JURUSAN TEKNIK INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

INSTITUT TEKNOLOGI SEPULUH NOPEMBER

USULAN TUGAS AKHIR

1. IDENTITAS PENGUSUL

NAMA : **Miftahuddin Al Anshory**
NRP : **5112100026**
DOSEN WALI : **Diana Purwitasari, S.Kom., M.Sc.**
DOSEN PEMBIMBING : **1. Diana Purwitasari, S.Kom., M.Sc.**
2. Abdul Munif, S.Kom., M.Sc.

2. JUDUL TUGAS AKHIR

"Implementasi Tesaurus Hirarkis Secara Dinamis pada Data Berita Berbahasa Indonesia Menggunakan Informasi *Co-occurrence*."

3. LATAR BELAKANG

Tesaurus, sebuah pedoman pengetahuan yang penting, 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 sendiri merupakan kamus yang berisi daftar kata-kata yang dikelompokkan berdasarkan kemiripan arti atau makna (sinonim) [3].

Tesaurus seperti WordNet [4] pada umumnya dibuat secara manual, sehingga akurasi kemiripan arti kata diperoleh dengan nilai akurasi yang tinggi. Namun, sulit untuk memperbarui tesaurus tersebut ke dalam skala yang lebih besar, memerlukan waktu yang besar karena proses dilakukan secara manual. Sedangkan untuk tesaurus berbahasa Indonesia, seperti kamus bahasa Kateglo [5] dan situs web Sinonim Kata, hanya menampilkan keluaran berupa daftar kata, tanpa menampilkan nilai kemiripan kata-kata tersebut, sehingga tidak dapat diketahui kemiripan antara kata masukan dan kata keluaran.

Di sisi lain, beberapa riset telah dilakukan untuk membangun tesaurus seacara otomatis. Misalnya, metode pembangkitan menggunakan algoritma *neural network* [6], teknik pengelompokan menggunakan relasi *co-occurrence* [7] [8] [9], dan beberapa riset yang lain. Namun, terdapat kesulitan, yaitu untuk membagi kata baru ke dalam *node* klasifikasi yang baru. Selain itu, karena konstruksi tesaurus yang statis, perubahan seperti penambahan kata baru tersebut membutuhkan biaya yang besar.

Informasi *co-occurrence* adalah sebuah informasi nilai frekuensi kejadian munculnya dua kata pada waktu bersamaan pada dokumen [10], yang dimaksud bersamaan adalah muncul pada posisi bersebelahan. Frekuensi didapatkan dengan cara menghitung berapa kali dua kata tersebut muncul dalam satu dokumen. Jika terdapat di banyak dokumen, maka frekuensi akan diakumulasikan.

Hirarki kata yang terbentuk pada tesaurus digunakan untuk menentukan kata-kata yang memiliki posisi lebih tinggi, memiliki arti lebih luas atau umum (*broad term*) dan kata-kata bersifat khusus di bawahnya (*narrow term*). Sehingga, kata-kata tersebut nantinya akan dibentuk menjadi struktur pohon (*tree*). Semakin tinggi posisi suatu kata, maka arti kata tersebut semakin luas. Selain digunakan untuk menentukan posisi kata, hirarki juga digunakan untuk menghitung nilai kemiripan, Kullback-Leiber *divergence* [11], salah satu metode untuk menghitung kemiripan kata menggunakan hirarki dari kata tersebut.

Konstruksi tersaurus secara dinamis dimaksudkan ketika terjadi perubahan atau penambahan kata baru pada tesaurus, hal tersebut dapat dilakukan secara otomatis. Terdapat sebuah algoritma yang digukakan untuk melakukan perubahan pada struktur susunan *tree* pada tesaurus, yang memungkinkan terjadi perubahan hirarki kata pada susunan *tree* tersebut. Sehingga hal tersebut membuat pembaruan tesaurus tidak lagi dilakukan secara manual.

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

4. RUMUSAN MASALAH

Berikut hal-hal yang menjadi rumusan masalah pada Tugas Akhir ini, yaitu:

1. Bagaimana cara untuk membentuk informasi *co-occurrence* dari hasil ekstraksi kata artikel berita berbahasa Indonesia?

2. Bagaimana cara untuk membangun tesaurus yang hirarkis dan dinamis memanfaatkan nilai dari informasi *co-occurrence* kata?
3. Bagaimana cara untuk melakukan evaluasi pada tesaurus yang dibangun?

