalian sama dengan *NULL*, maka secara otomatis, isi dari variabel *id_kateglo*, *leksikal_kata* dan *jenis_kata* adalah *NULL* sehingga tidak perlu disimpan pada basis data.

4.3.4 Pembentukan Informasi *Co-occurrence*

```
public function insertCooccurrence($jumlah, $kata)
{
   ...
   for ($i = 0; $i < $jumlah; $i++)
   {
      if ($i != $jumlah-1)
      {
         $kataA = $this->mpraproses->getIdKata($kata[$i]);
         $kataB = $this->mpraproses->getIdKata($kata[$i+1]);
         $id_cooccurrence = $this->mpraproses-
         >getIdCooccurrence($kataA, $kataB);
         if (!$id_cooccurrence)
         {
            $frekuensi = 1;
            $data = array('id kataA' => $kataA, 'id kataB' =>
            $kataB, 'frekuensi_cooccurrence' => $frekuensi);
            $insertCooccurrence = $this->mpraproses-
            >insertCooccurrence($data);
         }
         else
         {
            $frekuensi = $this->mpraproses-
            >getFrekuensi($id_cooccurrence);
            $frekuensi = $frekuensi + 1;
            $data = array('frekuensi cooccurrence' =>
            $frekuensi);
            $updateFrekuensi = $this->mpraproses-
            >updateFrekuensi($id_cooccurrence, $data);
         }
      }
   }
   ...
}
```

Kode Sumber 4.3.8 Proses Pembentukan Informasi *Co-occurrence*

Proses pembentukan informasi *co-occurrence* berada pada fungsi ***insertCooccurrence()***, karena setelah proses pembentukan informasi *co-occurrence* selesai dilakukan, pasangan kata hasil pembentukan informasi *co-occurrence* langsung dimasukkan ke dalam basis data. Parameter yang dikirim pada fungsi ***insertCooccurrence()*** adalah variabel jumlah dan variabel *array* kata hasil proses *stemming*.

```
function getIdKata($kata)
{
   $this->db->select('id_kata');
   $this->db->from('kata');
   $this->db->like('nama_kata', $kata, 'none');
   $query = $this->db->get();
   return $query->row('id_kata');
}

function insertCooccurrence($cooccurrence)
{
   if($this->db->insert('cooccurrence',$cooccurrence))
   {
      return true;
   }
   else
   {
      return false;
   }
}

function getIdCooccurrence($kataA, $kataB)
{
   $this->db->select('id cooccurrence');
   $this->db->from('cooccurrence');
   $this->db->where('id_kataA', $kataA);
   $this->db->where('id_kataB', $kataB);
   $query = $this->db->get();
   return $query->row('id_cooccurrence');
}

function getFrekuensi($id)
{
   $data['cooccurrence.id_cooccurrence'] = $id;
   $this->db->select('frekuensi cooccurrence');
   $this->db->from('cooccurrence');
   $this->db->where($data);
   $query = $this->db->get();
   return $query->row('frekuensi_cooccurrence');
}

function updateFrekuensi($id, $data)
{
   $this->db->where('id_cooccurrence', $id);
   $this->db->update('cooccurrence', $data);
}
```

Kode Sumber 4.3.9 *Model* Pembentukan Informasi *Co-occurrence*

Informasi *co-occurrence* terbentuk dari suatu kata dan kata selanjutnya serta frekuensi berapa kali pasangan kata tersebut muncul. Sebagai contoh kata "neraca" muncul setelah kata "defisit", maka informasi *co-occurrence* yang terbentuk adalah (defisit, neraca, 1). Sehingga pada Kode Sumber 4.3.8 digunakan nilai dari variabel jumlah sebagai banyaknya perulangan, banyaknya indeks *i*, yang dilakukan untuk membentuk pasangan kata dari variabel *array* kata. Berdasarkan contoh, misalkan kata "defisit" adalah nilai dari variabel *array* kata dengan indeks ke *i*, *kata[i]*, maka kata "neraca" merupakan nilai dari *kata[i+1]*.

Bentuk model yang digunakan pada pembentukan informasi *co-occurrence* sesuai dengan Kode Sumber 4.3.9, terdapat fungsi ***getIdKata()*** untuk mendapatkan *id* kata artikel berita pada basis data, ***getIdCooccurrence()*** untuk mendapatkan *id* dari pasangan kata *co-occurrence*, ***insertCooccurrence()*** untuk memasukkan hasil pembentukan informasi *co-occurrence*, ***getFrekunesi()*** untuk mendapatkan frekuensi informasi *co-occurrence* dan ***updateFrekuensi()*** untuk memperbarui frekuensi informasi *co-occurrence*.

4.4 Implementasi Pembentukan *Cluster* Kata

Tahapan implementasi dari hasil proses perancangan pembentukan *cluster* kata yang telah dilakukan pada Bab III. Implementasi pembentukan vektor kata dan penghitungan *cosine similarity* menggunakan bahasa pemrograman PHP dan kerangka kerja CodeIgniter (CI). Sedangkan untuk proses implementasi *hierarchical clustering* dan Kullback-Leiber *divergence* menggunakan bahasa pemrograman Python.

4.4.1 Implementasi Pembentukan Vektor Kata

Implementasi yang dapat merepresentasikan bentuk vektor kata pada bahasa pemrograman PHP adalah bentuk *array* dua dimensi. Kode Sumber 4.4.1 berfungsi untuk membentuk vektor kata dari bentuk informasi *co-occurrence* (defisit, neraca, 1) menjadi bentuk $vektorKata$["defisit"]["neraca"] = 1.

```
public function vektor()
{
   $kata_cooccurrence = $this->mtree->getKataCooccurrence();
   $kata_distinct = $this->mtree->getKataDistinct();
   $jumlahKata = count($kata_cooccurrence);
   $jumlahDistinct = count($kata_distinct);

   for ($i = 0; $i < $jumlahDistinct; $i++)
   {
      for ($j = 0; $j < $jumlahKata; $j++)
      {
         if ($kata_distinct[$i]['kata'] ==
         $kata cooccurrence[$j]['kataA'])
         {
            $vektorKata[$kata_distinct[$i]['kata']][$kata_coo
            ccurrence[$j]['kataB']] =
            $kata_cooccurrence[$j]['frekuensi_cooccurrence'];
         }
      }
   }
   ...
}
```

Kode Sumber 4.4.1 Proses Pembentukan Vektor Kata

```
function getKataCooccurrence()
{
   $this->db->select('cooccurrence.frekuensi_cooccurrence,
   k1.id_kata as id_kataA, k2.id_kata as id_kataB,
   k1.nama_kata as kataA, k2.nama_kata as kataB');
   $this->db->from('cooccurrence');
   $this->db->join('kata k1','cooccurrence.id kataA =
   k1.id_kata');
   $this->db->join('kata k2','cooccurrence.id_kataB =
   k2.id_kata');
   return $this->db->get()->result_array();
}

function getKataDistinct()
{
   $this->db->distinct();
   $this->db->select('k1.nama_kata as kata');
   $this->db->from('cooccurrence');
   $this->db->join('kata k1','cooccurrence.id kataA =
   k1.id_kata');
   return $this->db->get()->result_array();
}
```

Kode Sumber 4.4.2 Model Pembentukan Vektor Kata

$vektorKata$["defisit"]["neraca"] = 1 adalah bentuk vektor kata dari kata "defisit", sehingga setiap nilai indeks pertama dari variabel *vektorKata* adalah nilai dari *registered word*, sedangkan indeks kedua merupakan nilai dari kata *co-occurrence* dan nilai 1 adalah nilai frekuensi kemunculan pasangan kata tersebut.

Bentuk model yang digunakan pada proses pembentukan vektor kata dapat dilihat pada Kode Sumber 4.4.2. Terdapat fungsi ***getKataCooccurrence()*** untuk mendapatkan semua kata *co-occurrence* dan frekuensi kemunculannya dari setiap *registered word* dan fungsi ***getKataDistinct()*** untuk mendapatkan kata *registered word* saja.

4.4.2 Implementasi Penghitungan *Cosine Similarity*

Terdapat beberapa fungsi yang dibuat untuk melakukan penghitungan *cosine similarity*, yaitu fungsi ***dotProduct()***, ***absVektorKata()*** dan fungsi ***similaritas()***.

Fungsi ***dotProduct()*** membutuhkan parameter variabel *array* dari vektor kata A dan vektor kata B, di mana setiap variabel *array* vektor kata terdapat kata *co-occurrence* dari suatu *registered word* dan frekuensi dari kemunculan pasangan kata. Kembalian dari fungsi ini adalah nilai hasil penghitungan *dot product* jika terdapat kesamaan kata *co-occurrence* dari vektor kata A dan vektor kata B, dapat dilihat pada Kode Sumber 4.4.3.

Fungsi ***absVektorKata()***, dapat dilihat pada Kode Sumber 4.4.4, adalah untuk menghitung panjang vektor kata dengan menggunakan rumus penghitungan *euclidean distance*. Parameter yang dibutuhkan adalah variabel *array* vektor kata. Kembalian dari fungsi ini adalah nilai hasil penghitungan panjang suatu vektor kata.

Sedangkan untuk fungsi ***similaritas()*** digunakan untuk menggabungkan dua fungsi di atas sehingga sesuai dengan rumus penghitungan *cosine similarity*. Parameter yang dibutuhkan adalah variabel *array vektorKataA* dan *vektorKataB* dan variabel *kataA*, dapat dilihat pada Kode Sumber 4.4.5. Parameter berupa

variabel *kataA*, bernilai kata *registered word*, digunakan ketika nilai hasil penghitungan *cosine similarity* adalah 0 yang disebabkan tidak adanya kesamaan antar kata *co-occurrence vektorkataA* dan *vektorKataB* ketika proses penghitungan *dot product*. Variabel *kataA* dibandingkan terhadap kata *co-occurrence* dari *vektorKataB*. Jika *kataA* terdapat pada *vektorKataB*, maka yang digunakan sebagai pengganti nilai *dot product* adalah frekuensi kemunculan dari kata *co-occurrence* yang sama dengan *kataA*, sedangkan jika tidak terdapat kesamaan maka nilai *cosine similarity* tetap 0. Kembalian fungsi similaritas adalah nilai dari hasil penghitungan *cosine similarity*.

```
Public function dotProduct(array $vektorKataA, array
$vektorKataB)
{
   $hasil = 0;
   foreach (array_keys($vektorKataA) as $key1)
   {
      foreach (array_keys($vektorKataB) as $key2)
      {
         if ($key1 === $key2)
         {
            $hasil = $hasil + $vektorKataA[$key1] *
            $vektorKataB[$key2];
         }
      }
   }
   return $hasil;
}
```

Kode Sumber 4.4.3 Fungsi Penghitungan *Dot Product*

```
Public function absVektorKata(array $vektorKata)
{
   $hasil = 0;
   foreach (array_values($vektorKata) as $value)
   {
      $hasil = $hasil + $value * $value;
   }
   return sqrt($hasil);
}
```

Kode Sumber 4.4.4 Fungsi Penghitungan Panjang Vektor Kata

```
public function similaritas(array $vektorKataA, array
$vektorKataB, $kataA)
{
   $hasil = $this->dotProduct($vektorKataA, $vektorKataB) /
   ($this->absVektorKata($vektorKataA) * $this-
   >absVektorKata($vektorKataB));
   if ($hasil == 0)
   {
      $exist = array_key_exists($kataA, $vektorKataB);
      if ($exist != false)
      {
         $hasilBaru = $vektorKataB[$kataA] / ($this-
         >absVektorKata($vektorKataA) * $this-
         >absVektorKata($vektorKataB));
      }
      else
      {
         $hasilBaru = 0;
      }
      return $hasilBaru;
   }
   else
   {
      return $hasil;
   }
}
```

Kode Sumber 4.4.5 Fungsi Similaritas *Cosine Similarity*

Proses penghitungan *cosine similarity* berada pada satu fungsi yang sama dengan proses pembentukan vektor kata, yaitu fungsi ***vektor()***. Penghitungan *cosine similarity* dilakukan pada semua kata yang memiliki bentuk vektor kata.

Setelah didapatkan nilai penghitungan *cosine similarity*, dilakukan normalisasi yaitu mengurangkan angka 1 dengan nilai hasil penghitungan, tujuannya adalah mendapatkan nilai yang dapat digunakan pada proses *hierarchical clustering*, karena pada *hierarchical clustering*, semakin kecil jarak antar suatu vektor kata, maka semakin dekat jarak antar vektor kata tersebut. Sedangkan hasil penghitungan *cosine similarity*, semakin besar hasil penghitungan, maka semakin dekat jarak antar vektor kata, sehingga hal tersebut menyebabkan diperlukan normalisasi.

Sebagai contoh proses normalisasi, terdapat nilai *cosine similarity* 0,23185, nilai normalisasinya adalah $1 - 0{,}23185 =$

0,76815. Kemudian nilai dari *cosine similarity* dan hasil normalisasi disimpan ke dalam basis data dengan menyertakan *identifier* dari pasangan kata yang dilakukan penghitungan *cosine similarity*.

```
public function vektor()
{
   ...
   foreach ($vektorKata as $kata => $arrayCooccurrence)
   {
      foreach ($arrayCooccurrence as $kataCooccurrence =>
      $frekuensiCooccurrence)
      {
         for ($i = 0; $i < $jumlahIndex; $i++)
         {
            if ($kataCooccurrence == $index[$i])
            {
               $arrayA = $vektorKata[$kata];
               $arrayB = $vektorKata[$kataCooccurrence];
               $similaritas = $this->similaritas($arrayA,
               $arrayB, $kata, $kataCooccurrence);
            }
         }
      }
   }
   ...
}
```

Kode Sumber 4.4.6 Proses Penghitungan *Cosine Similarity*

4.4.3 Implementasi *Hierarchical Clustering*

Implementasi *hierarchical clustering* menggunakan bahasa pemrograman Python dengan pustaka NumPy untuk melakukan pembangkitan *array* dan pustaka SciPy untuk melakukan proses *hierarchical clustering*. Berikut beberapa alasan untuk tidak menggunakan bahasa pemrograman PHP:

1. Keterbatasan penggunaan memori pada bahasa pemrograman PHP. PHP membutuhkan kapasistas memori yang cukup besar untuk melakukan proses pembangkitan *array*. Sebagai contoh 100.000 *integers* dimasukkan ke dalam sebuah *array* yang seharusnya hanya membutuhkan 800.000 *bytes* (1 *integers* adalah 8

bytes) ketika program dijalankan, memori yang dibutuhkan mencapai 14.649.024 *bytes* atau sekitar 13,97 *megabytes* [19]. Sehingga tidak memungkinkan untuk melakukan pembangkitan *array* penampung nilai hasil penghitungan *cosine similarity* berdasarkan jumlah kata hasil ekstraksi menggunakan bahasa pemrograman PHP.

