TUGAS AKHIR – KI141502

PEMBENTUKAN TESAURUS PADA CROSS-LINGUAL TEXT DENGAN PENDEKATAN CONSTRAINT SATISFACTION PROBLEM

UMY CHASANAH NOOR RIZQI
NRP. 5113 100 082

Dosen Pembimbing 1
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing 2
Diana Purwitasari, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

TUGAS AKHIR – KI141502

PEMBENTUKAN TESAURUS PADA CROSS-LINGUAL TEXT DENGAN PENDEKATAN CONSTRAINT SATISFACTION PROBLEM

UMY CHASANAH NOOR RIZQI
NRP. 5113 100 082

Dosen Pembimbing 1
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Dosen Pembimbing 2
Diana Purwitasari, S.Kom., M.Sc.

JURUSAN TEKNIK INFORMATIKA
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya 2017

(Halaman ini sengaja dikosongkan)

FINAL PROJECT – KI141502

ESTABLISHMENT OF THESAURUS ON CROSS-LINGUAL TEXT WITH CONSTRAINT SATISFACTION PROBLEM APPROACH

UMY CHASANAH NOOR RIZQI
NRP. 5113 100 082

Supervisor 1
Dr.Eng. Chastine Fatichah, S.Kom., M.Kom.

Supervisor 2
Diana Purwitasari, S.Kom., M.Sc.

DEPARTMENT OF INFORMATICS
Faculty of Information Technology
Sepuluh Nopember Institute of Technology
Surabaya 2017

(Halaman ini sengaja dikosongkan)

LEMBAR PENGESAHAN

PEMBENTUKAN TESAURUS PADA CROSS-LINGUAL TEXT DENGAN PENDEKATAN CONSTRAINT SATISFACTION PROBLEM

TUGAS AKHIR

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

Oleh:
UMY CHASANAH NOOR RIZQI
NRP. 5113100082

Disetujui oleh Pembimbing Tugas Akhir:

1. Dr.Eng. Chastine Fatichah, S.Kom., M.Kom. .....
   NIP. 197512202001122002 (Pembimbing 1)

2. Diana Purwitasari, S.Kom., M.Sc. .....
   NIP. 197804102003122001 (Pembimbing 2)

SURABAYA
JUNI 2017

(Halaman ini sengaja dikosongkan)

PEMBENTUKAN TESAURUS PADA CROSS-LINGUAL TEXT DENGAN PENDEKATAN CONSTRAINT SATISFACTION PROBLEM

Nama : **Umy Chasanah Noor Rizqi**
NRP : **5113100082**
Jurusan : **Teknik Informatika**
Fakultas Teknologi Informasi ITS
Dosen Pembimbing I : **Dr.Eng. Chastine Fatichah, S.Kom., M.Kom**
Dosen Pembimbing II : **Diana Purwitasari, S.Kom., M.Sc.**

ABSTRAK

Pencarian dokumen adalah hal yang esensial dalam bidang text mining. Dokumen tugas akhir dan tesis sering kali disediakan dalam dua bahasa, yaitu bahasa Indonesia dan Inggris. Dalam pencarian, setiap mahasiswa memiliki kecenderungan mencari dokumen dengan menggunakan kata kunci dengan bahasa tertentu. Tujuan dari pembuatan Tugas Akhir ini adalah untuk membangun cross-lingual tesaurus bahasa Indonesia dan bahasa Inggris dengan pendekatan Constraint Satisfaction Problem. Dalam penelitian ini digunakan data Tugas Akhir serta Tesis mahasiswa FTIF, Jaringan Cerdas Multimedia, Statistika dan Teknik Multimedia Jaringan di Institut Teknologi Sepuluh Nopember.

Pada pengolahan dokumen dilakukan beberapa langkah yaitu document alignment, ekstraksi kata, pembobotan kata, dan pembentukan informasi co-occurrence, yang selanjutnya dilakukan Constraint Satisfaction Problem dengan backtracking sebagai solusi pencarian yang merupakan perbaikan dari metode

bruteforce. Pembobotan menggunakan TF-IDF (term frequency – inverse document frequency)

Hasil dari proses pembangunan tesaurus, pada proses document alignment membutuhkan waktu terlama dalam pembangunan tesaurus, yaitu 10.745 detik, sedangkan yang tercepat adalah proses Penghitungan Relevance Weight dengan waktu 10 detik. Tesaurus yang dibentuk dengan menggunakan CSP menghasilkan precision 91,38% sedangkan tesaurus yang dibentuk tanpa menggunakan CSP menghasilkan precision 45,23%.. Pencarian dokumen menggunakan tesaurus menghasilkan recall 86,67% precision 100% dan akurasi 86,67%.