5. BATASAN MASALAH

Batasan masalah pada Tugas Akhir ini adalah sebagai berikut:

1. Domain data untuk kata hanya berasal dari ekstraksi artikel berita.
2. Artikel berita diambil dari situs web Kompas, sehingga tersaurus yang nantinya dibuat menggunakan bahasa Indonesia.
3. Keluaran dari tesaurus hanya berupa daftar kata dan nilai kemiripannya, tanpa definisi dari kata tersebut.

6. TUJUAN PEMBUATAN TUGAS AKHIR

Tujuan pengerjaan Tugas Akhir ini adalah untuk membangun tesaurus yang memiliki hirarki antar kata. Hirarki kata digunakan untuk mempertimbangkan nilai kemiripan antar kata, dan menentukan kata yang memiliki posisi lebih tinggi (*broader term*) dan kata yang lebih rendah. Tujuan selanjutnya adalah untuk membuat tesaurus yang dapat beradaptasi dengan perubahan kata secara dinamis, melakukan pembaruan pada struktur *tree* tesaurus secara otomatis. Pembuatan tesaurus tersebut memanfaatkan informasi *co-occurrence* dari kata-kata pada artikel berita.

7. MANFAAT TUGAS AKHIR

Manfaat dari Tugas Akhir ini adalah tesaurus ini diharapkan dapat digunakan sebagai rujukan untuk mencari nilai kemiripan dari suatu kata, ataupun digunakan mencari kata-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.

8. TINJAUAN PUSTAKA

1. **Algoritma Pemrosesan Teks**
   Algoritma pemrosesan teks digunakan untuk melakukan proses ekstraksi kata pada kumpulan dokumen artikel berita dan proses pembentukan informasi *co-occurrence* informasi dari pasangan kata hasil ekstraksi tersebut. Algoritma ini bisa disebut sebagai tahapan praproses data. Proses-proses yang dilakukan pada algoritma ini adalah sebagai berikut:

   1. Tokenisasi

Tokenisasi adalah suatu proses untuk membagi suatu kalimat ke dalam bentuk unit-unit kecil berupa kata [12]. Proses tokenisasi pada bahasa Indonesia, pemenggalan kata dapat dilakukan dengan mudah karena struktur kalimat sudah jelas. Contohnya pada kalimat "Ketiga, mengurangi penerbitan obligasi pemerintah." menghasilkan lima token, yaitu: "Ketiga", "mengurangi", "penerbitan", "obligasi", "pemerintah".

2. Penghapusan *Stopword*
   Setelah dilakukan tokenisasi kalimat menjadi kata-kata, proses selanjutnya yaitu penghapusan *stopword*. *Stopword* yaitu kata-kata yang sering muncul pada suatu dokumen atau kata umum (*common words*), sehingga tidak memberikan informasi penting, seperti kata penghubung dan kata ganti orang [12]. Penghapusan *stopword* ini bertujuan agar kata-kata yang nantinya akan digunakan hanya yang memiliki arti penting dan memberikan suatu informasi penting.

3. *Stemming*
   Setelah menghapus *stopword*, dilakukan proses *stemming*. Proses *stemming* adalah proses suatu kata dikembalikan pada bentuk dasar kata tersebut [12]. Membuang semua awalan, sisipan atau akhiran hingga menjadi kata dasar yang sesuai dengan bahasa Indonesia yang baik dan benar. Proses ini menggunakan pencocokan kata artikel dengan kata pada kamus bahasa Kateglo [5] yang sudah berbentuk kata dasar. Sebagai contoh, kata "menghapus" menjadi "hapus" dan sebagainya.

4. Pembentukan Informasi *Co-occurrence*
   Proses dari algoritma pemrosesan teks yang terakhir adalah pembentukan informasi *co-occurrence* kata. Elemen penyusun informasi *co-occurrence* adalah kata yang muncul terlebih dahulu disebut sebagai kata terdaftar *X*, kata *co-occurrence Y* yaitu kata yang selanjutnya muncul dan frekuensi (*f*) kemunculan dua kata tersebut. Jika pasangan kata tersebut muncul pada banyak dokumen, maka frekuensi akan diakumulasikan. Contohnya, jika terdapat tiga kata berikut sesuai dengan urutan kemunculannya, yaitu "tanam", "modal", "perusahaan", informasi co-occurrence yang dapat dibentuk dari kata-kata tersebut sesuai dengan persaamaan (*X*, *Y*, *f*) adalah (tanam, modal, 1) dan (modal, perusahaan, 1). Proses ini dilakukan hingga seluruh kata pada kumpulan dokumen artikel berita mempunyai informasi *co-occurrence*.