2. Lamanya waktu pemrosesan yang dilakukan hanya untuk melakukan pembangkitan *array*.

Terdapat batasan-batasan yang ditentukan terhadap data hasil penghitungan *cosine similarity* yang digunakan pada proses *clustering* dengan tujuan untuk melakukan pengurangan jumlah kata yang akan dikelompokkan pada *cluster-cluster*. Batasan-batasan tersebut adalah sebagai berikut:

1. Nilai *cosine similarity* yang digunakan adalah nilai dari hasil penghitungan pasangan kata dengan leksikal "Nomina" dan "Nomina" karena pasangan kata tersebut merupakan pasangan kata yang paling sering muncul.
2. Rentang nilai *cosine similarity* yang digunakan dari pasangan kata tersebut adalah nilai antara 0,6 hingga 1,0 karena persebaran paling banyak pada rentang nilai tersebut.

Batasan-batasan tersebut ditentukan berdasarkan hasil dari proses pembentukan informasi *co-occurrence* pasangan kata berdasarkan leksikal dan proses penghitungan *cosine similarity*. Data hasil pembentukan informasi *co-occurrence* dan persebaran nilai *cosine similarity* ditampilkan pada tabel berikut:

Tabel 4.4.1 Jumlah Pasangan Kata Berdasarkan Leksikal

Registered Word	Kata *Co-occurrence*	Jumlah Artikel Berita				
		3.000	5.313	7.563	8.813	10.813
Adjektiva	Adjektiva	2.194	3.096	4.415	4.822	5.702
	Adverbia	90	121	171	199	243
	Lain-lain	580	855	1.153	1.301	1.756
	Nomina	12.889	18.524	26.849	28.627	34.769

Registered Word	Kata *Co-occurrence*	Jumlah Artikel Berita				
		3.000	5.313	7.563	8.813	10.813
	Numeralia	126	196	288	300	355
	Pronomina	74	104	129	157	190
	Verba	5.834	9.047	12.819	13.888	16.549
Adverbia	Adjektiva	83	120	168	189	217
	Adverbia	10	11	11	11	14
	Lain-lain	20	26	36	47	55
	Nomina	455	626	911	1023	1261
	Numeralia	5	7	10	13	15
	Pronomina	3	4	4	5	6
	Verba	279	400	553	623	739
Lain-lain	Adjektiva	450	668	865	962	1152
	Adverbia	20	32	38	48	59
	Lain-lain	193	258	307	357	426
	Nomina	2.904	4.276	5.661	6.184	7.551
	Numeralia	23	42	57	60	72
	Pronomina	18	24	30	39	44
	Verba	1.155	1.809	2.349	2.626	3.230
Nomina	Adjektiva	12.560	17.832	25.544	27.324	32.732
	Adverbia	503	708	1034	1136	1393
	Lain-lain	2.967	4.350	5.804	6.426	7.883
	Nomina	79.255	111.893	156.143	164.897	203.157
	Numeralia	842	1.371	1.875	1.953	2.404
	Pronomina	313	426	549	658	777
	Verba	30.224	46.732	65.339	69.734	85.526
Numeralia	Adjektiva	98	165	238	255	290
	Adverbia	6	7	7	7	9
	Lain-lain	26	54	73	78	88
	Nomina	978	1.640	2.254	2.343	2.777
	Numeralia	22	27	36	38	43
	Pronomina	4	4	6	8	9
	Verba	306	610	811	860	1.026
Pronomina	Adjektiva	73	103	122	152	175
	Adverbia	4	4	5	7	10
	Lain-lain	19	25	30	36	42
	Nomina	272	365	459	553	677
	Numeralia	1	1	4	6	7
	Pronomina	4	6	6	8	8
	Verba	170	232	293	342	401
Verba	Adjektiva	3.271	5.068	7.011	7.695	9.215

Registered Word	Kata *Co-occurrence*	Jumlah Artikel Berita				
		3.000	5.313	7.563	8.813	10.813
	Adverbia	171	252	323	361	427
	Lain-lain	862	1.348	1.782	2.028	2.479
	Nomina	30.257	45.213	62.865	66.793	81.772
	Numeralia	318	507	692	737	887
	Pronomina	120	164	188	234	286
	Verba	8.255	13.977	18.775	20.308	24.719

Tabel 4.4.2 Rentang Persebaran Nilai *Cosine Similarity* Pasangan Kata Leksikal "Nomina"

Rentang Cosine Similarity	Jumlah Artikel Berita				
	3.000	5.313	7.563	8.813	10.813
$0,0 < x \leq 0,1$	8	10	19	19	31
$0,1 < x \leq 0,2$	23	27	32	34	47
$0,2 < x \leq 0,3$	35	43	67	66	103
$0,3 < x \leq 0,4$	68	82	146	157	206
$0,4 < x \leq 0,5$	88	118	227	235	281
$0,5 < x \leq 0,6$	204	242	490	510	621
$0,6 < x \leq 0,7$	543	685	1.101	1.124	1.475
$0,7 < x \leq 0,8$	1.369	2.223	3.243	3.337	4.618
$0,8 < x \leq 0,9$	6.301	10.480	15.758	16.748	22.368
$0,9 < x \leq 1,0$	48.599	76.496	110.129	117.152	147.314

```
...
for i in range(0, countList):
   for j in range(0, countList):
      if (i == j):
         distanceMatrix[i][j] = 0
      else:
         if(str(normalValue.get((listMerge[i],listMerge[j]))
         ) != 'None'):
            distanceMatrix[i][j] =
            normalValue.get((listMerge[i],listMerge[j]))
cluster = cls.linkage(distanceMatrix, method='single',
metric='euclidean')
cut_tree = cls_new.cut_tree(cluster,height=1.393)
...
```

Kode Sumber 4.4.7 Proses *Hierarchical Clustering*

Variabel *distanceMatrix* merupakan hasil pembangkitan *array* menggunakan fungsi ***np.ones()*** dari pustaka NumPy. Isi

dari tiap elemen *distanceMatrix* setelah pembangkitan hanya bernilai 1. Kode Sumber 4.4.7 digunakan untuk mengisi elemen-elemen *distanceMatrix* dengan nilai hasil penghitungan *cosine similarity*.

Setelah elemen *distanceMatrix* terisi dengan nilai hasil penghitungan *cosine similarity*. Dilakukan *clustering* dengan fungsi ***linkage()*** dari modul ***scipy.cluster.hierarchy*** menggunakan *distanceMatrix* sebagai parameter pada fungsi ***linkage()***. Hasil proses *clustering* disimpan pada variabel *cluster* untuk dikirimkan sebagai parameter pada proses pembagian *cluster* menggunakan fungsi ***cut_tree()***. Hasil pembagian *cluster* digunakan untuk melakukan penghitungan Kullback-Leiber *divergence* antar kata pada satu *cluster*.

4.4.4 Implementasi Kullback-Leiber *divergence*

Fungsi ***klb()*** adalah fungsi untuk melakukan penghitungan nilai Kullback-Leiber *divergence*. Terdapat empat parameter, yaitu *arrayPA*, *arrayPB*, *sumFA* dan *sumFB*.

Parameter *arrayPA* dan *arrayPB* berisi frekuensi kata *co-occurrence* yang sama antara vektor kata A dan vektor kata B. sedangkan parameter *sumFA* dan *sumFB* berisi hasil penjumlahan frekuensi kata *co-occurrence* dari vektor kata A dan B.

```
Def klb(arrayPA, arrayPB, sumFA, sumFB):
  kl = 0.0
  if (len(arrayPA) != len(arrayPB)):
    if (len(arrayPB) == 0):
      tempA = ((arrayPA[0]*1.0)/sumFA)
      tempB = 1
      kl = (tempA*(log(tempA/tempB)))
  else:
    for i in range(len(arrayPA)):
      tempA = ((arrayPA[i]*1.0)/sumFA)
      tempB = ((arrayPB[i]*1.0)/sumFB)
      kl = kl + (tempA*(log(tempA/tempB)))
  return kl
```

Kode Sumber 4.4.8 Proses Penghitungan Kullback-Leiber *divergence*

$$P_{A_i} \log \frac{P_{A_i}}{P_{B_i}} = \begin{cases} 0 & \text{jika } P_{A_i} = 0 \\ P_{A_i}(\log P_{A_i}) & \text{jika } P_{B_i} = 0 \end{cases}$$

Terdapat dua kondisi pada penghitungan fungsi ***klb()***, berdasarkan syarat Kullback-Leiber *divergence* di atas, yaitu jika banyak elemen pada *arrayPA* dan *arrayPB* tidak sama, dilakukan pengecekan banyak elemen pada parameter *arrayPB*, jika *arrayPB* tidak memiliki elemen, maka penghitungan Kullback-Leiber *divergence* hanya melibatkan parameter *arrayPA* dan *sumFA*.

4.5 Implementasi Penampilan Hasil *Clustering*

Tahap implementasi dari penampilan hasil *clustering* yang telah dilakukan pada Bab III. Implementasi pembangkitan JSON menggunakan hasil dari proses pembagian *clustering* dengan bahasa pemrograman Python.

Kode Sumber 4.5.1 merupakan proses pembangkitan berkas JSON menggunakan hasil pembagian *cluster*. Jika terdapat kelompok kata di dalam satu *cluster*, maka *child* pada parameter JSON berisi nama dari kata-kata kelompok tersebut.

```
...
dictionary = {}
  dictionary["name"] = "cluster"
  dictionary["children"] = []
  for i in enumerate(word):
    if (len(i[1]) != 1):
      response  = []
      for j in range(len(i[1])):
        response.append({"name": word[i[0]][j], "size" :
         500})
       dictionary["children"].append({"children": response})
...
path = 'C:/xampp/htdocs/thesaurus/json/'
  fileName = os.path.join(path, 'data.json')
  with open(fileName, 'w') as outfile:
    json.dump(dictionary, outfile)
...
```

Kode Sumber 4.5.1 Proses Pembangkitan Berkas JSON

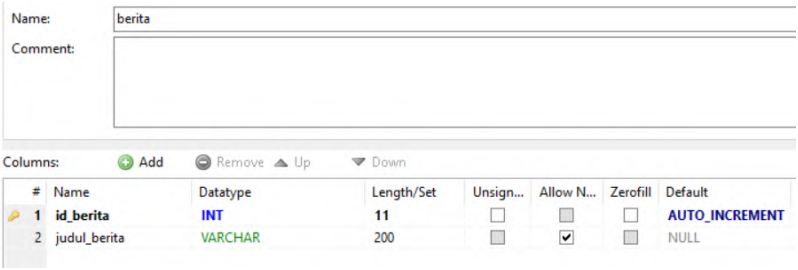
```
{
  "name": "cluster",
  "children": [
    {
      "children": [
        {
          "name": "defisit",
          "size": 500
        },
        {
          "name": "neraca",
          "size": 500
        },
        {
          "name": "pendapatan",
          "size": 500
        }
      ]
    }
  ]
}
```

Kode Sumber 4.5.2 Contoh Hasil Pembangkitan Berkas JSON

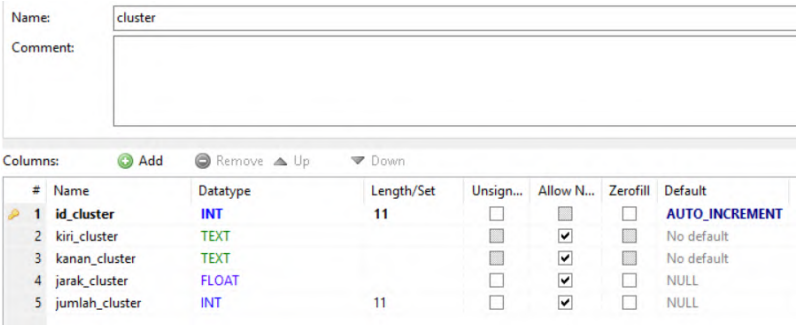
Contoh dari isi berkas JSON yang dibangkitkan dapat dilihat pada Kode Sumber 4.5.2. Berdasarkan informasi di atas, kata “defisit”, “neraca” dan “pendapatan” terdapat pada *child* yang sama sehingga ketika ditampilkan, kata-kata tersebut tedapat pada satu *cluster*.

4.6 Implementasi Basis Data

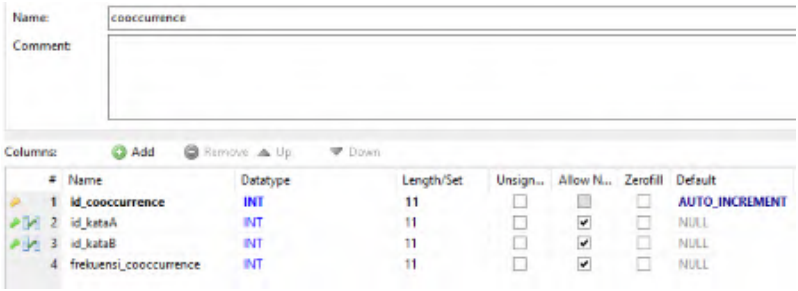
Tahapan implementasi dari hasil proses perancangan basis data yang telah dilakukan pada Bab III menggunakan MySQL dengan kakas bantu HeidiSQL. Berdasarkan hasil perancangan pada CDM dan PDM basis data yang telah dilakukan, terdapat enam tabel yang digunakan untuk melakukan penyimpanan data ketika dilakukan proses-proses untuk membangun tesaurus dan ketika pemakaian tesaurus, yaitu tabel berita, tabel cluster, tabel cooccurrence, tabel kata, tabel kullback dan tabel similaritas. Berikut adalah gambar dari masing-masing tabel yang telah diimplementasi pada MySQL.



Gambar 4.6.1 Implementasi Tabel Berita



Gambar 4.6.2 Implementasi Tabel Cluster



Gambar 4.6.3 Implementasi Tabel Cooccurrence

Name: kata

Comment:

Columns: Add Remove Up Down

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	**id_kata**	**INT**	**11**	☐	☐	☐	**AUTO_INCREMENT**
2	id_kateglo	INT	11	☐	☑	☐	NULL
3	dasar_kata	VARCHAR	50	☐	☑	☐	NULL
4	nama_kata	VARCHAR	100	☐	☑	☐	NULL
5	jenis_kata	VARCHAR	15	☐	☑	☐	NULL
6	leksikal_kata	VARCHAR	10	☐	☑	☐	NULL
7	tf_kata	INT	11	☐	☑	☐	NULL
8	df_kata	INT	11	☐	☑	☐	NULL

Gambar 4.6.4 Implementasi Tabel Kata

Name: kullback

Comment:

Columns: Add Remove Up Down

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	**id_kullback**	**INT**	**11**	☐	☐	☐	**AUTO_INCREMENT**
2	id_kataA	INT	11	☐	☑	☐	NULL
3	id_kataB	INT	11	☐	☑	☐	NULL
4	nilai_kullback	FLOAT		☐	☑	☐	NULL

Gambar 4.6.5 Implementasi Tabel Kullback

Name: similaritas

Comment:

Columns: Add Remove Up Down

#	Name	Datatype	Length/Set	Unsign...	Allow N...	Zerofill	Default
1	**id_similaritas**	**INT**	**11**	☐	☐	☐	**AUTO_INCREMENT**
2	id_cooccurenceSim	INT	11	☐	☑	☐	NULL
3	nilai_similaritas	FLOAT		☐	☑	☐	NULL
4	normalisasi_similaritas	FLOAT		☐	☑	☐	NULL

Gambar 4.6.6 Implementasi Tabel Similaritas

4.7 Implementasi Antar Muka

Tahapan implementasi antar muka dilakukan pada bagian *view* dari CodeIgniter (CI) dengan membuat halaman tampilan sesuai dengan perancangan pada Bab III. Halaman yang dibuat adalah halaman utama atau halaman antar muka pencarian kata, halaman antar muka hasil pencarian kata dan halaman antar muka visualisasi hasil *clustering*. Ketiga halaman tersebut diimplementasikan menggunakan HTML, PHP dan JavaScript. JavaScript digunakan untuk melakukan implementasi penggunaan D3.js, sedangkan HTML dan PHP untuk implementasi tampilan yang dibuat sebagai antar muka pengguna.