Kata kunci: Backtracking, Co-occurrence, Constraint Satisfaction Problem , Cross-lingual, Tesaurus

ESTABLISHMENT OF THESAURUS ON CROSS-LINGUAL TEXT WITH CONSTRAINT SATISFACTION PROBLEM APPROACH

Name	**: Umy Chasanah Noor Rizqi**
NRP	**: 5113100082**
Department	**: Department of Informatics Faculty of Information Technology ITS**
Supervisor I	**: Dr.Eng. Chastine Fatichah, S.Kom., M.Kom**
Supervisor II	**: Diana Purwitasari, S.Kom., M.Sc.**

ABSTRACT

Document search is essential in text mining. The final project and thesis document are often provided in two languages, there are Indonesian and English. To search document, each student has a tendency to search for documents by using keywords in a particular language. The purpose of this Final Project is to build cross-lingual thesaurus of Indonesian and English with approach of Constraint Satisfaction Problem. In this research used final project and thesis document FTIF, Multimedia and Network, Statistics and Multimedia Network Technique departement at Sepuluh Nopember Institute of Technology.

In the document processing, there are several steps, there are word extraction, document alignment, weighting, and co-occurrence, which is then performed by the Constraint Satisfaction Problem with backtracking as its search solution which is an improvement of bruteforce method. Weighting using TF-IDF (term frequency - inverse document frequency)

The result of the development process thesaurus on document alignment process takes the longest time in the

development of thesaurus, which is 10.745 seconds, while the fastest is the process of Calculating Relevance Weight with time 10 seconds. The thesaurus formed by using CSP produces 91.38% precision whereas the thesaurus formed without using CSP produces precision 45.23%. The search document uses a thesaurus yielding 86,67% precision 100% recall and 86.67% accuracy.

Key words: Backtracking, Co-occurrence, Constraint Satisfaction Problem, Cross-lingual, Thesaurus

KATA PENGANTAR

Segala puji dan syukur kepada Allah SWT yang telah melimpahkan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan tugas akhir yang berjudul **"PEMBENTUKAN TESAURUS PADA CROSS-LINGUAL TEXT DENGAN PENDEKATAN CONSTRAINT SATISFACTION PROBLEM"**.

Tugas akhir ini dilakukan dengan tujuan agar penulis dapat menghasilkan sesuatu atau menerapkan apa yang telah dipelajari selama pada masa perkuliahan dan untuk memenuhi salah satu syarat memperoleh gelar Sarjana di Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Penulis mengucapkan banyak terima kasih kepada semua pihak yang telah memberikan dukungan dan bimbingan kepada penulis, baik secara langsung maupun tidak langsung dalam proses pengerjaan tugas akhir ini, yaitu kepada:

1. Allah SWT dan Nabi Muhammad SAW.
2. Keluarga penulis, Bapak Djonet Boedi Wiyanto, Ibu Sofiatun, Mbak Fatma Noor Annisa dan Mas Muhammad Ridlo yang selalu memberikan semangat, perhatian dan doa kepada penulis selama menjalani masa studi dan pengerjaan tugas akhir penulis di Teknik Informatika.
3. Ibu Dr. Eng. Chastine Fatichah, S.Kom., M.Kom sebagai dosen wali dan dosen pembimbing I yang telah memberikan bimbingan dan arahan kepada penulis dalam pengerjaan tugas akhir ini dan selama masa perkuliahan.
4. Ibu Diana Purwitasari, S.Kom., M.Sc. sebagai dosen pembimbing II yang telah memberikan arahan dan bantuan kepada penulis dalam pengerjaan tugas akhir dan penulisan buku tugas akhir ini hingga selesai.
5. Grup "ulululululu", yang selalu memberikan penyemangat sendiri bagi penulis.