2. **Algoritma Pengelompokan Kata**
   Sebelum melakukan pengelompokan kata, dibentuk vektor atribut yaitu vektor kata dan vektor *node* dengan menggunakan informasi *co-occurrence*. Vektor kata terdiri dari kata *co-occurrence* dan frekuensi *co-occurrence* kata, sedangkan vektor *node* berisi kumpulan vektor kata dari kata terdaftar yang besangkutan dengan *node* tersebut. Contoh dari vektor kata adalah sebagai berikut:

$$WV_{\boldsymbol{X}} = \begin{bmatrix} f_{X_1} \\ f_{X_2} \\ f_{X_3} \\ \vdots \\ f_{X_n} \end{bmatrix} \begin{matrix} k_{X_1} \\ k_{X_2} \\ k_{X_3} \\ \vdots \\ k_{X_n} \end{matrix} \quad (1)$$

$VW_{\boldsymbol{X}}$ adalah vektor kata berupa matriks dari kata *X* yang memiliki kata *co-occurrence k* sebanyak *n* dan frekuensi *co-occurrence f* sebanyak *n*. Sedangkan untuk contoh vektor *node* adalah sebagai berikut:

$$NV_{\boldsymbol{i}} = \begin{bmatrix} WV_1 \\ WV_2 \\ WV_3 \\ \vdots \\ WV_m \end{bmatrix} \quad (2)$$

$NV_{\boldsymbol{i}}$ adalah vektor *node* dengan indeks *i* berupa matriks yang berisi vektor kata *WV* sebanyak *m*.

Pengelompokan atau *clustering* dilakukan terhadap kata-kata hasil ekstraksi berdasarkan nilai dari jarak antar kata. Metode untuk menghitung jarak antar kata yang telah direpresentasikan ke dalam bentuk vektor kata adalah metode *cosine similarity* [13]. Sedangkan, metode yang akan digunakan untuk pengelompokan adalah metode *hierarchical clustering* [12] dengan pendekatan *agglomerative* atau *bottom-up*, yaitu metode untuk melakukan pengelompokan dengan cara membandingkan nilai kedekatan antar vektor kata dari perhitungan *cosine similarity*, lalu menggabungkan dua vektor kata jika terdapat nilai dengan jarak terdekat, hingga semua menjadi sebuah kelompok besar atau hingga pada kondisi tertentu yang diinginkan atau kondisi *terminate*.

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

$$D(WV_A, WV_B) = \frac{\sum_{i=1}^{t}(f_{A_i} \cdot f_{B_i})}{\sqrt{\sum_{i=1}^{t} {f_{A_i}}^2} \sqrt{\sum_{i=1}^{t} {f_{B_i}}^2}} \quad (3)$$

Distance atau jarak antara vektor kata WV_A kata *A*, dan vektor kata WV_B kata *B* dihitung dengan menggunakan nilai t yaitu jumlah perbedaan 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*. Penghitungan jarak dilakukan pada semua kata mengacu pada informasi *co-occurrence* yang terbentuk atau vektor kata dari kata tersebut. Misalnya, pada vektor kata 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*.

Setelah dilakukan penghitungan jarak antar vektor kata, hasil dari penghitungan tersebut digunakan sebagai nilai masukan untuk melakukan pengelompokan atau *clustering*. *Clustering* secara hirarkis dengan pendekatan *bottom-up* dilakukan dari bagian terkecil yaitu dari nilai-nilai masukan hasil perhitungan jarak antar vektor kata. Hasil dari clustering hirarkis direpresentasikan dalam *dendogram* [12], yaitu grafik yang menunjukkan penggabungan vektor kata berdasarkan urutan penggabungan vektor kata tersebut. Sebagai contoh, terdapat matriks yang menunjukkan nilai jarak antara vektor kata WV_A, hingga vektor kata WV_F sebagai berikut:

	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 antar vektor kata berdasarkan matriks di atas adalah jarak antara vektor kata WV_Ddan vektor kata WV_F yaitu 0.5. Kemudian, kedua vektor kata tersebut digabungkan, hasilnya adalah 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 untuk mencari nilai perubahan jarak tersebut adalah sebagai berikut:

$$D\big((WV_D, WV_F), WV_A\big) = \min\big(D(WV_D, WV_A), D(WV_F, WV_A)\big)$$
$$D\big((WV_D, WV_F), WV_A\big) = \min(3.61, 3.20) \quad (4)$$
$$D\big((WV_D, WV_F), WV_A\big) = 3.20$$

Mencari jarak terkecil atau minimal antara vektor kata yang digabung dengan vektor kata yang besangkutan, dalam hal ini jarak minimal dari vektor kata WV_D terhadap WV_A dan WV_F terhadap WV_A. Kemudian jarak minimal tersebut yang akan menjadi jarak yang baru.