(Halaman ini sengaja dikosongkan)

BAB V
UJI COBA DAN EVALUASI

Bab ini membahas mengenai uji coba dan evaluasi terhadap tesaurus yang dibangun berdasarkan implementasi yang telah dilakukan. Terdiri dari deskripsi uji coba yang dilakukan dan skenario-skenario untuk menguji kebenaran tesaurus.

5.1 Deskripsi Uji Coba

Deskripsi uji coba menjelaskan lingkungan uji coba, dataset uji coba, *threshold cluster* yang digunakan untuk melakukan pengujian terhadap tesaurus yang dibangun. Lingkungan uji coba meliputi perangkat keras dan perangkat lunak yang digunakan, sedangkan dataset uji coba meliputi penjelasan tentang data yang digunakan pada uji coba tesaurus.

5.1.1 Lingkungan Uji Coba

Lingkungan uji coba yang digunakan untuk menguji tugas akhir ini adalah sebagai berikut:

1. Perangkat Keras
   Tipe : ASUS ROG GL552VW
   Prosesor : Inter® Core(TM) i7-6700HQ CPU (8 CPUs) @ 2.6GHz
   Memori (RAM) : 8192 MB
2. Perangkat Lunak
   Sistem operasi : Windows 10 Enterprise 64-bit
   Web Server : Apache 2.4.4
   Basis Data : MySQL 5.5.32

5.1.2 Dataset Uji Coba

Data yang digunakan untuk membangun dan melakukan uji coba pada tesaurus adalah berkas xml kumpulan artikel berita dari situs portal berita Kompas. Rentang tanggal berita yang diambil adalah antara bulan Januari 2011 hingga bulan September 2015. Terdapat lima topik berita, yaitu ekonomi, hiburan, olahraga, politik dan teknologi.

Artikel berita dibagi ke dalam beberapa jumlah tertentu yaitu 3.000 xml, 5.313 xml, 7.563 xml, 8.813 xml dan 10.813 xml diambil secara acak dari kelima kategori berita. Terdapat juga berita sejumlah 750 xml diambil dari artikel berita olahraga dan 4.500 xml yang diambil dari artikel berita bertopik hiburan.

Pembagian banyak jumlah data dan pembagian artikel berdasarkan topik berita bertujuan untuk pembagian tersebut digunakan ketika melakukan uji coba terhadap tesaurus. Jumlah data digunakan untuk mengetahui keluaran tesaurus jika menggunakan banyak data yang berbeda, sedangkan topik berita digunakan untuk mengetahui keluaran tesaurus dipengaruhi oleh topik berita yang digunakan untuk membangun tesaurus.

5.1.3 *Threshold Cluster*

Pembatasan nilai parameter *height* pada fungsi ***cut_tree()*** bertujuan untuk menentukan jumlah *cluster* yang dibentuk, pembagian *cluster* setelah terbentuknya satu *cluster* besar hasil dari proses *hierarchical clustering*.

Berdasarkan persebaran data nilai jarak antar *cluster* pada basis data proses *hierarchical clustering*, data tersebar antara rentang nilai 0,8 hingga 2,8 dengan persebaran data paling banyak antara 1,3 hingga 1,42, yaitu 7.095 dari 8.460 pada data artikel berita sebanyak 10.813 xml.

Setelah mengetahui persebaran data terbanyak, didapatkan pembentukan *cluster* yang optimal terletak di antara nilai 1,3 hingga 1,4. Jika parameter *height* menggunakan nilai di atas 1,4 maka *cluster* yang terbentuk berupa satu *cluster* besar yang beranggotakan ribuan kata dan beberapa *cluster* kecil yang beranggotakan tidak lebih dari 10 kata. Sedangkan jika menggunakan nilai di bawah 1,3 maka *cluster* yang terbentuk memiliki anggota kurang dari 10 kata.

Selanjutnya, penentuan nilai di antara rentang 1,3 hingga 1,4 bergantung dengan jumlah nilai jarak terbanyak yang terdapat pada rentang tersebut, misalnya terdapat nilai jarak 1,3814

sebanyak 500, maka nilai tersebut yang akan digunakan sebagai nilai parameter *height*.

5.2 Skenario Uji Coba 1

Skenario uji coba 1 adalah skenario penghitungan waktu eksekusi program yang dibutuhkan untuk membangun tesaurus. Skenario ini bertujuan untuk mengetahui jumlah waktu yang dibutuhkan untuk membangun tesaurus menggunakan setiap suatu jumlah data tertentu.

Uji coba dilakukan sebanyak lima kali dengan menggunakan jumlah data yang telah dibagi sebelumnya, yaitu 3.000 xml, 5.313 xml, 7.563 xml, 8.813 xml dan 10.813 xml dengan topik berita acak. Catat setiap waktu yang dibutuhkan pada suatu proses dalam setiap uji coba yang dilakukan. Proses-proses yang dimaksud adalah proses ekstraksi kata, pembentukan informasi *co-occurrence*, penghitungan *cosine similarity*, *clustering* dan proses penghitungan Kullback-Leiber *divergence*.

5.2.1 Tahapan Uji Coba

Tahapan-tahapan uji coba yang dilakukan untuk mendapatkan hasil penghitungan waktu proses dari skenario uji coba 1 adalah sebagai berikut:

1. Tahap pertama yang dilakukan adalah memilih jumlah data yang diproses. Sebagai contoh melakukan pemrosesan pada jumlah data 3.000 xml.
2. Tahap selanjutnya adalah memasukkan data sebanyak yang telah ditentukan, 3.000 xml, ke dalam tesaurus dan menjalankan program pembangunan tesaurus.
3. Selanjutnya, catat waktu yang dibutuhkan untuk masing-masing proses, waktu yang dibutuhkan untuk melakukan proses ekstraksi kata, pembentukan informasi *co-occurrence*, penghitungan *cosine similarity*, *clustering* dan penghitungan Kullback-Leiber *divergence*.

4. Setelah program berhenti, lakukan kembali pada tahap 1 terhadap jumlah data selanjutnya, yaitu 5.313 xml, 7.563 xml, 8.813 xml dan 10.813 xml.

5.2.2 Hasil Uji Coba

Tabel 5.2.1 Tabel Hasil Waktu Eksekusi

Jumlah Artikel	Nama Proses	Waktu Eksekusi (menit)
3.000 xml	Ekstraksi Kata	980
	Pembentukan Informasi *Co-occurrence*	484
	Penghitungan *Cosine Similarity*	222
	Clustering	6
	Penghitungan Kullback-Leiber *Divergence*	13
5.313 xml	Ekstraksi Kata	1.735
	Pembentukan Informasi *Co-occurrence*	857
	Penghitungan *Cosine Similarity*	393
	Clustering	10
	Penghitungan Kullback-Leiber *Divergence*	22
7.563 xml	Ekstraksi Kata	2.470
	Pembentukan Informasi *Co-occurrence*	1.220
	Penghitungan *Cosine Similarity*	560
	Clustering	15
	Penghitungan Kullback-Leiber *Divergence*	32
8.813 xml	Ekstraksi Kata	2.878
	Pembentukan Informasi *Co-occurrence*	1.422
	Penghitungan *Cosine Similarity*	653
	Clustering	17
	Penghitungan Kullback-Leiber *Divergence*	37
10.813 xml	Ekstraksi Kata	3.531
	Pembentukan Informasi *Co-occurrence*	1.744
	Penghitungan *Cosine Similarity*	801
	Clustering	21
	Penghitungan Kullback-Leiber *Divergence*	46

Pada data sejumlah 3.000 xml artikel berita, total waktu yang dibutuhkan untuk membangun tesaurus adalah 1.705 menit atau 28,417 jam. Proses ekstraksi kata membutuhkan 980 menit, proses pembentukan informasi *co-occurrence* membutuhkan 484 menit, penghitungan *cosine similarity* membutuhkan 222 menit, proses *clustering* membutuhkan waktu 6 menit dan penghitungan Kullback-Leiber *divergence* membutuhkan waktu 13 menit.

Sedangkan data yang dihasilkan oleh tesaurus dari proses pembangunan tesaurus tersebut terdapat pada Tabel 5.2.2. Pada data sejumlah 3.000 xml artikel berita, data yang dihasilkan adalah 17.031 kata, 221.449 informasi *co-occurrence* kata, 212.340 nilai penghitungan *cosine similarity*, 4.976 *cluster*, 44.230 nilai penghitungan Kullback-Leiber *divergence*. Data yang dihasilkan berbanding lurus dengan banyaknya artikel berita yang digunakan.

Tabel 5.2.2 Tabel Jumlah Data yang Dihasilkan

Jumlah Artikel	Nama Data	Jumlah Data
3.000 xml	Kata	17.031
	Informasi *Co-occurrence*	221.449
	Cosine Similarity	212.340
	Cluster	4.976
	Kullback-Leiber *divergence*	44.230
5.313 xml	Kata	20.035
	Informasi *Co-occurrence*	320.334
	Cosine Similarity	311.217
	Cluster	6.890
	Kullback-Leiber *divergence*	60.680
7.563 xml	Kata	22.706
	Informasi *Co-occurrence*	446.192
	Cosine Similarity	437.064
	Cluster	7.261
	Kullback-Leiber *divergence*	78.890
8.813 xml	Kata	24.329
	Informasi *Co-occurrence*	474.920
	Cosine Similarity	465.784
	Cluster	7.589
	Kullback-Leiber *divergence*	95.390

Jumlah Artikel	Nama Data	Jumlah Data
10.813 xml	Kata	25.829
	Informasi *Co-occurrence*	579.634
	Cosine Similarity	570.504
	Cluster	8.460
	Kullback-Leiber *divergence*	100.058

5.2.3 Analisis dan Evaluasi

Pada hasil skenario uji coba 1 didapatkan hasil waktu yang *linear* atau berbanding lurus, yaitu semakin banyak data yang diproses, maka semakin lama pula waktu yang dibutuhkan untuk menjalankan program. Beberapa hal yang dapat diambil dari hasil uji coba di atas selain waktu yang dibutuhkan adalah sebagai berikut:

1. Proses yang membutuhkan waktu paling lama adalah proses ektraksi kata dari artikel berita karena proses tersebut terdiri dari beberapa bagian proses, seperti proses untuk memuat berkas xml yang digunakan ke dalam tesaurus, kemudian melakukan proses tokenisasi, penghapusan *stopword* dan proses *stemming*. Selain proses-proses tersebut, ekstraksi kata juga melakukan proses untuk memasukkan kata hasil ekstraksi ke dalam basis data dan menghapus kata yang tidak memiliki *identifier* atau *id* berdasarkan Kateglo dari basis data.
2. Proses yang membutuhkan waktu paling cepat adalah proses *clustering*, karena proses tersebut hanya terdiri dari satu proses saja yaitu untuk mengelompokkan kata. Proses *clustering* berjalan sangat cepat karena dibangun menggunakan pustaka bahasa pemrograman Python, yaitu ***scipy.cluster.hierarchy***.
3. Tiga proses awal, yaitu proses ekstraksi kata, pembentukan informasi *co-occurrence*, penghitungan *cosine similarity* dibangun dengan menggunakan bahasa pemrograman PHP, sedangkan dua proses selanjutnya menggunakan bahasa pemrograman Python. Perbedaan waktu yang cukup jauh dapat dipengaruhi oleh cara

membangun program dari proses tersebut. Semua proses yang menggunakan PHP dibangun dari awal, atau *build from scratch*. Sedangkan yang menggunakan Python dibangun dengan cara memanfaatkan pustaka bahasa pemrograman dan modul-modul yang sudah ada untuk digunakan kembali, sehingga dapat berjalan lebih cepat.

4. Jumlah data hasil *clustering* terbentuk dari proses *clustering* yang menggunakan pasangan leksikal kata "Nomina" dan rentang nilai *cosine similarity* antara 0,6 hingga 1,0.
5. Informasi *co-occurrence* kata yang terbentuk tidak semuanya memiliki nilai *cosine similarity*, karena terdapat kata-kata yang menjadi kata *co-occurrence* saja, sehingga tidak memiliki bentuk vektor kata yang digunakan untuk menghitung nilai *cosine similarity*.

5.3 Skenario Uji Coba 2

Skenario uji coba 2 adalah skenario pencarian kata yang bertujuan untuk mengetahui hasil keluaran tesaurus jika terdapat kata masukan sebagai kueri pencarian yang diberikan oleh pengguna untuk melakukan pencarian suatu kata pada banyaknya jumlah data yang berbeda.

Uji coba dilakukan dengan menggunakan 10 kata berbeda dan dilakukan sebanyak lima kali, yaitu pada basis data hasil skenario uji coba 1, basis data 3.000 xml, 5.313 xml, 7.563 xml, 8.813 xml dan 10.813 xml. Catat setiap hasil keluaran daftar kata dan nilai kedekatan yang diberikan oleh tesaurus.

5.3.1 Tahapan Uji Coba

Tahapan-tahapan uji coba yang dilakukan untuk mendapatkan hasil keluaran tesaurus berupa daftar kata dan nilai kedekatannya berdasarkan diagram alur pada Gambar 3.2.9 adalah sebagai berikut:

1. Membuka halaman tampilan antar muka pencarian kata.
2. Memasukkan kata pencarian pada kotak pencarian.

3. Jika terdapat hasil keluaran, dilakukan pencatatan kata pencarian, kata keluaran dan nilai kedekatan.
4. Jika tidak terdapat hasil keluaran, lakukan kembali tahapan kedua.
5. Selanjutnya, lakukan kembali hingga terdapat 10 kata pencarian dan masing-masing hasil kata keluaran dan nilai kedekatan dari setiap kata pencarian.