6. Bapak dan Ibu dosen Jurusan Teknik Informatika ITS lainnya yang telah banyak menyampaikan ilmu dan bimbingan yang tak ternilai harganya bagi penulis.
7. Dan kepada seluruh pihak yang tidak bisa penulis sebutkan satu persatu yang telah memberikan dukungan dan semangat kepada penulis untuk menyelesaikan tugas akhir ini.
8. Terkhusus untuk M. Faris Ponighzwa R atas bantuan, dorongan, motivasi, semangat, kesabaran, kesetiaanya dalam menemani penulis mengerjakan Tugas Akhir.

Penulis menyadari bahwa masih terdapat banyak kekurangan dalam pengerjaan tugas akhir ini. Penulis sangat mengharapkan kritik dan saran dari pembaca untuk perbaikan dan pembelajaran kedepannya. Semoga tugas akhir yang penulis buat dapat memberikan manfaat.

Surabaya, Juni 2017

Penulis

DAFTAR ISI

DAFTAR GAMBAR

(Halaman ini sengaja dikosongkan)

DAFTAR TABEL

DAFTAR KODE SUMBER

(Halaman ini sengaja dikosongkan)

BAB I
PENDAHULUAN

1.1 Latar Belakang

Pencarian dokumen adalah hal yang esensial dalam bidang *text mining*. Dokumen tugas akhir dan tesis sering kali disediakan dalam dua bahasa, yaitu bahasa Indonesia dan Inggris. Dalam pencarian, setiap mahasiswa memiliki kecenderungan mencari dokumen dengan menggunakan kata kunci dengan bahasa tertentu.

Cross-lingual tesaurus adalah padanan kata dari suatu kalimat pada bahasa tertentu ke dalam dua bahasa, yaitu bahasa Indonesia dan bahasa Inggris. *Cross-lingual* tesaurus yang terbentuk bisa sebagai sinonim, terjemahan kata maupun akronim. *Cross-lingual* tesaurus memungkinkan pengguna mendapatkan dokumen dengan input bahasa yang berbeda dengan bahasa pada dokumen.

Cross-lingual information retrieval memproses *query* pengguna dalam satu bahasa dan mengambil dokumen dalam bahasa lain yang relevan [1]. Pendekatan kosakata mengadopsi satu set kosakata yang telah ditetapkan untuk permintaan pengguna dan pengindeksan dokumen. Namun, efektivitas pengambilan sangat tergantung pada pemilihan kosakata. Pendekatan berbasis pengetahuan mengadopsi ontologi dan kamus untuk menerjemahkan *query* dari satu bahasa ke bahasa lain. Namun, banyak istilah teknis, singkatan, nama orang, organisasi, dan peristiwa mungkin tidak *term*asuk dalam kamus. Terjemahan juga bisa ambigu dalam beberapa kasus. Selama penerjemahan, kalimat dapat dibagi, digabungkan, dihapus, dimasukkan atau diatur kembali oleh penerjemah[2].

Pada penelitian Bel et al[3] *cross-lingual* tesaurus digunakan untuk terjemahan. Pembentukan tesaurus digunakan untuk *query expansion* pada pencarian dokumen. *Query expansion* merupakan proses merumuskan *query* asli untuk meningkatkan kinerja pengambilan dalam operasi pencarian informasi. Dalam konteks mesin pencari, *query expansion* melakukan evaluasi masukan

pengguna dan memperluas permintaan pencarian untuk mencocokkan dokumen tambahan.

Pembentukan *Cross-lingual* tesaurus, dibutuhkan *parallel corpus*, *parallel corpus* adalah kumpulan dari pasangan dokumen dalam dua bahasa (bahasa Indonesa dan bahasa Inggris), dimana pasangan dokumen adalah terjemahan dari antar kedua dokumen. Sebuah *parallel corpus* adalah kumpulan dokumen dalam bahasa masing-masing dan dikombinasikan atas dasar kesamaan konten pada judul.