Dari contoh tersebut di atas, vektor kata WV_D dikelompokkan 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 didapat. *Cluster* direpresentasikan dalam vektor *node* dengan jumlah vektor kata sebanyak *m*.

3. **Algoritma Penghitungan Kemiripan (*similarity measurement*)**
   Setelah terbentuk *cluster-cluster*, proses selanjutnya adalah menghitung kemiripan antar kata yang terdapat pada *cluster-cluster* tersebut. Tujuannya adalah untuk memperoleh nilai kemiripan kata dan posisi hirarki kata. Penghitungan tersebut akan dilakukan dengan menggunakan metode Kullback-Leiber *divergence*, yaitu menghitung kemiripan antar kata dengan mempertimbangkan hirarki antar kata tersebut [11]. Metode ini tidak memiliki simetri, artinya jarak antara dua kata, kata *A* dengan kata *B* tidak sama dengan jarak kata *B* dengan 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:

$$P_{A_i} = \frac{f_{A_i}}{\sum_{i=1}^{t} f_{A_i}} \tag{5}$$

$$P_{B_i} = \frac{f_{B_i}}{\sum_{i=1}^{t} f_{B_i}} \tag{6}$$

   Nilai t adalah jumlah perbedaan kata *co-occurrence* dari 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*. Kata *A* dan kata *B* masing-masing telah direpresentasikan dalam vektor kata WV_A untuk kata *A* dan WV_B untuk kata *B*. Sehingga nilai Kullback-Leiber *divergence*, $KL(WV_A, WV_B)$ yaitu:

$$KL(WV_A, WV_B) = \sum_{i=1}^{t} P_{A_i} \log \frac{P_{A_i}}{P_{B_i}} \tag{7}$$

   Didefinisikan beberapa pengecualian sebagai berikut:

$$P_{A_i} \log \frac{P_{A_i}}{P_{B_i}} = \begin{cases} 0 & \text{jika } P_{A_i} = 0 \\ P_{A_i}(\log P_{A_i}) & \text{jika } P_{B_i} = 0 \end{cases} \tag{8}$$

   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*.

4. **Algoritma Pembaruan Struktur *Tree***

Proses yang dilakukan hingga penghitungan kemiripan menggunakan Kullback-Leiber *divergence* adalah proses inisiasi awal pembentukan struktur *tree* tesaurus, dengan asumsi pembangunan tesaurus dimulai dari awal. Setelah dilakukan inisiasi pembentukan struktur *tree* tesaurus, pembaruan struktur *tree* akan dilakukan ketika tesaurus mendapatkan masukan berupa artikel berita baru. Pada proses ini yang nantinya menyebabkan tesaurus bersifat dinamis. Proses-proses yang dilakukan untuk melakukan pembaruan *tree* tesaurus adalah sebagai berikut:

1. Penghapusan Vektor Kata
   Pada artikel berita baru dilakukan praproses data untuk melakukan ekstraksi kata-kata. Kemudian dilakukan pengecekan, jika suatu kata yang terdapat di artikel baru, kata *X*, sudah terdaftar dalam bentuk vektor kata pada salah satu *cluster* struktur *tree* tesaurus, vektor kata tersebut akan dihapus untuk sementara dengan persamaan sebagai berikut:

   $$NV_i' = NV_i - WV_X \tag{9}$$

   NV_i adalah vektor *node i* tempat vektor kata WV_X dari kata *X* berada. NV_i diperbarui menjadi NV_i' ketika WV_X dihapus. Jika terjadi suatu kasus $NV_i = WV_X$ atau pada *cluster* tersebut hanya terdapat satu vektor kata, maka vektor *node i* akan dihapus, sedangkan *child* vektor *node i* akan terhubung dengan *parent* dari vektor *node i*.

   Setelah vektor kata WV_X dihapus, dilakukan pembaruan pada informasi *co-occurrence* kata *X*, yaitu penambahan jumlah frekuensi *co-occurrence* atau penambahan jumlah kata *co-occurrence* dari kata *X*, yaitu $(X, Y, f + f')$ atau $(X, Y + Y', f)$ dengan nilai f' adalah nilai penambahan frekuensi dari suatu kata *co-occurrence Y* dan nilai Y' adalah suatu penambahan kata *co-occurrence* baru.