5.3.2 Hasil Uji Coba

Tabel 5.3.1 Tabel Hasil Uji Coba Pencarian Kata

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
5.313 xml	Laras	Senapan	-0,026605
		Pistol	-0,001548
		Pengendara	0,014510
		Senjata	0,224436
	Senapan	Peluru	0,038762
		Laras	0,075382
		Perampok	0,090678
		Pistol	0,141990
		Senjata	0,278597
	Prosedur	Dikoordinasikan	-0,063439
		Sop	-0,019114
		Pembagian	-0,002645
		Substansi	0,010323
		Mekanisme	0,012681
	Dampak	Pengendalian	-0,058098
		Pemantauan	-0,009921
		Kekeringan	-0,007307
		Klarifikasi	-0,007307
		Antisipasi	-0,001397
		Pemberitaan	0,028202
	Susunan	Sasaki	-0,053825
		Perselisihan	-0,008385
		Direksi	0,003158
		Kecurigaan	0,014752
		Intervensi	0,015296
		Komisaris	0,054273
		Piala	0,092804
		Penghargaan	0,097542
	Publikasi	Risalah	-0,072397

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Pemberitaan	-0,022829
		Pemanggilan	-0,005636
		Pemantauan	-0,004611
		Cetak	0,011565
		Klarifikasi	0,026623
	Frekuensi	Nilai	0,751785
	Mahasiswa	Katolik	-0,063898
		Elektro	-0,061647
		Fisika	-0,027254
		Ring	-0,022147
		Elemen	-0,022123
		Tesis	-0,016717
		Demo	-0,015890
		Ulama	-0,013577
		Penghapusan	-0,011469
		Dipukuli	-0,011458
		Kristen	-0,011162
		Dragon	-0,010193
		Tanding	-0,010020
		Wulan	-0,009925
		Beo	-0,008463
		Boneka	-0,006483
		Persiapan	-0,005921
		Fisik	-0,005467
		Isi	-0,004532
		Sasana	-0,004337
		Rudi	-0,004258
		Dewi	-0,003472
		Nada	-0,003382
		Rumahnya	-0,002439
		Upah	-0,001530
		Tenaga	-0,001396
		Islam	0,003208
		Latihan	0,005482
		Jurusan	0,021646
		Aksi	0,029158
		Teknik	0,031647
		Kuliah	0,047076
		Universitas	0,060441
	Terkini	Situasi	0,117959

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
	Pemantauan	Pemberitaan	-0,021591
		Klarifikasi	-0,004845
		Pemanggilan	-0,001821
		Pengendalian	-0,000456
		Publikasi	0,005465
		Cetak	0,017609
		Antisipasi	0,156803
		Dampak	0,193315

Tabel 5.3.1 adalah tabel hasil skenario uji coba 2 pada basis data dengan jumlah artikel sebanyak 5.313 xml. Terdapat daftar kata-kata keluaran yang dihasilkan oleh tesaurus dan nilai kedekatannya dengan kata masukan, sebagai contoh kata "terkini" memiliki kedekatan dengan kata "situasi" bernilai **0,117959**. Hasil skenario uji coba 2 pada basis data selain 5.313 xml terlampir pada buku tugas akhir ini, Tabel A.1.

5.3.3 Analisis dan Evaluasi

Pada hasil skenario uji coba 2 yang diberikan tesaurus, didapatkan beberapa hal yang dapat diambil dan dilakukan analisis, yaitu analisis terhadap kata keluaran, nilai kedekatan yang dihasilkan dan pengaruh jumlah data yang digunakan .

Terdapat hasil keluaran nilai kedekatan yang bernilai negatif. Hal tersebut disebabkan oleh penghitungan Kullback-Leiber *divergence* bagian proses penghitungan logaritma pada persamaan (2.7).

Jika nilai P_{B_i} lebih besar dari P_{A_i} maka hasil pembagian bernilai diantara 0 dan 1. Sehingga hal tersebut menyebabkan hasil penghitungan logaritma bernilai negatif. Sebagai contoh, terdapat suatu nilai 0,26534. Hasil penghitungan dari nilai tersebut adalah $\log 0{,}26534 = -0{,}57619$.

Banyaknya data yang digunakan berpengaruh pada hasil yang diberikan oleh tesaurus, pengaruh dalam bentuk jumlah kata yang memiliki nilai kedekatan, kata-kata yang dikeluarkan ataupun dalam bentuk kata-kata yang terdapat dalam satu *cluster*.

Sebagai contoh pencarian kata “senapan” dan “terkini” pada jumlah data yang berbeda, yaitu 5.313 berkas xml dan 7.563 berkas xml.

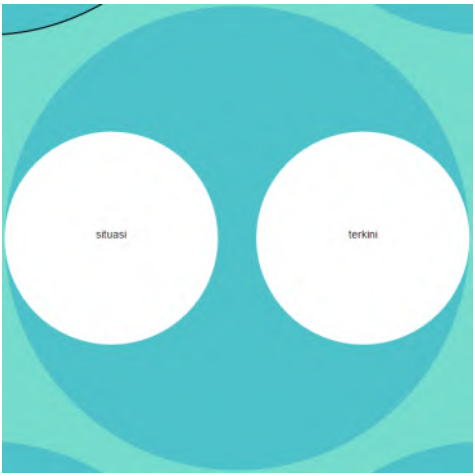
Pada kata “senapan” lima kata yang memiliki nilai Kullback-Leiber *divergence* terdekat pada data 5.313 berkas xml dan 7.563 berkas xml tidak mengalami perbedaan kata, perbedaan hanya terdapat pada nilai kedekatan yang dihasilkan. Sedangkan kata “terkini” pada data 5.313 berkas xml dan 7.563 berkas xml mengalami perbedaan pada jumlah kata keluaran maupun pada nilai kedekatan yang dihasilkan.

Terdapat perberdaan pada jumlah kata-kata yang berada dalam satu *cluster* yang terbentuk dari jumlah data artikel berita yang berbeda. Kata “terkini” pada data yang menggunakan 5.313 berkas xml terdapat pada *cluster* yang memiliki jumlah anggota sebanyak 2 kata, sedangkan pada data yang menggunakan 7.563 berkas xml artikel berita, kata “terkini” terdapat pada *cluster* yang memiliki jumlah anggota sebanyak 5 kata. Kata “senapan” pada data yang menggunakan 5.313 berkas xml terdapat pada *cluster* yang memiliki jumlah anggota sebanyak 9 kata, sedangkan pada data yang menggunakan 7.563 berkas xml artikel berita, kata “senapan” terdapat pada *cluster* yang memiliki jumlah anggota sebanyak 25 kata.

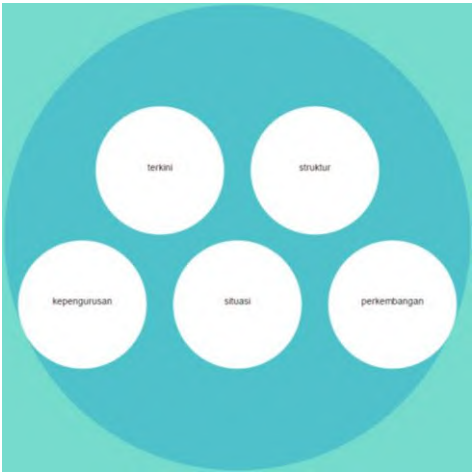
Tabel 5.3.2 Tabel Pengaruh Jumlah Data Terhadap Keluaran

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
5.313 xml	Senapan	Peluru	0,038762
		Laras	0,075382
		Perampok	0,090678
		Pistol	0,141990
		Senjata	0,278597
7.563 xml	Senapan	Peluru	0,050171
		Laras	0,079520
		Perampok	0,090678
		Pistol	0,194526
		Senjata	0,298345
5.313 xml	Terkini	Situasi	0,117959
7.563 xml	Terkini	Struktur	-0,012012

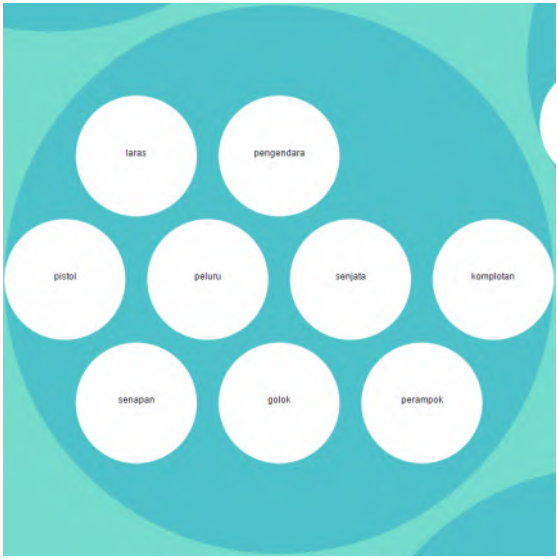
Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Kepengurusan	-0,004003
		Situasi	0,047194
		Perkembangan	0,094652



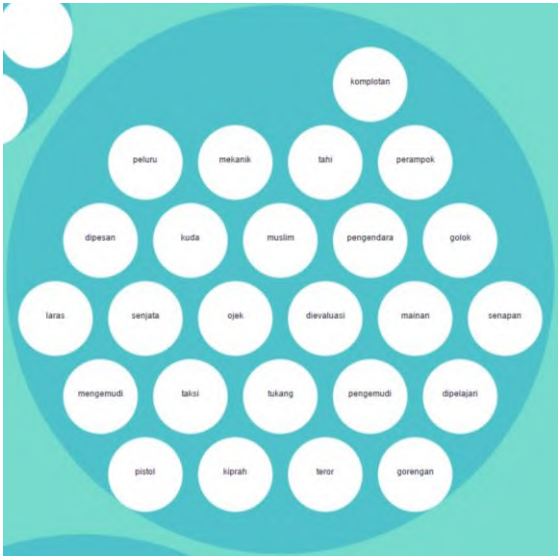
Gambar 5.3.1 *Cluster* Kata Terkini pada Data 5.313 xml



Gambar 5.3.2 *Cluster* Kata Terkini pada Data 7.563 xml



Gambar 5.3.3 *Cluster* Kata Senapan pada Data 5.313 xml



Gambar 5.3.4 *Cluster* Kata Senapan pada Data 7.563 xml

5.4 Skenario Uji Coba 3

Skenario uji coba 3 adalah skenario penggunaan satu topik berita. Skenario ini bertujuan untuk mengetahui hasil keluaran yang dihasilkan oleh tesaurus jika tesaurus dibangun hanya menggunakan satu topik berita saja.

Skenario dilakukan sebanyak dua kali dengan menggunakan dua topik yang berbeda dan jumlah data yang berbeda untuk masing-masing uji coba yang dilakukan, yaitu topik berita olahraga sebanyak 750 xml berita dan topik hiburan sebanyak 4.500 xml berita.

5.4.1 Tahapan Uji Coba

Tahapan-tahapan uji coba yang dilakukan untuk mendapatkan hasil keluaran yang dihasilkan oleh skenario uji coba 3 adalah sebagai berikut:

1. Tahap pertama yang dilakukan adalah memilih topik dan jumlah berita yang digunakan, yaitu olahraga sebanyak 750 xml dan topik hiburan sebanyak 4.500 xml.
2. Tahap selanjutnya yang dilakukan adalah memasukkan data berdasarkan topik sebanyak yang telah ditentukan secara bergantian ke dalam tesaurus, misalnya dilakukan terlebih dahulu terhadap data bertopik olahraga.
3. Kemudian, lakukan tahapan skenario uji coba 2 untuk mendapatkan kata keluaran dan nilai kedekatan kata.
4. Selanjutnya adalah lakukan pencatatan terhadap hasil keluaran yang diberikan oleh tesaurus.
5. Kembali lakukan dari tahap 2 untuk topik hiburan.

5.4.2 Hasil Uji Coba

Tabel 5.4.1 Tabel Hasil Pencarian Satu Topik Berita

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
750 xml	Akibat	Podium	0,048101
Olahraga		Meter	0,063378
		Insiden	0,094569
		Kecelakaan	0,123742

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Lap	0,419648
	Emas	Ilham	-0,061905
		Kemala	-0,056164
		Dada	-0,040375
		Podium	-0,037178
		Kupu	-0,027914
	Fisik	Stamina	0,003120
	Gubernur	Bonus	-0,018825
		Insentif	-0,007525
4.500 xml	Akibat	Paparan	-0,029887
Hiburan		Sanksi	-0,018950
		Pelaku	-0,018941
		Ledakan	-0,017941
		Listrik	-0,017211
	Emas	Poster	-0,056024
		Perhiasan	-0,016249
		Bowo	-0,010526
		Penetapan	-0,006239
		Pemasaran	-0,005249
	Fisik	Idaman	-0,030648
		Tindak	-0,029093
		Perdagangan	-0,028104
		Listrik	-0,027300
		Revolusi	-0,027051
	Gubernur	Kendal	-0,043567
		Camat	-0,037927
		Lurah	-0,031029
		Calon	-0,013817
		Wakit	-0,013780

Tabel 5.4.1 adalah tabel hasil pencarian pada skenario uji coba 3 pada basis data dengan artikel bertopik olahraga sebanyak 750 xml dan artikel bertopik hiburan sebanyak 4.500 xml dengan menggunakan kata-kata masukan “akibat”, “emas”, “fisik” dan “gubernur”. Tabel tersebut digunakan untuk membandingkan

pengaruh topik berita terhadap kata keluaran pada analisis dan evaluasi.

5.4.3 Analisis dan Evaluasi

Pada hasil skenario uji coba 3 didapatkan hasil daftar kata dan nilai kedekatan yang diberikan oleh tesaurus berdasarkan topik berita dan jumlah banyaknya artikel berita yang diproses. Beberapa hal yang dapat diambil dan dilakukan analisis dari hasil uji coba tersebut adalah sebagai berikut:

1. Daftar kata hasil keluaran tesaurus dipengaruhi oleh topik berita yang digunakan untuk membangun tesaurus, sebagai contoh kata "gubernur". Pada topik berita olahraga, kata "gubernur" memiliki nilai kedekatan dengan kata "bonus" dan "insentif". Kedua kata tersebut berhubungan dengan "gubernur" dalam konteks olahraga atau dapat berarti "bonus" atau "insentif" yang diberikan oleh "gubernur" tehadap atlet yang berprestasi. Sedangkan kata "gubernur" pada topik berita hiburan memiliki kedekatan dengan kata-kata yang berhubungan dengan pemerintahan seperti "camat", "lurah", "calon" dan "wakil".
2. Jika dilakukan pencarian menggunakan kata-kata di luar topik yang digunakan untuk membangun tesaurus, hasil keluaran yang diberikan adalah *NULL* yang berarti kata tersebut tidak ada pada tesaurus.

5.5 Skenario Uji Coba 4

Skenario uji coba 4 adalah skenario pembandingan hasil keluaran tesaurus antara tesaurus yang dibangun menggunakan artikel berita dengan topik acak dan tesaurus yang dibangun menggunakan satu topik artikel berita.

Uji coba dilakukan dengan 3 kata yang berbeda pada masing-masing basis data artikel sejumlah 5.313 xml dengan topik acak dan 4.500 xml dengan topik hiburan. Setiap kata keluaran yang dihasilkan oleh tesaurus dilakukan pencatatan.

5.5.1 Tahapan Uji Coba

Tahapan-tahapan uji coba yang dilakukan untuk mendapatkan hasil hasil keluaran yang dihasilkan oleh skenario uji coba 4 adalah sebagai berikut:

1. Tahap pertama yang dilakukan adalah menentukan kata yang digunakan melakukan pencarian terhadap tesaurus.
2. Selanjutnya adalah melakukan pencarian menggunakan kata yang telah ditentukan sebagai kueri pencarian pada basis data 5.313 xml dan 4.500 xml.
3. Catat hasil keluaran yang diberikan oleh tesaurus.
4. Melakukan pembandingan terhadap hasil keluaran.