Constraint Satisfaction Problem (CSP) merupakan salah satu metode yang dapat dipergunakan untuk pembentukan *cross lingual* tesaurus. *Constraint Satisfaction Problem* (CSP) merupakan salah satu metode yang dapat dipergunakan untuk pembentukan *cross lingual* tesaurus dengan mempertimbangkan *constraint* sehingga diharapkan kata yang dihasilkan lebih relevan. Untuk memenuhi *constraint satisfaction problem* dibutuhkan suatu metode pencarian, salah satunya adalah *backtracking*. *Backtracking* adalah algoritma untuk *mencari* solusi persoalan secara lebih ringkas dan juga merupakan perbaikan dari algoritma *brute-force*.

Cross lingual tesaurus memungkinkan pengguna untuk memperbaiki pencarian dengan otomatis. Pendekatan ini membahas relevansi informasi dan memungkinkan pengambilan informasi di beberapa bahasa.

1.2 Rumusan Masalah

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

1. Bagaimana cara mengumpulkan *parallel corpus*?
2. Bagaimana cara ekstraksi kata penting?
3. Bagaimana *Constraint Satisfaction Problem* membangun tesaurus?
4. Bagaimana melakukan evaluasi pada tesaurus yang telah terbentuk?

1.3 Batasan Masalah

Batasan masalah dalam pengerjaan tugas akhir ini adalah sebagai berikut:

1. Menggunakan data Tugas Akhir serta Tesis mahasiswa FTIF, Jaringan Cerdas Multimedia, Statistika, Teknik Multimedia Jaringan di Institut Teknologi Sepuluh Nopember.
2. Bahasa pemrograman yang digunakan adalah PHP, basis data MySQL, dan Python.

1.4 Tujuan

Adapun tujuan dari pembuatan Tugas Akhir ini adalah untuk membangun *cross-lingual* tesaurus bahasa Indonesia dan bahasa Inggris dengan pendekatan *Constraint Satisfaction Problem*.

1.5 Manfaat

Manfaat dari tugas akhir ini adalah C*ross-lingual* tesaurus yang dibangun diharapkan dapat digunakan sebagai rujukan mencari kemiripan dari suatu kata bahasa Indonesia dan Bahasa Inggris. *Cross-lingual* tesaurus diharapkan dapat digunakan sebagai terjemahan kata.

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 parallel *corpus*
   b. Metode penghitungan jarak antar kata, *jacard distance*.

c. Metode pembentukan informasi *co-occurrence*.
d. Metode *Constraint Satisfaction Problem*.
e. Metode *backtracking*.
f. Kamus bahasa Inggris - Indonesia, gkamus.
g. Pustaka *fetching* data dari *web*, urllib2
h. Pustaka pembangkitan *array*, NumPy.
i. Pustaka penampilan data hasil.

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, perancangan basis data dan perancangan antar muka.
3. Implementasi sistem
   Pada tahap ini dilakukan pembangunan tesaurus menggunakan informasi *co-occurrence* dan *Constraint Satisfaction Problem*, 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
DASAR TEORI

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 dokumen dan pembentukan informasi *co-occurrence* dari kata hasil ekstraksi. Algoritma ini bisa disebut sebagai tahapan praproses data.

2.1.1 Tokenisasi

Tokenisasi merupakan proses pengambilan kata-kata dari sebuah dokumen yang dipisahkan berdasarkan spasi. Sebagai contoh pada kalimat "Aplikasi Tumpangan Kendaraan Berbasis Android dengan Google Maps dan GPS" menghasilkan sepuluh token, yaitu: "Aplikasi", "Tumpangan", "Kendaraan", "Berbasis", "Android", "dengan", "Google" , "Maps", "dan", "GPS". Pada proses tokenisasi, yang dijadikan acuan pemisah antar kata adalah tanda baca dan spasi. Contoh program tokenisasi yang dapat diakses via *online* adalah *NLTK Tokenizer*.

2.1.2 Case Folding

Case folding merupakan tahap paling awal dan penting, tahapan ini bertujuan untuk mengkonversi keseluruhan teks dalam dokumen menjadi bentuk standar (huruf kecil atau *lowercase*). Sehingga contoh pada kalimat tokenisasi diatas akan menjadi "aplikasi", "tumpangan", "kendaraan", "berbasis", "android", "dengan", "google" , "maps", "dan", "gps".

2.1.3 Penghapusan *Stopword*

Setelah dilakukan proses tokenisasi pada judul dokumen, 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. Penghapusan *stopword* ini bertujuan agar kata-kata yang digunakan hanya kata-kata yang memiliki arti penting dan memberikan suatu informasi.