2. Pembentukan Vektor Kata
   Proses yang dilakukan selanjutnya adalah membentuk kembali vektor kata yang dihapus sebelumnya dengan menggunakan informasi *co-occurrence* yang baru, $(X, Y, f + f')$ atau $(X, Y + Y', f)$.

3. Pencarian *Similar Node*
   Setelah vektor kata baru terbentuk, proses yang dilakukan selanjutnya adalah melakukan penghitungan jarak *cosine similarity* antara vektor kata baru dengan vektor *node* atau *cluster*. Jika vektor kata baru tersebut sudah terkelompokkan pada salah satu *cluster*, maka dilakukan pencarian *similar node* antara vektor kata baru dengan anggota *cluster* tersebut menggunakan persamaan Kullback-Leiber *divergence* sebagai berikut:

   $$sim\_node(NV_i, WV_X) = \min\big(KL(NV_i, WV_X), KL(WV_X, NV_i)\big)$$

$$= \min(\min(KL(WV_1, WV_X), \cdots, KL(WV_z, WV_X)), \min(KL(WV_X, WV_1), \cdots, KL(WV_X, WV_z))) \quad (10)$$

$WV_1, \cdots, WV_z \in NV_i$, vektor kata WV_1 hingga WV_z adalah vektor kata anggota elemen dari vektor node NV_i.

Perhitungan *similar node* antara vektor *node* NV_i dan vektor kata WV_X didapatkan dengan menghitung nilai minimal Kullback-Leiber *divergence* antara vektor *node* NV_i dan vektor kata WV_X dan bentuk simetrisnya. Nilai Kullback-Leiber *divergence* antara vektor *node* NV_i dan vektor kata WV_X didapatkan dengan cara menghitung nilai minimal dari Kullback-Leiber *divergence* vektor kata penyusun NV_i, yaitu vektor kata WV_1 hingga WV_z, dengan vektor kata WV_X. Nilai indeks *z* tergantung banyaknya vektor kata yang menyusun NV_i. Begitu juga pada bentuk simetrisnya. Setelah didapatkan nilai dari kedua perhitungan tersebut, diambil nilai terkecil yang selanjutnya nilai tersebut adalah nilai dari *similar node* antara vektor *node* NV_i dan vektor kata WV_X.

Sedangkan untuk posisi hirarki, NV_i adalah *broader term* jika nilai $KL(NV_i, WV_X)$ bernilai kecil atau mendekati 0, sebaliknya WV_X adalah *broader term* jika nilai $KL(NV_i, WV_X)$ bernilai besar atau menjauhi 0.

4. Penyisipan *Node*
   Proses selanjutnya adalah penyisipan *node*. WV_X disisipkan sebagai *parent node* dari NV_n jika nilai perhitungan Kullback-Leiber *divergence* bernilai besar atau menjauhi 0. Sebaliknya jika perhitungan Kullback-Leiber *divergence* bernilai kecil atau mendekati 0, maka vektor kata WV_X disisipkan sebagai *child node* dari NV_n dan dilakukan penghitungan Kullback-Leiber *divergence* kembali antara vektor kata WV_X dengan anggota dari vektor node NV_n hingga kondisi vektor kata WV_X berada di posisi yang tepat.