5.5.2 Hasil Uji Coba

Tabel 5.5.1 adalah tabel hasil pencarian pada skenario uji coba 4 pada basis data dengan artikel bertopik acak sebanyak 5.313 xml dan artikel bertopik hiburan sebanyak 4.500 xml dengan menggunakan kata-kata masukan “efisiensi”, “fraksi” dan “gerakan”. Tabel tersebut digunakan untuk membandingkan pengaruh topik berita dan jumlah data terhadap kata keluaran pada analisis dan evaluasi.

Tabel 5.5.1 Tabel Hasil Topik Acak dan Satu Topik

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
5.313 xml	Efisiensi	Transparasi	-0,047614
Acak		Penghematan	-0,019997
		Efektivitas	-0,011474
		Pupuk	-0,001032
		Pos	0,014510
		Pemangkasan	0,020918
		Subsidi	0,045355
	Fraksi	Partainya	0,691914
	Gerakan	Simbol	-0,018843
		Media	0,024005
4.500 xml	Efisiensi	Penggunaan	0,354541
Hiburan	Fraksi	Simpatisan	-0,026529
		Pengurus	0,514263
		Ketua	0,670787

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Rapat	0,737229
		Anggota	1,155520
	Gerakan	Fenomena	-0,030209
		Kepentingan	-0,023629
		Interaksi	-0,018446
		Media	-0,015005

5.5.3 Analisis dan Evaluasi

Pada hasil skenario uji coba 4 yang diberikan oleh tesaurus, didapatkan beberapa hal yang dapat dilakukan analisis, yaitu analisis terhadap kata keluaran dan nilai kedekatan.

Topik artikel berita yang digunakan berpengaruh pada hasil yang diberikan oleh tesaurus, pengaruh dalam bentuk jumlah kata yang memiliki nilai kedekatan, kata-kata yang keluaran yang dihasilkan oleh tesaurus ataupun dalam bentuk kata-kata yang terbentuk ke dalam satu *cluster*. Sebagai contoh pencarian kata "efisiensi" dan "fraksi" pada topik acak sebanyak 5.313 xml dan topik hiburan sebanyak 4.000 xml.

Terdapat perbedaan pada daftar kata keluaran dan nilai kedekatan ketika menggunakan kata "efisiensi" dan "fraksi" pada basis data 5.313 xml dan 4.500 xml. Perbedaan tersebut dipengaruhi oleh topik berita yang digunakan.

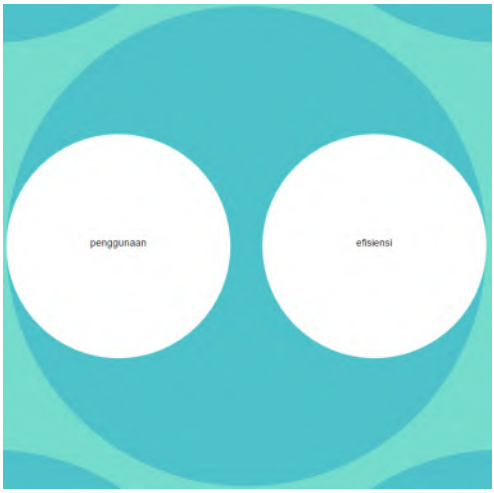
Kata "efisiensi" menghasilkan 7 kata yang memiliki nilai kedekatan pada basis data topik acak 5.313 xml dengan *cluster* yang beranggotakan 12 kata. Sedangkan pada basis data topik hiburan 4.500 xml hanya menghasilkan 1 kata yang memiliki nilai kedekatan dengan *cluster* yang beranggotakan 2 kata.

Kata "fraksi" menghasilkan 5 kata yang memiliki nilai kedekatan pada basis data topik hiburan 4.500 xml dengan *cluster* yang beranggotakan 6 kata. Sedangkan pada basis data topik acak 5.313 xml hanya menghasilkan 1 kata yang memiliki nilai kedekatan dengan *cluster* yang beranggotakan 2 kata.

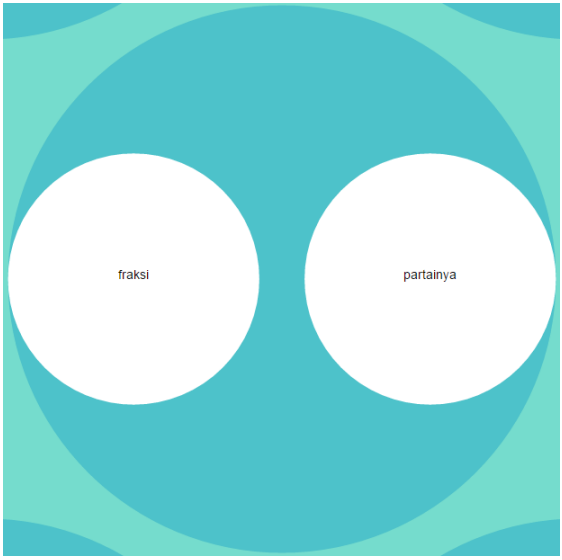
Beradasarkan hal tersebut, dapat ditarik suatu pernyataan bahwa topik berita sangat berpengaruh kepada hasil keluaran yang diberikan oleh tesaurus. Selain topik berita yang digunakan,

banyak artikel yang diproses juga dapat memengaruhi keluaran tesaurus.

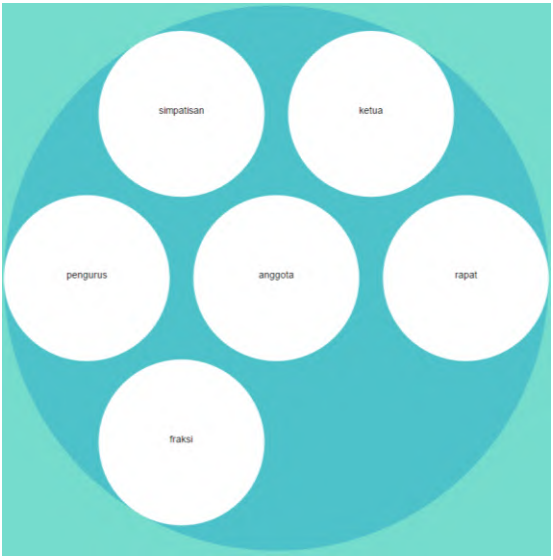
Gambar 5.5.1 *Cluster* Kata Efisiensi pada Data 5.313 xml



Gambar 5.5.2 *Cluster* Kata Efisiensi pada Data 4.500 xml



Gambar 5.5.3 *Cluster* Kata Fraksi pada Data 5.313 xml



Gambar 5.5.4 *Cluster* Kata Fraksi pada Data 4.500 xml

5.6 Skenario Uji Coba 5

Skenario uji coba 5 adalah skenario pembangunan tesaurus menggunakan pasangan kata yang memiliki leksikal "Verba" dan "Verba". Skenario ini bertujuan untuk mengetahui *cluster* yang terbentuk dan kata keluaran yang dihasilkan oleh tesaurus jika dilakukan pencarian kata terhadap tesaurus.

Uji coba dilakukan pada jumlah data 5.313 xml dan 7.563 xml dengan topik berita acak menggunakan 5 kata pencarian pada masing-masing banyaknya data artikel berita. Catat setiap hasil keluaran daftar kata dan nilai kedekatan yang diberikan oleh tesaurus.

5.6.1 Tahapan Uji Coba

Tahapan-tahapan uji coba yang dilakukan untuk mendapatkan hasil keluaran yang dihasilkan oleh skenario uji coba 5 adalah sebagai berikut:

1. Lakukan proses *clustering* dengan menggunakan pasangan kata yang memiliki leksikal "Verba" dan "Verba" pada jumlah data 5.313 xml.
2. Setelah terbentuk satu *cluster* besar, bagi *cluster* tersebut menggunakan fungsi ***cut_tree()*** dengan nilai parameter *height* sesuai dengan yang telah dijelaskan pada subbab 5.1.3.
3. Setelah terbentuk beberapa *cluster*, lakukan penghitungan Kullback-Leiber *divergence* antar kata yang terdapat dalam satu *cluster* dan simpan hasil penghitungan tersebut pada basis data.
4. Kemudian, lakukan pencarian kata seperti yang dilakukan pada skenario uji coba 2 untuk mendapatkan kata keluaran dan nilai kedekatan kata.
5. Lakukan pencatatan terhadap hasil keluaran yang diberikan oleh tesaurus.
6. Kembali lakukan dari tahap 1 pada banyaknya jumlah data 7.563 xml.

5.6.2 Hasil Uji Coba

Tabel 5.6.1 adalah tabel hasil pencarian pada skenario uji coba 5 pada tesaurus yang dibangun pada proses *clustering* menggunakan pasangan kata yang memiliki leksikal "Verba" dan "Verba". Banyaknya artikel yang digunakan adalah 5.313 xml dan 7.563 xml dengan topik artikel acak. Kata-kata yang digunakan untuk melakukan pencarian adalah "meningkat", "memperbaiki", "terpuruk", "tempuh" dan "memacu". Tabel tersebut digunakan untuk membandingkan pengaruh leksikal pasangan kata dan jumlah data terhadap kata keluaran pada analisis dan evaluasi.

Tabel 5.6.1 Hasil Pencarian Pasangan Kata Leksikal "Verba"

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
5.313 xml	Meningkat	Terbilang	0,072169
	Memperbaiki	Mempelajari	-0,036898
		Mengoperasikan	-0,015994
		Berwisata	-0,010963
		Meneliti	-0,009002
		Wisata	-0,007716
		Mengembangkan	-0,003182
		Berbicara	0,030887
		Membangun	0,123955
	Terpuruk	Bangkit	0,184126
	Tempuh	Berjarak	0,133070
	Memacu	Menyeimbangkan	-0,050280
7.563 xml	Meningkat	Tumbuh	-0,022001
		Musiman	-0,005665
		Berkembang	-0,001903
		Tercatat	0,051151
	Memperbaiki	Menyiasati	-0,021043
		Beruntung	0,040688
		Mengatasi	0,082713
		Mengurangi	0,087446
	Terpuruk	Rangkaian	-0,009530
		Teratur	-0,006492
		Memasuki	0,031386
		Mengikuti	0,126943

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Bangkit	0,220597
		Meninggalkan	0,255853
	Tempuh	Lari	-0,007851
		Berjarak	0,049582
		Berlari	0,054085
	Memacu	Menyeimbangkan	-0,039044
		Menggenjot	-0,016969
		Menandai	-0,001676
		Membalap	0,016487
		Memunculkan	0,036047
		Balapan	0,366446

5.6.3 Analisis dan Evaluasi

Pada hasil skenario uji coba 5 didapatkan hasil daftar kata dan nilai kedekatan yang diberikan oleh tesaurus berdasarkan topik berita dan jumlah banyaknya artikel berita yang diproses. Beberapa hal yang dapat diambil dan dilakukan analisis dari hasil uji coba tersebut adalah sebagai berikut:

1. Tabel 5.6.1 merupakan daftar kata dan nilai kedekatan yang dihasilkan dari pencarian kata terhadap tesaurus yang dibangun menggunakan pasangan kata yang memiliki leksikal “Verba”. Secara konteks, kata-kata yang dihasilkan sudah cukup sesuai, misalnya dari kata masukan “memacu” menghasilkan kata keluaran yang sesuai dengan konteks, yaitu “menggenjot”, “membalap” dan “balapan”. Konteks dari kata-kata tersebut behubungangan dengan suatu kompetisi balap. Kesesuaian konteks tersebut dapat dibandingkan dengan tesaurus yang dibangun menggunakan pasangan kata yang memiliki leksikal “Nomina” pada halaman C. Lampiran Keterangan Penjelasan.
2. Jika dilakukan pencarian menggunakan kata-kata yang memiliki leksikal selain leksikal “Verba”, maka hasil keluaran yang diberikan adalah *NULL* yang berarti kata tersebut tidak ada pada tesaurus.

5.7 Skenario Uji Coba 6

Skenario uji coba 6 adalah skenario pembangunan tesaurus menggunakan pasangan kata yang memiliki leksikal "Nomina" dan "Verba". Skenario ini bertujuan untuk mengetahui *cluster* yang terbentuk dan kata keluaran yang dihasilkan oleh tesaurus jika dilakukan pencarian kata terhadap tesaurus.

Uji coba dilakukan pada jumlah data 5.313 xml dan 7.563 xml dengan topik berita acak menggunakan 5 kata pencarian pada masing-masing banyaknya data artikel berita. Catat setiap hasil keluaran daftar kata dan nilai kedekatan yang diberikan oleh tesaurus.

5.7.1 Tahapan Uji Coba

Tahapan-tahapan uji coba yang dilakukan untuk mendapatkan hasil keluaran yang dihasilkan oleh skenario uji coba 5 adalah sebagai berikut:

1. Lakukan proses *clustering* dengan menggunakan pasangan kata yang memiliki leksikal "Nomina" dan "Verba" pada jumlah data 5.313 xml.
2. Setelah terbentuk satu *cluster* besar, bagi *cluster* tersebut menggunakan fungsi ***cut_tree()*** dengan nilai parameter *height* sesuai dengan yang telah dijelaskan pada subbab 5.1.3.
3. Setelah terbentuk beberapa *cluster*, lakukan penghitungan Kullback-Leiber *divergence* antar kata yang terdapat dalam satu *cluster* dan simpan hasil penghitungan tersebut pada basis data.
4. Kemudian, lakukan pencarian kata seperti yang dilakukan pada skenario uji coba 2 untuk mendapatkan kata keluaran dan nilai kedekatan kata.
5. Lakukan pencatatan terhadap hasil keluaran yang diberikan oleh tesaurus.
6. Kembali lakukan dari tahap 1 pada banyaknya jumlah data 7.563 xml.

5.7.2 Hasil Uji Coba

Tabel 5.7.1 adalah tabel hasil pencarian pada skenario uji coba 6 pada tesaurus yang dibangun pada proses *clustering* menggunakan pasangan kata yang memiliki leksikal "Nomina" dan "Verba". Banyaknya artikel yang digunakan adalah 5.313 xml dan 7.563 xml dengan topik artikel acak. Kata-kata yang digunakan untuk melakukan pencarian adalah "diundang", "terbang", "tamu", "anggota" dan "renang". Tabel tersebut digunakan untuk membandingkan pengaruh leksikal pasangan kata dan jumlah data terhadap kata keluaran pada analisis dan evaluasi. Visualisasi *cluster* hasil proses pembangunan tesaurus dan gambaran *tree* yang terbentuk terlampir pada halaman lampiran buku tugas akhir.