Contoh kata-kata yang termasuk kedalam *stopword* adalah : ada, yang, di, ke, agak, adalah, dan lain sebagainya [4]. Sehingga token seperti contoh diatas akan berubah menjadi "aplikasi", "tumpangan", "kendaraan", "berbasis", "android", "google", "maps", "gps". Karena kata "dengan", "dan" merupakan *stopword* dan sudah dihilangkan dari susunan kata dalam token [4]

2.1.4 Stemming

Stemming merupakan suatu proses untuk menemukan kata dasar dari sebuah kata. Proses *stemming* dilakukan dengan menghilangkan semua imbuhan (afiks) baik yang terdiri dari awalan (prefiks) sisipan (infiks) maupun akhiran (sufiks) dan kombinasi dari awalan dan akhiran (konfiks)[4].

Sehingga token seperti contoh diatas akan berubah menjadi "aplikasi", "tumpang", "kendara", "basis", "android", "google" , "maps", "gps". Pustaka Python yang bisa dipakai saat ini untuk bahasa Indonesia adalah Sastrawi dan untuk bahasa Inggris adalah *NLTK Snowball*.

2.1.5 Penghitungan *Term Frequency* (TF)

TF menyatakan jumlah berapa banyak keberadaan suatu *term* dalam satu dokumen[4]. Rumus TF sebagai mana ditunjukkan pada Persamaan (*2.1)*.

$$TF_{t,d} = \begin{cases} f_{t,d} & if\ f_{t,d} > 0 \\ 0, & if\ f_{t,d} = 0 \end{cases} \tag{2.1}$$

dimana :

t = *term*

d = dokumen

$f_{t,d}$ = frekuensi *term t* pada dokumen d

$TF_{t,d}$ = nilai TF dari *term t* pada dokumen *d*

Contoh kalimat "Aplikasi Tumpangan Kendaraan Berbasis Android dengan Google Maps dan GPS" dan kalimat 2 "Aplikasi Monitoring Keberadaan Objek Melalui Perangkat Bergerak Berbasis Android" dianggap sebagai dokumen dan setiap kata adalah term maka didapat perhitungan sesuai dengan Tabel 2.1.

Tabel 2.1. Contoh Menghitung TF

No	Term	F_{d1}	TF_{d1}	F_{d2}	TF_{d2}
1	Aplikasi	1	1	0	0
2	Monitoring	1	1	0	0
3	Keberadaan	1	1	0	0
4	Objek	1	1	0	0
5	Melalui	1	1	0	0
6	Perangkat	1	1	0	0
7	Berbasis	1	1	0	0
8	Bergerak	1	1	0	0
9	Android	0	0	1	1
10	Tumpangan	0	0	1	1
11	Kendaraan	0	0	1	1
12	Google	0	0	1	1
13	Maps	0	0	1	1
16	GPS	0	0	1	1

2.1.6 Penghitungan Invers *Document* Frequncy (IDF)

Terkadang suatu *term* muncul di hampir sebagian besar dokumen mengakibatkan proses pencarian *term* unik terganggu. IDF berfungsi mengurangi bobot suatu *term* jika kemunculannya banyak tersebar di seluruh koleksi dokumen kita[4]. Rumusnya adalah dengan inverse *document frequency*. *Document frequency* adalah seberapa banyak suatu *term* muncul di seluruh *document*

yang diselidiki. Rumus penghitungan IDF terdapat pada Persamaan (*2.2)*.

$$idf_t = log_{10}\left(\frac{N+1}{df_t}\right) \tag{2.2}$$

dimana :

t = *term*

N = total dokumen

df_t = frekuensi kemunculan *term t* pada seluruh dokumen

idf_t = nilai IDF dari *term t*

Tabel 2.2 ditampilkan contoh suatu kasus. Misalnya kita memiliki kalimat 1 "Aplikasi Tumpangan Kendaraan Berbasis Android Dengan Google Maps Dan GPS" dan kalimat 2 "Aplikasi Monitoring Keberadaan Objek Melalui Perangkat Bergerak Berbasis Android" dianggap sebagai dokumen dan setiap kata adalah *term* maka didapat sesuai dengan:

Tabel 2.2 Contoh Penghitungan IDF

No	*Term*	DF	IDF
1	Aplikasi	2	0,47
2	Monitoring	1	0,77
3	Keberadaan	1	0,77
4	Objek	1	0,77
5	Melalui	1	0,77
6	Perangkat	1	0,77
7	Berbasis	2	0,47
8	Bergerak	1	0,77
9	Android	2	0,47
10	Tumpangan	1	0,77
11	Kendaraan	1	0,77
12	Google	1	0,77

No	*Term*	DF	IDF
13	Maps	1	0,77
14	GPS	1	0,77

$$idf_{aplikasi} = log_{10}\left(\frac{5+1}{2}\right) = 0{,}47$$

Perhitungan diatas adalah contoh pencarian nilai *idf* pada kata "aplikasi", berdasarkan Persamaan (*2.2)* dan diketahui *N* (total dokumen) adalah 5, maka diperolah hasil *idf* = log(3) yaitu 0,47.

2.1.7 Penghitungan TF x IDF

Untuk mendapatkan bobot akhir dari suatu *term* maka dilakukan pengalian antara TF dan IDF. Contoh penghitungan TF x IDF menggunakan nilai dari TF pada Tabel 2.1 dan IDF pada Tabel 2.2 terdapat pada Tabel 2.3. Berikut Persamaan (*2.3)* untuk menghitung nilai TF IDF.

$$TFIDF_{dt} = TF_{dt}\ x\ IDF_t \tag{2.3}$$

dimana :

t = *term*

d = dokumen

$TF_{t,d}$ = nilai TF dari term *t* pada dokumen *d*

idf_t = nilai IDF dari *term t*

Tabel 2.3 Contoh Menghitung TF x IDF

No	Term	TF_{d1}	TF_{d2}	IDF	$TFIDF_{d1}$	$TFIDF_{d2}$
1	Aplikasi	1	1	0,47	0,47	0,47
2	Monitoring	0	1	0,77	0	0,77
3	Keberadaan	0	1	0,77	0	0,77

No	Term	TF_{d1}	TF_{d2}	IDF	$TFIDF_{d1}$	$TFIDF_{d2}$
4	Objek	0	1	0,77	0	0,77
5	Melalui	0	1	0,77	0	0,77
6	Perangkat	0	1	0,77	0	0,77
7	Berbasis	1	1	0,47	0,47	0,47
8	Bergerak	0	1	0,77	0	0,77
9	Android	1	1	0,47	0,47	0,47
10	Tumpangan	1	0	0,77	0,77	0
11	Kendaraan	1	0	0,77	0,77	0
12	Google	1	0	0,77	0,77	0
13	Maps	1	0	0,77	0,77	0
14	GPS	1	0	0,77	0,77	0

2.2 Algoritma Pembentukan *Parallel Corpus*

Pengumpulan *parallel corpus* menggunakan data Tugas Akhir serta Tesis mahasiswa FTIF, Statistika, Telekomunikasi Multimedia, dan Jaringan Cerdas Multimedia (S2) di Institut Tekologi Sepuluh Nopember, karena judul dokumen tersebut tersedia dalam dua bahasa, yaitu bahasa Indonesia dan bahasa Inggris.

Terdapat 865 corpus berbahasa Indonesia dan 637 corpus berbahasa Inggris yang digunakan. Pengumpulan *parallel corpus* dilakukan dengan menghitung nilai kesamaan antar dokumen menggunakan *Jaccard similarity*. Selanjutnya diambil dokumen bahasa Inggris dengan nilai kesamaan tertinggi dari setiap dokumen bahasa Indonesia, yang selanjutnya pasangan tersebut disebut *parallel corpus*.