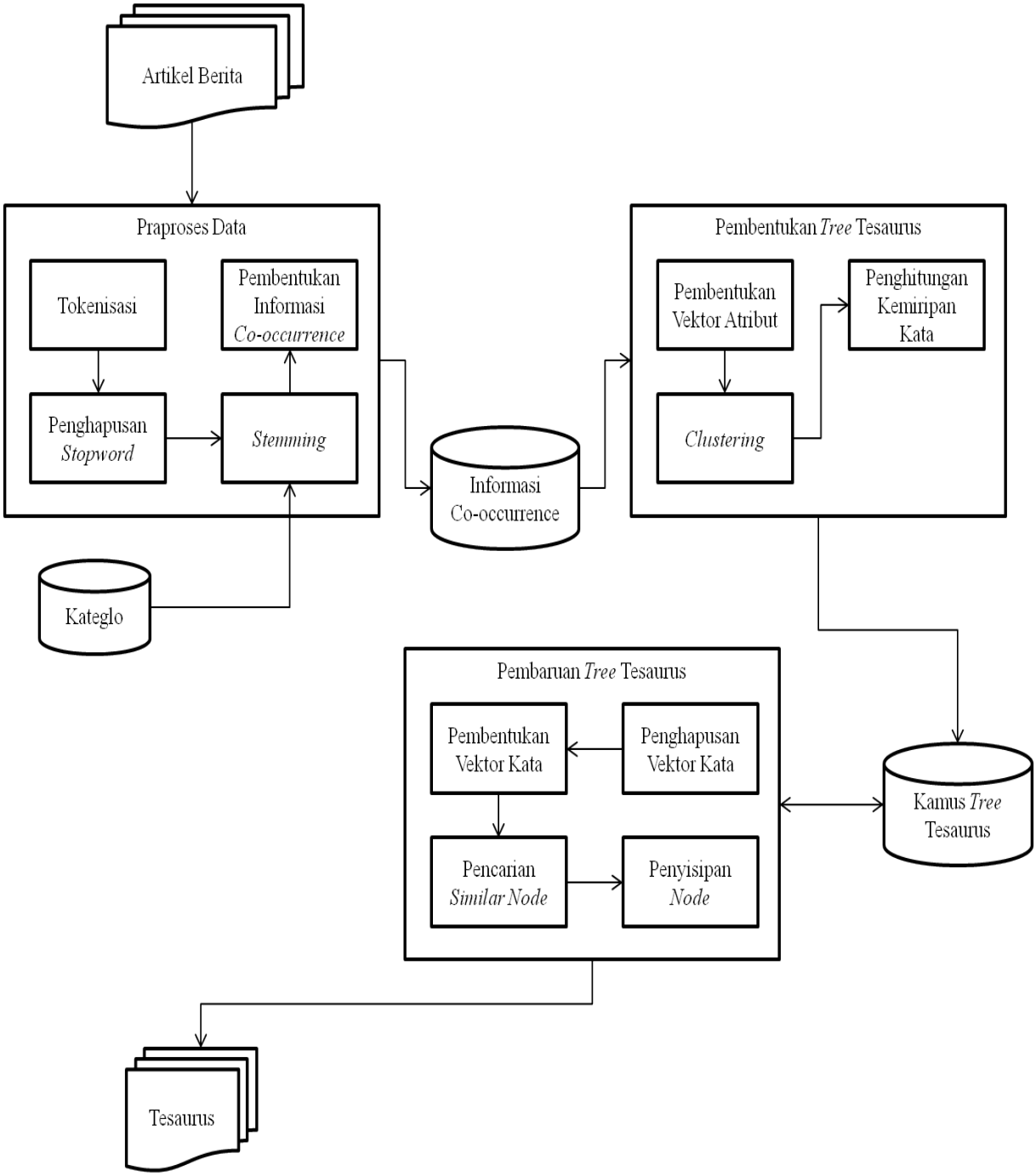
   Jika terdapat vektor kata yang merupakan vektor kata baru, vektor kata baru tersebut akan langsung masuk pada proses pencarian *similar node*, tanpa melalui proses sebelumnya yaitu penghapusan vektor kata dan pembentukan vektor kata menggunakan informasi *co-occurrence* baru.

9. RINGKASAN ISI TUGAS AKHIR

Sistem yang akan dikembangkan pada Tugas Akhir ini terbagi menjadi tiga proses utama, yaitu praproses data, pembentukan tree tesaurus dan proses pembaruan *tree* tesaurus. Gambar 1 adalah gambaran umum dari Tugas Akhir yang akan dikerjakan. Penjelasan untuk setiap proses adalah sebagai berikut:

1. Praproses Data

Praproses data adalah proses pertama yang dilakukan terhadap kumpulan artikel berita. Tujuan proses ini adalah untuk melakukan ekstraksi kata dan membentuk informasi *co-occurrence* [10]. Proses-proses yang termasuk di dalamnya adalah tokenisasi, penghapusan *stopword*, *stemming* menggunakan Kateglo [5] dan pembentukan informasi *co-occurrence*.



Gambar 1. Gambaran Umum Sistem

2. Pembentukan *Tree* Tesaurus
   Pembentukan *tree* tesaurus diawali dengan membentuk vektor kata dari suatu kata beserta kata *co-occurrence* dari kata tersebut dan vektor *node*. Kemudian dilakukan penghitungan jarak antar vektor kata menggunakan *cosine similarity*. Tujuan penghitungan jarak ini adalah mengetahui kedekatan antar vektor kata yang selanjutnya akan digunakan dalam proses pengelompokan atau clustering. Metode yang digunakan untuk proses *clustering* adalah *hierarchical clustering* dengan pendekatan *bottom-up*. Setelah pembentukan proses *clustering* selesai, bentuk kata yang tersimpan dalam setiap *cluster* berupa vektor kata, sedangkan *cluster* adalah vektor *node*. Kemudian dibentuk struktur *tree* pada masing-masing *cluster* menggunakan hasil penghitungan dari Kullback-Leiber *divergence*, yaitu penghitungan kemiripan mempertimbangkan hirarki kata [11].