Tabel 5.7.1 Hasil Pencarian Pasangan Kata Leksikal "Nomina" – "Verba"

Jumlah Artikel	Kata Masukan	Kata Keluaran	Leksikal Kata	Nilai Kedekatan
5.313 xml	Diundang (Nomina)	Menghadiri	Verba	-0,042181
		Rombongan	Nomina	0,010615
		Dibicarakan	Nomina	0,028297
		Ditemui	Verba	0,045274
		Wawancara	Nomina	0,047199
		Tamu	Nomina	0,050176
		Tahunan	Nomina	0,096358
		Penyelenggaraan	Nomina	0,132798
		Hadir	Verba	0,164189
	Terbang (Verba)	Rute	Nomina	0,022664
	Tamu (Nomina)	Menghadiri	Verba	-0,032290
		Diundang	Nomina	-0,020546
		Rombongan	Nomina	-0,005237
		Dibicarakan	Nomina	0,001979
		Penyelenggaraan	Nomina	0,036546
		Ditemui	Verba	0,077164
		Tahunan	Nomina	0,086609
		Wawancara	Nomina	0,122713
		Hadir	Verba	0,192784

Jumlah Artikel	Kata Masukan	Kata Keluaran	Leksikal Kata	Nilai Kedekatan
	Anggota (Nomina)	Membentuk	Verba	0,058359
	Renang (Verba)	Kontingen	Nomina	-0,046838
		Stadion	Nomina	-0,024330
		Menggeser	Verba	-0,014903
		Olahraga	Nomina	0,110561
		Medali	Nomina	0,053214
7.563 xml	Diundang (Nomina)	Menghadiri	Verba	-0,040496
		Dibicarakan	Nomina	0,037896
		Wawancara	Nomina	0,056384
		Tamu	Nomina	0,057190
		Ditemui	Verba	0,094677
		Penyelenggaraan	Nomina	0,142482
		Hadir	Nomina	0,169458
	Terbang (Verba)	Keselamatan	Nomina	-0,021613
		Mandala	Nomina	0,011390
		Mencabut	Verba	0,013899
		Sanksi	Nomina	0,015060
		Rute	Nomina	0,028468
		Penerbangan	Nomina	0,102864
	Tamu (Nomina)	Diundang	Nomina	-0,020060
		Menghadiri	Verba	-0,008253
		Dibicarakan	Nomina	0,002316
		Penyelenggaraan	Nomina	0,034432
		Ditemui	Verba	0,088071
		Wawancara	Nomina	0,134891
		Hadir	Nomina	0,213116
	Anggota (Nomina)	Didiet	Nomina	-0,043997
		Gebrakan	Nomina	-0,023604
		Markas	Nomina	-0,022228
		Kesediaan	Nomina	-0,016645
		Melancarkan	Verba	-0,007874
		Oposisi	Nomina	-0,000885
		Personel	Nomina	0,003611
		Bergabung	Verba	0,062776
		Membentuk	Verba	0,064740
	Renang (Verba)	Kontingen	Nomina	-0,047932
		Stadion	Nomina	-0,024464
		Medali	Nomina	0,053021

Jumlah Artikel	Kata Masukan	Kata Keluaran	Leksikal Kata	Nilai Kedekatan
		Olahraga	Nomina	0,116692

5.7.3 Analisis dan Evaluasi

Pada hasil skenario uji coba 6 didapatkan hasil daftar kata, leksikal kata keluaran dan nilai kedekatan yang diberikan oleh tesaurus berdasarkan leksikal dari pasangan kata yang digunakan dalam proses *clustering* dan berdasarkan jumlah banyaknya artikel berita yang diproses. Hal yang dapat diambil dan dilakukan analisis dari hasil uji coba tersebut adalah pengaruh leksikal kata *registered word* terhadap hasil keluaran dari proses pencarian.

Pada Tabel 5.7.1 menunjukkan bahwa rata-rata lebih dari 65% leksikal dari kata hasil keluaran proses pencarian memiliki leksikal "Nomina", terlepas dari leksikal kata yang digunakan sebagai masukan proses pencarian. Sedangkan pasangan kata yang digunakan untuk membangun tesaurus pada proses *clustering* skenario uji coba 6 adalah pasangan kata yang memiliki leksikal "Nomina" dan "Verba", sehingga yang bertindak sebagai leksikal dari *registered word* adalah "Nomina" dan leksikal dari kata *co-occurrence* adalah "Verba". Hal tersebut menunjukkan bahwa leksikal dari *registered word* yang digunakan dalam proses *clustering* memengaruhi leksikal dari kata keluaran proses pencarian yang secara langsung juga memengaruhi kata keluaran itu sendiri.

Selain pengaruh terhadap kata keluaran dan leksikal dari kata keluaran, pengaruh selanjutnya dari leksikal *registered word* adalah anggota kata yang terdapat dalam satu *cluster*. Melihat dari hasil leksikal kata keluaran yang sebagian besar adalah "Nomina", dapat tergambarkan keanggotaan dari kata pada masing-masing *cluster* (Gambar B.1 hingga Gambar B.4). Hal tersebut dikarenakan proses penghitungan nilai kedekatan dengan Kullback-Leiber *divergence* dilakukan pada tiap kata yang merupakan anggota di satu *cluster*.

5.8 Skenario Uji Coba 7

Skenario uji coba 7 adalah skenario pembangunan tesaurus menggunakan pasangan kata yang memiliki leksikal "Verba" dan "Nomina". Skenario ini bertujuan untuk mengetahui *cluster* yang terbentuk dan kata keluaran yang dihasilkan oleh tesaurus jika dilakukan pencarian kata terhadap tesaurus.

Uji coba dilakukan pada jumlah data 5.313 xml dan 7.563 xml dengan topik berita acak menggunakan 5 kata pencarian pada masing-masing banyaknya data artikel berita. Catat setiap hasil keluaran daftar kata dan nilai kedekatan yang diberikan oleh tesaurus.

5.8.1 Tahapan Uji Coba

Tahapan-tahapan uji coba yang dilakukan untuk mendapatkan hasil keluaran yang dihasilkan oleh skenario uji coba 5 adalah sebagai berikut:

1. Lakukan proses *clustering* dengan menggunakan pasangan kata yang memiliki leksikal "Verba" dan "Nomina" pada jumlah data 5.313 xml.
2. Setelah terbentuk satu *cluster* besar, bagi *cluster* tersebut menggunakan fungsi ***cut_tree()*** dengan nilai parameter *height* sesuai dengan yang telah dijelaskan pada subbab 5.1.3.
3. Setelah terbentuk beberapa *cluster*, lakukan penghitungan Kullback-Leiber *divergence* antar kata yang terdapat dalam satu *cluster* dan simpan hasil penghitungan tersebut pada basis data.
4. Kemudian, lakukan pencarian kata seperti yang dilakukan pada skenario uji coba 2 untuk mendapatkan kata keluaran dan nilai kedekatan kata.
5. Lakukan pencatatan terhadap hasil keluaran yang diberikan oleh tesaurus.
6. Kembali lakukan dari tahap 1 pada banyaknya jumlah data 7.563 xml.

5.8.2 Hasil Uji Coba

Tabel 5.8.1 adalah tabel hasil pencarian pada skenario uji coba 7 pada tesaurus yang dibangun pada proses *clustering* menggunakan pasangan kata yang memiliki leksikal "Verba" dan "Nomina". Banyaknya artikel yang digunakan adalah 5.313 xml dan 7.563 xml dengan topik artikel acak. Kata-kata yang digunakan untuk melakukan pencarian adalah "merambah", "memengaruhi", "norma", "menumbangkan" dan "mengatasi". Tabel tersebut digunakan untuk membandingkan pengaruh leksikal pasangan kata dan jumlah data terhadap kata keluaran pada analisis dan evaluasi. Visualisasi *cluster* hasil proses pembangunan tesaurus dan gambaran *tree* yang terbentuk terlampir pada halaman lampiran buku tugas akhir.

Tabel 5.8.1 Hasil Pencarian Pasangan Kata Leksikal "Verba" – "Nomina"

Jumlah Artikel	Kata Masukan	Kata Keluaran	Leksikal Kata	Nilai Kedekatan
5.313 xml	Merambah (Verba)	Menggeluti	Verba	-0,062131
		Melintang	Verba	-0,052815
		Berprestasi	Verba	0,039488
		Berkarya	Verba	0,068160
		Dunia	Nomina	0,772603
	Memengaruhi (Verba)	Menggerakkan	Verba	-0,020424
		Dorong	Verba	-0,004263
		Menyinggung	Verba	0,011601
		Perbaikan	Nomina	0,026065
		Pertumbuhan	Nomina	0,098604
		Penyelenggaraan	Nomina	0,125111
	Norma (Nomina)	Melanggar	Verba	-0,017361
		Mematuhi	Verba	-0,016219
		Mengacu	Verba	0,023156
		Ketentuan	Nomina	0,026223
		Mengedepankan	Verba	0,115781
	Menumbangkan (Verba)	Menutup	Verba	0,012322
		Menundukkan	Verba	0,181732
		Kemenangan	Nomina	0,920475
	Mengatasi	Kemacetan	Nomina	-0,011504

Jumlah Artikel	Kata Masukan	Kata Keluaran	Leksikal Kata	Nilai Kedekatan
	(Verba)			
7.563 xml	Merambah (Verba)	Melintang	Verba	-0,054732
		Berkecimpung	Verba	-0,041686
		Minatnya	Nomina	-0,020770
		Kader	Nomina	0,040013
		Memajukan	Verba	0,091716
		Sektor	Nomina	0,303871
		Dunia	Nomina	0,846455
	Memengaruhi (Verba)	Pertumbuhannya	Verba	-0,024225
		Merosotnya	Verba	-0,020650
		Dorong	Verba	-0,011835
		Menyinggung	Verba	0,006728
		Penyelenggaraan	Nomina	0,121383
		Pertumbuhan	Nomina	0,387111
	Norma (Nomina)	Melanggar	Verba	-0,026443
		Mematuhi	Verba	-0,006220
		Bertentangan	Verba	0,009741
		Ketentuan	Nomina	0,026721
		Tertuang	Verba	0,030491
		Mengacu	Verba	0,034567
	Menumbangkan (Verba)	Merebut	Verba	-0,004424
		Menutup	Verba	0,020972
		Beruntun	Verba	0,150070
		Menundukkan	Verba	0,181732
		Kemenangan	Nomina	0,922337
	Mengatasi (Verba)	Terendam	Verba	-0,055811
		Tren	Nomina	-0,014846
		Memicu	Verba	-0,010478
		Meningkatnya	Nomina	-0,006364
		Merasakan	Verba	-0,005746
		Mengindikasikan	Verba	-0,005578
		Dampak	Nomina	0,055986

5.8.3 Analisis dan Evaluasi

Pada hasil skenario uji coba 7 didapatkan hasil daftar kata, leksikal kata keluaran dan nilai kedekatan yang diberikan oleh tesaurus berdasarkan leksikal dari pasangan kata yang digunakan dalam proses *clustering* dan berdasarkan jumlah

banyaknya artikel berita yang diproses. Hal yang dapat diambil dan dilakukan analisis dari hasil uji coba tersebut adalah pengaruh leksikal kata *registered word* terhadap hasil keluaran dari proses pencarian.

Pada Tabel 5.8.1 menunjukkan bahwa rata-rata lebih dari 65% leksikal dari kata hasil keluaran proses pencarian memiliki leksikal "Verba", terlepas dari leksikal kata yang digunakan sebagai masukan proses pencarian. Sedangkan pasangan kata yang digunakan untuk membangun tesaurus pada proses *clustering* skenario uji coba 7 adalah pasangan kata yang memiliki leksikal "Verba" dan "Nomina", sehingga yang bertindak sebagai leksikal dari *registered word* adalah "Verba" dan leksikal dari kata *co-occurrence* adalah "Nomina". Hal tersebut menunjukkan bahwa leksikal dari *registered word* yang digunakan dalam proses *clustering* memengaruhi leksikal dari kata keluaran proses pencarian yang secara langsung juga memengaruhi kata keluaran itu sendiri.

Selain pengaruh terhadap kata keluaran dan leksikal dari kata keluaran, pengaruh selanjutnya dari leksikal *registered word* adalah anggota kata yang terdapat dalam satu *cluster*. Melihat dari hasil leksikal kata keluaran yang sebagian besar adalah "Verba", dapat tergambarkan keanggotaan dari kata pada masing-masing *cluster* (Gambar B.5 hingga Gambar B.8). Hal tersebut dikarenakan proses penghitungan nilai kedekatan dengan Kullback-Leiber *divergence* dilakukan pada tiap kata yang merupakan anggota di satu *cluster*.

(Halaman ini sengaja dikosongkan)

LAMPIRAN

A. Lampiran Tabel

Tabel A.1 Tabel Hasil Uji Coba Pencarian Kata

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
3.000 xml	Laras	Senapan	-0,078406
		Senjata	0,213795
	Senapan	Pistol	0,051054
		Peluru	0,058006
		Senjata	0,131028
	Prosedur	Dikoordinasikan	-0,068327
		Sop	-0,019475
		Apresiasi	-0,004394
		Pemenang	-0,001713
		Kandidat	0,000881
		Kritik	0,002981
		Substansi	0,004066
		Debat	0,004338
		Akibat	0,008174
		Mekanisme	0,016578
		Daftar	0,029204
		Peserta	0,035044
	Dampak	Pengendalian	-0,055709
		Pemantauan	-0,011210
		Pure	-0,009784
		Pendatang	-0,009617
		Model	-0,008866
		Kejutan	-0,006053
		Frekuensi	-0,005471
		Istrinya	-0,004965
		Ra	-0,003936
		Peringkat	-0,003495
		Istri	-0,002407
		Konser	0,007439
		Pemberitaan	0,016424
		Nilai	0,066008
	Susunan	Raden	0,030801
		Polemik	0,039611
		Direksi	0,045879
		Komisaris	0,149442

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Agenda	0,166845
		Detail	0,177389
	Publikasi	Risalah	-0,078597
		Pemberitaan	-0,025644
		Pemantauan	-0,008924
		Suasana	0,053678
		Iklan	0,067744
		Konser	0,075667
		Istrinya	0,092323
		Tangga	0,147550
		Nilai	0,250995
	Frekuensi	Pemantauan	0,016021
		Iklan	0,050531
		Dampak	0,095544
		Pemberitaan	0,134967
		Tangga	0,210164
		Nilai	0,431069
	Mahasiswa	Swadaya	-0,041555
		Jemaah	-0,032755
		Mahasiswi	-0,028353
		Diwarnai	-0,027846
		Elemen	-0,025030
		Demonstrasi	-0,023776
		Konsistensi	-0,021929
		Amanat	-0,020293
		Demo	-0,019546
		Bicarakan	-0,019316
		Kepercayaan	-0,018490
		Haji	-0,018386
		Kongres	-0,018248
		Dinamika	-0,017971
		Ibunya	-0,016078
		Darurat	-0,015785
		Pentas	-0,015301
		Petualangan	-0,015196
		Sambutan	-0,014957
		Pembinaan	-0,014607
		Skala	-0,014119
		Pelantikan	-0,014040
		Penguasa	-0,013509

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Disaksikan	-0,012773
		Debut	-0,011734
		Drama	-0,010941
		Rancangan	-0,010451
		Pengobatan	-0,010388
		Cip	-0,010091
		Kurikulum	-0,009644
		Raung	-0,009569
		Ribuan	-0,009499
		Jaringan	-0,009264
		Jantung	-0,009177
		Produser	-0,009121
		Diterapkan	-0,008888
		Wulan	-0,008799
		Ulama	-0,008505
		Vampir	-0,007741
		Kewenangan	-0,007719
		Pembahasan	-0,007585
		Pembukaan	-0,007042
		Peraturan	-0,006988
		Pengawasan	-0,006767
		Komunikasi	-0,006606
		Transfer	-0,006565
		Darah	-0,005570
		Ilmu	-0,005553
		Bumbu	-0,005487
		Ketentuan	-0,005441
		Dokumen	-0,004857
		Ketenagakerjaan	-0,004846
		Instansi	-0,004492
		Sifat	-0,004220
		Siswa	-0,003932
		Gunung	-0,003814
		Hubungan	-0,003806
		Pengganti	-0,003523
		Titik	-0,003404
		Penolakan	-0,003293
		Revisi	-0,003194
		Kartu	-0,002786
		Tema	-0,002553