3. Pembaruan *Tree* Tesaurus
   Proses pembaruan *tree* tesaurus dilakukan ketika terdapat masukan berupa artikel berita baru. Pada proses ini yang menyebabkan tesaurus akan bersifat dinamis. Setiap terdapat perubahan informasi *co-occurrence* kata atau penambahan kata baru dapat dilakukan secara otomatis. Proses yang dilakukan adalah sebagai berikut:
   1. Penghapusan vektor kata
      Jika suatu kata yang terdapat di artikel baru sudah terdaftar sebagai kata berbentuk vektor kata pada *tree* tesaurus, vektor kata tersebut akan dihapus sementara dari *tree*. Jika belum terdaftar, maka akan dimasukkan ke dalam *cluster* atau vektor *node* yang memiliki jarak terdekat dengan kata tersebut dan kemudian dihitung nilai kemiripannya dengan kata-kata anggota *cluster*. Sehingga dapat dicari kata yang merupakan *similar node* dengan kata baru tersebut.
   2. Pembentukan vektor kata
      Setelah vektor kata dihapus, vektor tersebut dibentuk kembali dengan menggunakan informasi *co-occurrence* yang baru. Informasi *co-occurrence* yang baru dapat berupa perubahan frekuensi *co-occurrence* kata atau perubahan jumlah kata *co-occurrence*.
   3. Pencarian *similar node*
      Proses selanjutnya adalah melakukan pencarian *similar node* dari vektor kata baru. Dilakukan penghitungan jarak antara kata yang memiliki vektor kata dengan *cluster*, kata tersebut dimasukkan pada *cluster* yang memiliki jarak terdekat. Kemudian dilakukan penghitungan nilai kemiripan Kullback-Leiber *divergence* dengan kata-kata anggota cluster yang sama untuk menentukan posisi kata tersebut pada *tree* tesaurus.
   4. Penyisipan *node*
      Setelah dihitung kemiripannya, kata tersebut disisipkan pada similar node, sebagai *parent* atau *child* tergantung pada hasil perhitungan Kullback-Leiber *divergence* sebelumnya. Jika hasil dari penghitungan bernilai kecil atau mendekati 0 antara kata dengan *similar node*, berarti kata tersebut adalah kata yang lebih umum dari *similar node* dan sebaliknya.

10.METODOLOGI

a. Penyusunan proposal tugas akhir

Penyusunan proposal Tugas Akhir ini dilakukan dengan tujuan untuk merumuskan masalah yang berhubungan dengan topik Tugas Akhir yang diusulkan serta menetapkan desain dan gambaran dasar dari sistem yang akan dibuat dan dikembangkan dalam pelaksanaan Tugas Akhir.

b. Studi literatur

Untuk membantu dalam pengerjaan dalam membangun sistem di Tugas Akhir ini, diperlukan untuk mempelajari lebih lanjut tentang penggunaan-penggunaan metode dan pustaka (*library*) yang terkait dengan sistem yang akan dibuat, antara lain adalah sebagai berikut:

- Pembentukan infomasi *co-occurrence*
- Vektor atribut, yaitu vektor kata dan vektor *node*
- Metode pengelompokan kata, *hierarchical clustering*
- Metode penghitungan jarak antar kata, *cosine similarity*
- Metode penghitungan kemiripan, Kullback-Leiber *divergence*
- *Library* PHP Tree

c. Analisis dan desain perangkat lunak

Tesaurus yang akan dibangun, memiliki beberapa modul yaitu:

- Modul praproses data atau pemrosesan teks
- Modul pengelompokan kata
- Modul penghitungan kemiripan kata
- Modul pembaruan *tree* tesaurus

Fungsi dari modul-modul tersebut telah dijelaskan pada bab 9, Ringkasan Tugas Akhir. Selain modul-modul yang akan dibuat, dibutuhkan desain perangkat lunak sebagai berikut:

- Desain basis data untuk mengakomodasi penyimpanan data pada implementasi Tugas Akhir.

d. Desain antarmuka sistem dan implementasi perangkat lunak

Tugas Akhir ini akan diimplementasikan menggunakan kakas bantu dan bahasa pemrograman sebagai berikut:

- Bahasa pemrograman: PHP 5.5.6
- Kerangka Kerja: CodeIgniter 3.0.3
- Web server: Apache 2.4.7
- Basis data: MySQL 5.6.14
- Kakas bantu basis data: HeidiSQL 19.1.0.4867

- Desain antarmuka: HTML5, CSS3

Desain dan implementasi antarmuka dari sistem yang akan dibuat hanya berupa tampilan situs web sederhana yang terdiri dari kotak pencarian, tabel dan formulir masukan.

Selain kakas bantu dan bahasa pemrograman di atas, Tugas Akhir ini juga akan diimplementasikan menggunakan pustaka (*library*) sebagai berikut:

- Tree
  Library PHP yang digunakan untuk membantu membentuk *node* atau membantu membangun suatu struktur *tree*.

e. Pengujian dan evaluasi

Pengujian yang akan dilakukan pada sistem bertujuan untuk mengetahui perubahan nilai kemiripan yang terjadi dan perubahan posisi kata pada struktur *tree* tesaurus. Perubahan tersebut tentunya akan berpengaruh pada keluaran kata yang diberikan oleh tesaurus. Skenario pengujian yang akan dilakukan adalah sebagai berikut:

1. Melakukan masukan berupa kata atau *query* ke dalam tesaurus.
2. Mencatat keluaran kata yang diberikan.
3. Mengunggah ke dalam sistem sejumlah artikel berita baru.
4. Pemrosesan artikel berita baru oleh sistem.
5. Melakukan *query* dengan kata yang sama.
6. Mencatat keluaran kata yang diberikan.
7. Memberikan keterangan pada catatan uji coba jika terdapat perbedaan keluaran kata yang diberikan.

Pengujian sistem dilakukan lima kali dengan perbedaan jumlah artikel yang diunggah ke dalam sistem, yaitu 10, 20, 30, 40 dan 50 artikel berita baru.

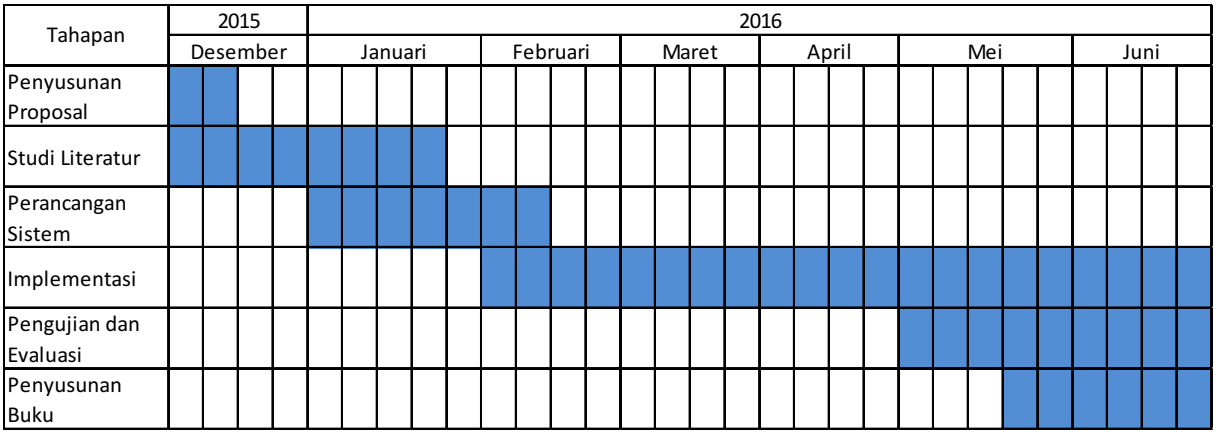
f. Penyusunan buku Tugas Akhir

Pada tahap ini dilakukan penyusunan laporan yang menjelaskan dasar teori dan metode yang digunakan dalam tugas akhir ini serta hasil dari implementasi aplikasi perangkat lunak yang telah dibuat. Sistematika penulisan buku tugas akhir secara garis besar antara lain:

1. Pendahuluan
   a. Latar Belakang
   b. Rumusan Masalah
   c. Batasan Tugas Akhir
   d. Tujuan
   e. Metodologi
   f. Sistematika Penulisan
2. Tinjauan Pustaka
3. Desain dan Implementasi
4. Pengujian dan Evaluasi

5. Kesimpulan dan Saran
6. Daftar Pustaka

11. JADWAL KEGIATAN



12. 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 12 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.