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Dewi	-0,002143
		Pertunjukan	-0,001927
		Massa	-0,001021
		Jaminan	0,000046
		Konsesi	0,000640
		Pasal	0,000807
		Undang	0,001836
		Manusia	0,002461
		Pelayanan	0,003090
		Kisah	0,004774
		Pelatihan	0,005493
		Korban	0,006310
		Bundaran	0,007104
		Kepolisian	0,007569
		Harapan	0,008619
		Kualitas	0,009704
		Panggung	0,010777
		Artis	0,016618
		Petisi	0,019324
		Lingkungan	0,029884
		Aksi	0,039570
		Kuliah	0,054854
		Pekerja	0,055321
		Pengamat	0,065241
		Manajemen	0,077244
		Sekolah	0,151557
	Terkini	Sorotan	-0,014094
		Polemik	-0,010640
		Kecurigaan	-0,009030
		Intervensi	-0,007916
		Detail	0,003203
		Raden	0,004624
		Uangnya	0,010324
		Agenda	0,034308
		Situasi	0,106681
	Pemantauan	Pemberitaan	-0,022293
		Frekuensi	-0,009917
		Peringkat	0,000835
		Pengendalian	0,004679
		Publikasi	0,011898

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Suasana	0,013065
		Suami	0,029865
		Iklan	0,064090
		Istri	0,072434
		Kejutan	0,115014
		Konser	0,248298
		Dampak	0,265076
		Nilai	0,641635
7.563 xml	Laras	Senapan	-0,026506
		Pistol	0,005914
		Pengendara	0,025596
		Ojek	0,079520
		Senjata	0,221528
	Senapan	Peluru	0,050171
		Laras	0,079520
		Perampok	0,090678
		Pistol	0,194526
		Senjata	0,298345
	Prosedur	Dikoordinasikan	-0,048697
		Didesain	-0,037987
		Sop	-0,016400
		Syaratnya	-0,007647
		Substansi	-0,000295
	Dampak	Sentimen	-0,055948
		Dorongan	-0,017018
		Mimpi	-0,014880
		Antisipasi	0,033083
		Sinyal	0,102410
	Susunan	Direksi	0,043355
	Publikasi	Risalah	-0,043876
		Pemberitaan	-0,021344
		Diseleksi	-0,007977
		Pemanggilan	-0,006275
		Klarifikasi	-0,006091
		Pemantauan	0,003011
		Komite	0,019926
		Peserta	0,178892
	Frekuensi	Penggelaran	-0,078287
		Spektrum	-0,016967
		Uji	-0,015005

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Kepatutan	-0,014930
		Perpindahan	-0,012950
		Pita	-0,012548
	Mahasiswa	Katolik	-0,068377
		Dosen	-0,042298
		Bakatnya	-0,033454
		Diploma	-0,024691
		Diundang	-0,024264
		Elemen	-0,022033
		Matematika	-0,021836
		Sarjana	-0,018737
		Sultan	-0,017007
		Wulan	-0,016859
		Fakultas	-0,016644
		Teks	-0,014776
		Demo	-0,014407
		Pengajar	-0,014127
		Kristen	-0,013776
		Keluarganya	-0,013495
		Siswa	-0,013185
		Lulusan	-0,012831
		Latar	-0,012472
		Sahabat	-0,012277
		Pakar	-0,012053
		Banding	-0,011518
		Tamu	-0,011518
		Ulama	-0,011471
		Minggunya	-0,011313
		Niaga	-0,011244
		Sains	-0,010523
		Dipukuli	-0,009951
		Keguruan	-0,009951
		Pembinaan	-0,009290
		Kepribadian	-0,009102
		Praperadilan	-0,008703
		Diputihkan	-0,008691
		Kepolisian	-0,008336
		Kejaksaan	-0,007966
		Curiga	-0,007542
		Sensasi	-0,007422

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Terdakwa	-0,007225
		Persidangan	-0,007088
		Penyelundupan	-0,007087
		Rumpun	-0,007026
		Penyelidikan	-0,006939
		Rudi	-0,006879
		Jero	-0,005910
		Pelatihan	-0,005719
		Magister	-0,005594
		Wawasan	-0,005290
		Permohonan	-0,005204
		Penyidik	-0,004889
		Ilmu	-0,004612
		Nostalgia	-0,004601
		Siswi	-0,004315
		Penyuluh	-0,004253
		Status	-0,004249
		Tari	-0,004102
		Dewi	-0,003277
		Saksi	-0,003230
		Keterangan	-0,003124
		Berkas	-0,002573
		Rumahnya	-0,002058
		Putusan	-0,001791
		Upah	-0,001420
		Sidang	-0,001062
		Perdamaian	0,000467
		Penjelasan	0,000585
		Bukti	0,000734
		Profesor	0,001028
		Kampus	0,001570
		Hakim	0,002507
		Gugatan	0,004652
		Pengadilan	0,005980
		Komunikasi	0,007384
		Kuliah	0,009869
		Pemeriksaan	0,012832
		Jurusan	0,020691
		Polisi	0,036030
		Pesan	0,050357

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Juri	0,059509
		Pengamat	0,061841
		Guru	0,068859
	Terkini	Struktur	-0,012012
		Kepengurusan	-0,004003
		Situasi	0,047194
		Perkembangan	0,094652
	Pemantauan	Pemberitaan	-0,021722
		Pemanggilan	-0,012423
		Klarifikasi	-0,010483
		Publikasi	-0,002685
		Komite	0,003660
		Pengendalian	0,009527
		Risalah	0,082861
		Peserta	0,351525
8.813 xml	Laras	Senapan	-0,016723
		Pistol	0,005914
		Pengendara	0,025596
		Kebudayaan	0,039098
		Bekasi	0,054598
		Dikabarkan	0,060287
		Ojek	0,080848
		Pelaku	0,148110
		Senjata	0,223137
		Lintasan	0,238112
	Senapan	Golok	-0,019565
		Peluru	0,013882
		Laras	0,033447
		Perampok	0,056461
		Pistol	0,090552
		Senjata	0,161052
		Pelaku	0,198282
	Prosedur	Dikoordinasikan	-0,048425
		Didesain	-0,037666
		Sop	-0,015016
		Antrean	-0,011844
		Pemiliknya	-0,009119
		Syaratnya	-0,004281
		Rupa	-0,002093
		Substansi	-0,000580

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Kendaraan	0,087803
	Dampak	Sentimen	-0,056024
		Dorongan	-0,015041
		Sinyal	0,100819
	Susunan	Direksi	0,037141
	Publikasi	Risalah	-0,042967
		Pemberitaan	-0,021568
		Jaringannya	-0,008853
		Diseleksi	-0,008123
		Pemanggilan	-0,006853
		Klarifikasi	-0,003348
		Cakupan	0,002841
		Pemantauan	0,002841
		Komite	0,019044
		Kebakaran	0,054225
		Pemenang	0,101937
		Peserta	0,174044
	Frekuensi	Penggelaran	-0,077763
		Spektrum	-0,017200
		Uji	-0,014877
		Kepatutan	-0,014803
		Perpindahan	-0,012150
		Pita	-0,011710
		Prototipe	-0,007142
		Serangkaian	-0,005136
		Tes	-0,004847
		Pengujian	-0,004382
		Verifikasi	0,000483
	Mahasiswa	Katolik	-0,059751
		Sastra	-0,053144
		Dosen	-0,041390
		Psikologi	-0,031946
		Mahasiswi	-0,031650
		Diploma	-0,023917
		Matematika	-0,023006
		Dimas	-0,020250
		Sarjana	-0,019018
		Lulusan	-0,017403
		Wisatawan	-0,016363
		Pendeta	-0,016278

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Rahman	-0,016172
		Kristen	-0,015965
		Ratna	-0,015937
		Pakar	-0,015874
		Fakultas	-0,015723
		Wulan	-0,015619
		Bunda	-0,014662
		Maulana	-0,014516
		Pengajar	-0,014306
		Perkebunan	-0,012579
		Panji	-0,012324
		Indra	-0,012155
		Niaga	-0,011296
		Suaminya	-0,011057
		Minggunya	-0,011049
		Kebersihan	-0,011008
		Pengamanan	-0,010761
		Rama	-0,010558
		Kuarter	-0,009875
		Pertemanan	-0,009591
		Keguruan	-0,009591
		Cita	-0,009508
		Pembinaan	-0,009428
		Gitaris	-0,009239
		Damian	-0,008775
		Kepribadian	-0,008600
		Mahasiswanya	-0,008387
		Personel	-0,008243
		Niat	-0,008024
		Puji	-0,007989
		Dewi	-0,007951
		Syukur	-0,007732
		Sertifikat	-0,007619
		Piano	-0,007238
		Rumpun	-0,006795
		Kedudukan	-0,006705
		Pentolan	-0,006545
		Sena	-0,006479
		Pelatihan	-0,006117
		Kelapa	-0,006020

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Drum	-0,005827
		Magister	-0,005426
		Diah	-0,005275
		Wawasan	-0,005007
		Pedangdut	-0,005007
		Bas	-0,004984
		Perselisihan	-0,004669
		Sains	-0,004571
		Perceraian	-0,004476
		Gereja	-0,004374
		Penyuluh	-0,004144
		Kampus	-0,004021
		Pisang	-0,003931
		Ilmu	-0,003902
		Pembelajaran	-0,003770
		Batasan	-0,003366
		Pohon	-0,002662
		Gitar	-0,001489
		Profesi	-0,001319
		Batas	0,000568
		Profesor	0,005255
		Kuliah	0,014587
		Jurusan	0,027002
		Guru	0,078159
	Terkini	Struktur	-0,012124
		Kepengurusan	-0,004041
		Situasi	0,046645
		Perkembangan	0,093159
	Pemantauan	Pemberitaan	-0,021840
		Modernisasi	-0,015257
		Pemanggilan	-0,013011
		Klarifikasi	-0,008377
		Cakupan	-0,007921
		Publikasi	-0,002542
		Komite	0,002648
		Pengendalian	0,008948
		Pemenang	0,008974
		Penguasa	0,040421
		Risalah	0,078852
		Kebakaran	0,153770

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Peserta	0,343564
10.813 xml	Laras	Senapan	-0,006980
		Pistol	0,006272
		Pembelajaran	0,007169
		Layang	0,011915
		Teras	0,012619
		Jembatan	0,016142
		Pengendara	0,025111
		Dikabarkan	0,057876
		Ojek	0,083277
		Lintasan	0,222719
		Senjata	0,264115
	Senapan	Sepucuk	-0,060364
		Golok	-0,026284
		Mekanik	-0,019361
		Rakitan	-0,007481
		Peluru	0,006475
		Laras	0,009473
		Dipesan	0,021502
		Perampok	0,022590
		Pistol	0,053959
		Balap	0,058404
		Promosi	0,068405
		Lintasan	0,076147
		Dikabarkan	0,088019
		Senjata	0,423242
	Prosedur	Penegakan	-0,047990
		Kaidah	-0,042615
		Biro	-0,028186
		Ketaatan	-0,019262
		Legalitas	-0,017573
		Penundaan	-0,016235
		Kemunduran	-0,015311
		Reputasi	-0,011406
		Kepatuhan	-0,011071
		Komplain	-0,010995
		Empati	-0,010874
		Persamaan	-0,010816
		Kredibilitas	-0,010686
		Asas	-0,009761

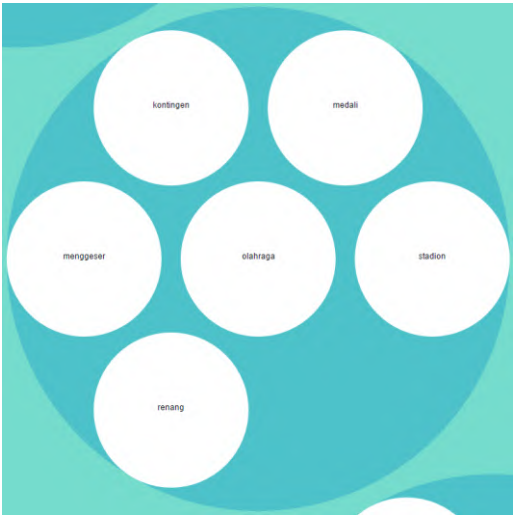
Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
		Bros	-0,008304
		Kompromi	-0,008063
		Pembatalan	-0,007831
		Sukarelawan	-0,006874
		Kalo	-0,005567
		Pencari	-0,005192
		Masyarakatnya	-0,005076
		Bon	-0,004971
		Moral	-0,003900
		Integritas	-0,001865
		Konsernya	-0,000893
		Kesejahteraan	-0,000753
		Kesamaan	0,000053
		Perlindungan	0,001301
		Aspek	0,007516
		Suasana	0,013897
		Emosi	0,016755
		Bantuan	0,023216
		Fisik	0,037580
	Dampak	Sentimen	-0,057422
		Sinergi	-0,012018
		Dorongan	-0,010398
		Antisipasi	0,041553
		Sinyal	0,102262
	Susunan	Perombakan	-0,040323
		Jajaran	0,171831
		Dewan	0,617885
	Publikasi	Risalah	-0,032012
		Pemantauan	-0,015065
		Pemberitaan	0,041919
	Frekuensi	Penggelaran	-0,076986
		Spektrum	-0,017521
		Perpindahan	-0,015364
		Uji	-0,012375
		Kepatutan	-0,009007
		Pita	-0,008873
		Kelayakan	0,010320
	Mahasiswa	Kampus	-0,029634
		Kuliah	-0,014490
		Auditorium	-0,009279

Jumlah Artikel	Kata Masukan	Kata Keluaran	Nilai Kedekatan
	Terkini	Situasi	0,073826
		Perkembangan	0,125911
	Pemantauan	Pengendalian	-0,011016
		Pemberitaan	0,000017
		Publikasi	0,034041
		Risalah	0,053087

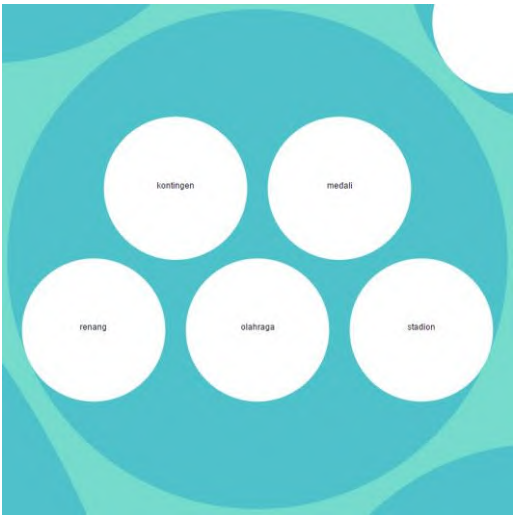
B. Lampiran Gambar

Gambar B.1 *Cluster* Kata Tamu pada Nomina–Verba Data 5.313 xml

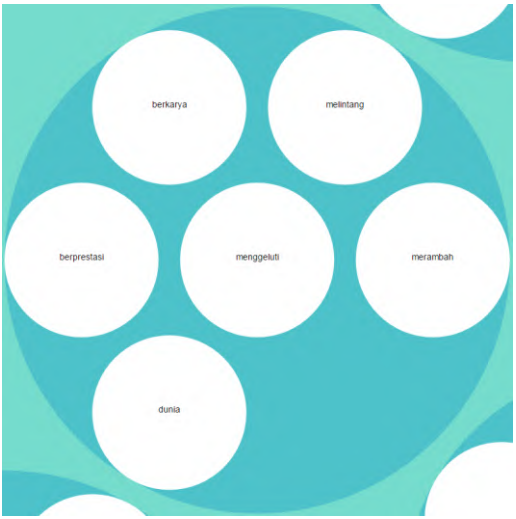
Gambar B.2 *Cluster* Kata Tamu pada Nomina–Verba Data 7.563 xml



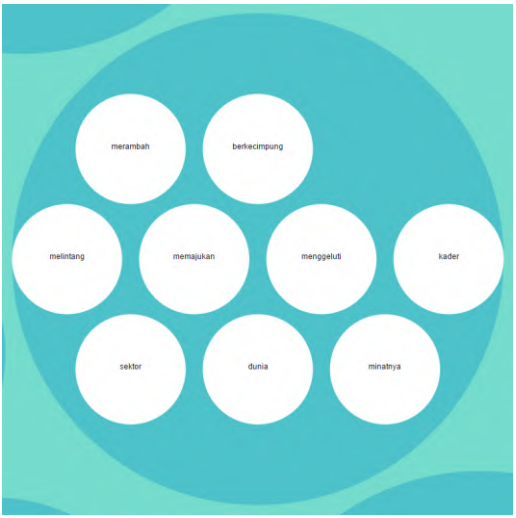
Gambar B.3 *Cluster* Kata Renang pada Nomina–Verba Data 5.313 xml



Gambar B.4 *Cluster* Kata Renang pada Nomina–Verba Data 7.563 xml



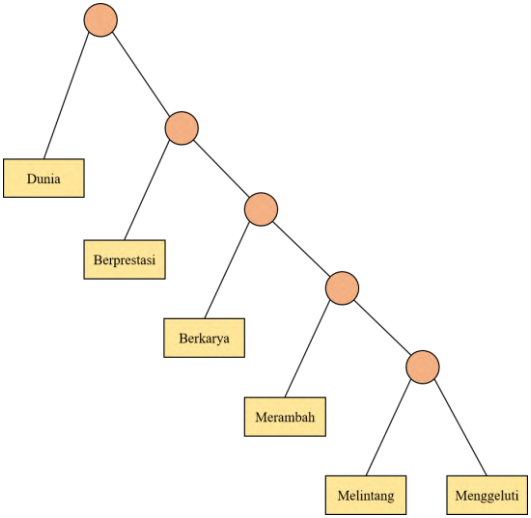
Gambar B.5 *Cluster* Kata Merambah pada Verba–Nomina Data 5.313 xml



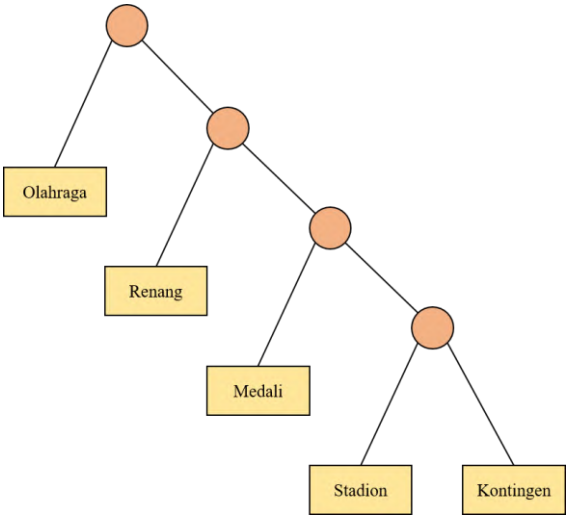
Gambar B.6 *Cluster* Kata Merambah pada Verba–Nomina Data 7.563 xml

Gambar B.7 *Cluster* Kata Memengaruhi pada Verba–Nomina Data 5.313 xml

Gambar B.8 *Cluster* Kata Memengaruhi pada Verba–Nomina Data 7.563 xml



Gambar B.9 *Tree* Kata Merambah pada Verba–Nomina Data 5.313 xml



Gambar B.10 *Tree* Kata Renang pada Nomina–Verba Data 7.563 xml

C. Lampiran Keterangan Penjelasan

Contoh dari Judul berita yang terdapat pasangan kata berleksikal "Verba" adalah sebagai berikut:

- Tempuh Jarak 230 Kilometer untuk Kompetisi.
  Pada berita tersebut terdapat pasangan kata yang memiliki leksikal "Verba" yaitu kata "tempuh" dan "berjarak". Hal tersebut menunjukkan bahwa pasangan kata dari hasil pencarian yang dilakukan sesuai dengan konteks yang ada pada berita tersebut, yaitu behubungan dengan jarak tempuh dari suatu tempat ke tempat lain.
- Hayden Hanya Jadi Penonton pada Balapan Kandang.
  Pada berita tersebut terdapat pasangan kata yang memiliki leksikal "Verba" yaitu kata "balapan" dan "memacu". Hal tersebut menunjukkan bahwa pasangan kata dari hasil pencarian yang dilakukan sesuai dengan konteks yang ada pada berita tersebut, yaitu behubungan dengan suatu kegiatan balapan untuk memacu motor.

Contoh dari Judul berita yang terdapat pasangan kata berleksikal "Nomina" adalah sebagai berikut:

- Ganda Putra Pastikan Raih Medali Emas.
  Pada berita tersebut terdapat pasangan kata yang memiliki leksikal "Nominta" yaitu kata "emas" dan "podium". Hal tersebut menunjukkan bahwa pasangan kata dari hasil pencarian yang dilakukan sesuai dengan konteks yang ada pada berita tersebut, yaitu behubungan dengan emas yang diraih sebagai juara yang diserahkan di atas podium tempuh dari suatu tempat ke tempat lain.
- Seorang PNS Terlibat Komplotan Perampok Rumah Mewah.
  Pada berita tersebut terdapat pasangan kata yang memiliki leksikal "Nomina" yaitu kata "senjata" dan "perampok". Hal tersebut menunjukkan bahwa pasangan kata dari hasil pencarian yang dilakukan sesuai dengan konteks yang ada

pada berita tersebut, yaitu perampok yang melakukan kejahatan dengan menggunakan senjata.

BAB VI
KESIMPULAN DAN SARAN

Bab ini membahas kesimpulan dari hasil uji coba terhadap tesaurus yang dibuat. Berisi jawaban dari rumusan masalah yang diangkat pada Bab I dan saran untuk pengembangan tesaurus pada tugas akhir ini untuk kedepannya.

6.1 Kesimpulan

Kesimpulan yang dapat diberikan setelah proses pengerjaan dari perancangan, implementasi dan uji coba yang telah dilakukan terhadap tesaurus yang dibangun adalah sebagai berikut:

1. Informasi *co-occurrence* dibentuk dengan membuat pasangan kata antara suatu kata dan kata selanjutnya. Informasi *co-occurrence* yang terbentuk merupakan kata hasil pemrosesan tokenisasi, penghapusan *stopword* dan *stemming*.
2. Tesaurus dibangun menggunakan dua proses utama yaitu praproses data dan pembentukan *cluster* kata. Praproses data meliputi tokenisasi, penghapusan *stopword*, *stemming* dan pembentukan informasi *co-occurrence*. Sedangkan pembentukan *cluster* kata meliputi proses pembentukan vektor kata, penghitungan *cosine similarity*, *hierarchical clustering* dan penghitungan Kullback-Leiber *divergence*.
3. Evaluasi yang dilakukan pada tesaurus yang dibangun atau tesaurus dilakukan dengan melakukan empat skenario pengujian yaitu penghitungan waktu eksekusi yang dibutuhkan untuk membangun tesaurus, pencarian kata dan penggunaan satu topik berita dan pembandingan antara satu topik berita dan topik acak.
4. Semakin banyak jumlah artikel yang diproses untuk membangun tesaurus, waktu yang dibutuhkan juga semakin lama.

5. Proses yang membutuhkan waktu paling lama adalah proses ekstraksi kata dari berkas xml artikel berita. Pada data artikel berita sebanyak 5.313 xml, dibutuhkan waktu selama 1.735 menit. Sedangkan proses yang membutuhkan waktu paling singkat adalah proses *clustering* dengan menggunakan pustaka bahasa pemrograman Python, SciPy, yang pada data artikel berita sebanyak 5.313 xml hanya membutuhkan waktu 10 menit.
6. Banyaknya berkas xml artikel berita dan topik yang digunakan untuk membangun tesaurus berpengaruh terhadap hasil keluaran yang diberikan, dalam bentuk perbedaan daftar kata keluaran, perbedaan nilai kedekatan ataupun perbedaan kata-kata anggota *cluster* yang terbentuk. Pada data hasil uji coba pencarian menggunakan kata "senapan" dan "terkini" pada data artikel berita sebanyak 5.313 xml dan 7.563 xml berdasarkan skenario uji coba 2 menunjukkan bentuk-bentuk perbedaan yang dimaksud.
7. Leksikal kata dari pasangan kata yang digunakan untuk membangun tesaurus memengaruhi hasil keluaran dari tesaurus ketika dilakukan pencarian kata.
8. Pengaruh tersebut berupa rata-rata 65% dari kata keluaran akan memiliki leksikal yang sama dengan leksikal dari *registered word* yang digunakan untuk membangun tesaurus. Sebagai contoh, pasangan leksikal kata yang digunakan adalah "Nomina" dan "Verba", leksikal dari *registered word* adalah "Nomina", maka kemungkinan kata keluaran memiliki leksikal "Nomina" adalah 65%.

6.2 Saran

Saran yang dapat diberikan untuk pengembangan tesaurus pada tugas akhir ini untuk kedepannya adalah sebagai berikut:

1. Penggunaan bahasa pemrograman lain, misalnya Python untuk melakukan proses ektraksi kata, pembentukan

informasi *co-occurrence* dan penghitungan *cosine similarity*, karena dibutuhkan waktu yang cukup lama jika dilakukan menggunakan bahasa pemrograman PHP.
2. Gunakan data yang lebih banyak agar tesaurus dapat memberikan hasil keluaran yang lebih baik.
3. Perbaiki tampilan hasil *clustering* berupa bentuk *dendogram* atau representasi dari hirarki antar kata yang digunakan untuk membangun tesaurus.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

[1] P. Wang, J. Hu, H.-J. Zeng and Z. Chen, "Using Wikipedia knowledge to improve text classification," *Knowledge and Information Systems*, vol. 19, no. 3, pp. 265-281, 2008.

[2] F. J. Pinto, A. F. Martinez and C. F. Perez-Sanjulian, "Joining automatic query expansion based on thesaurus and word sense disambiguation using WordNet," *International Journal of Computer Applications in Technology*, vol. 33, no. 4, pp. 271-279, 2008.

[3] Wikipedia, "Thesaurus - Wikipedia, the free encyclopedia," [Online]. Available: https://en.wikipedia.org/wiki/Thesaurus. [Accessed 5 Desember 2015].

[4] C. Fellbaum, WordNet: An Electronic Lexical Database, Cambridge, MA: MIT Press, 1998.

[5] A. S. Febriana, "Ivan Lanin: Indonesian Language Evangelist," 25 January 2010. [Online]. Available: www.thejakartapost/com/news/2010/01/25/ivan-lanin-indonesian-language-evangelist.html. [Accessed 14 December 2015].

[6] V. J. Hodge and J. Austin, "Hierarchical word clustering—Automatic thesaurus generation," *Neurocomputing*, vol. 48, no. 1-4, pp. 819-846, 2002.

[7] H. Chen, B. R. Schatz, T. Yim and D. Fye, "Automatic Thesaurus Generation for an Electronic Community," *Journal of the American Society for Information Science*, vol. 46, no. 3, pp. 175-193, 1995.

[8] H. Schutze and J. O. Pedersen, "A Cooccurrence based

Thesaurus and Two Applications to Information Retrieval," *International Journal of Information Processing and Management,* vol. 33, no. 3, pp. 307-318, 1997.

[9] K. Morita, E.-S. Atlam, M. Fuketa, K. Tsuda, M. Oono and J. Aoe, "Word classification and hierarchy using co-occurrence word information," *Information Processing & Management,* vol. 40, no. 6, pp. 957-972, 2004.

[10] Y. Matsuo and M. Ishizuka, "Keyword Extraction from a Single Document using Word Co-occurrence Statistical Information," *International Journal on Artificial Intelligence Tools,* vol. 13, no. 01, 2003.

[11] K. Bessho, T. Uchiyama and R. Kataoka, "Extraction of Hierarchical Relations among Words Based on Cooccurrences between Words and Semantic Attributes," *IEIC Technical Report (Institute of Electronics, Information and Communication Engineers),* vol. 106, no. 518, pp. 31-36, 2007.

[12] C. D. Manning, P. Raghavan and H. Schutze, Introduction to Information Retrieval, Cambridge University Press, 2008.

[13] A. Singhal, "Modern Information Retrieval: A Brief Overview," *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering,* vol. 24, no. 4, p. 35–43, 2001.

[14] Kateglo.com, "README.txt - Kateglo," [Online]. Available: http://kateglo.com/?mod=doc&doc=README.txt. [Accessed 8 June 2016].

[15] NumPy, "NumPy - Array Objects," [Online]. Available: http://docs.scipy.org/doc/numpy-1.10.1/reference/arrays.html. [Accessed 8 June 2016].

[16] SciPy, "Scientific Computing Tools for Python — SciPy.org," [Online]. Available: https://www.scipy.org/about.html. [Accessed 8 June 2016].

[17] M. Bostock, "Data-Driven Documents," [Online]. Available: https://d3js.org. [Accessed 12 May 2016].

[18] F. Z. Tala, "A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia.," Institute for Logic, Language and Computation, Universiteit van Amsterdam, 2003.

[19] N. Popov, "How big are PHP arrays (and values) really?," 12 December 2011. [Online]. Available: https://nikic.github.io/2011/12/12/How-big-are-PHP-arrays-really-Hint-BIG.html. [Accessed 2 May 2016].

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Miftahuddin Al Anshory, Lahir pada tanggal 14 Maret 1994. Anak pertama dari dua bersaudara. Saat ini sedang menempuh pendidikan perguruan tinggi negeri di Institut Teknologi Sepuluh Nopember Surabaya di jurusan Teknik Informatika, Fakultas Teknologi Informasi, angkatan tahun 2012.

Pernah mengikuti beberapa organisasi dan beberapa kepanitian diantaranya adalah Staf Media Informasi Himpunan Mahasiswa Teknik Computer-Informatika 2013-2014, Staf Komunikasi dan Informasi BEM ITS 2013-2014, Badan Pengurus Harian SCHEMATICS 2013 dan SCHEMATICS 2014.

miftah.anshory12@gmail.com