Langkah – langkah secara umum pembentukan *parallel corpus*:

1. Melakukan ekstraksi *term* seperti pada subbab 2.1 pada masing- masing dokumen bahasa Indonesia dan bahasa Inggris

2. Melakukan terjemahan kata bahasa Inggris kedalam bahasa Indonesia seperti yang dijelaskan pada subbab 2.2.1 dan contoh hasil terjemahan dapat dilihat pada Tabel 2.4
3. Lakukan penghitungan kemiripan antara dokumen bahasa Indonesia dan juga dokumen bahasa Inggris yang telah diterjemahkan menggunakan *Jaccard similarity* sesuai (*2.4*), contoh perhitungan *Jaccard similarity* terdapat pada Tabel 2.5
4. Akan terdapat *n* (jumlah dokumen bahasa Inggris) pasang pada setiap satu dokumen bahasa Indonesia beserta nilai kemiripan. Lalu ambil nilai kemiripan tertinggi pada setiap dokumen bahasa Indonesia. Sehingga didapatkan semua dokumen bahasa Indonesia memiliki pasangan dokumen bahasa Inggris dengan nilai kemiripan paling tinggi
5. *Parallel corpus* sudah terbentuk dari pasangan dokumen seperti contoh Tabel 2.6

2.2.1 Penterjemahan Kata

Penterjemahan kata bahasa Inggris ke dalam bahasa Indonesia melalui dua cara, yang pertama menggunakan google terjemahan dan yang kedua menggunakan *database* kamus bahasa Inggris-Indonesia.

Pertama, terjemahan google terjemahan menggunakan bantuan pustaka Python *urllib2* untuk mensiasati limit pada API google terjemahan yaitu 1000 kata perhari.*Urllib2* adalah modul Python untuk mengambil URL (*Uniform Resource Locators*). Ini menawarkan antarmuka yang sangat sederhana, dalam bentuk fungsi *urlopen*. Ini mampu mengambil URL menggunakan berbagai protokol yang berbeda. *Urllib2* juga menawarkan antarmuka yang sedikit lebih kompleks untuk menangani situasi umum, seperti otentikasi dasar, *cookies*, *proxy* dan sebagainya.

Urllib2 mendukung pengambilan URL untuk banyak "skema URL" (diidentifikasi oleh string sebelum ":" di URL, misalnya "http" skema URL "http:// translate.google.com") dengan

menggunakan protokol jaringan mereka yang terkait (url: Misalnya FTP, HTTP)[5]. *Urllib2* digunakan untuk mengakses translate.google.com yang akan digunakan untuk melakukan penterjemahan kata bahasa Inggris kedalam bahasa Indonesia.

Kedua, penterjemahan juga dilakukan dengan melakukan pencocokan kata bahasa Inggris kedalam *database* kamus bahasa Inggris–Indonesia, sehingga akan didapatkan beberapa kemungkinan kata terjemahan bahasa Indonesia dari kata bahasa Inggris.

Pengaturan *urllib2* untuk menangani *proxy* menggunakan *urllib2.ProxyHandler* ditunjukan pada Kode Sumber 2.1. Contoh hasil penterjemahan kata ditunjukan pada Tabel 2.4.

```
1.  import urllib2
2.  ....
3.      proxy_support = urllib2.ProxyHandler({})
4.      opener=urllib2.build_opener(proxy_support)
5.      urllib2.install_opener(opener)
6.      agents = {'User-
    Agent':"Mozilla/4.0 (compatible; MSIE 6.0; Windows
     NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.5072
    7; .NET CLR 3.0.04506.30)"}
7.      before_trans = 'class="t0">'
8.      link = "http://translate.google.com/m?hl=%s&sl
    =%s&q=%s" % (to_langage, langage, to_translate.rep
    lace(" ", "+"))
9.      request = urllib2.Request(link, headers=agents
    )
10.     page = urllib2.urlopen(request).read()
11. ....
```

Kode Sumber 2.1 Pengaturan urllib2

Tabel 2.4 Contoh Hasil Terjemahan Kata

Kata	Google translate	$Kamus_1$	$Kamus_2$	$Kamus_3$	$Kamus_4$	$Kamus_5$	$Kamus_6$
android	Android	-	-	-	-	-	-
application	Aplikasi	terap	mohon	-	-	-	-
device	Alat	rencana	-	-	-	-	-
google	Google	-	-	-	-	-	-
gps	Gps	-	-	-	-	-	-
hitchhike	bonceng	Tumpang	nunut	-	-	-	-
maps	peta	-	-	-	-	-	_

Kolom kata merupakan kata dari proses ekstraksi kata dokumen bahasa Inggris, kolom Google translate adalah terjemahan dalam bahasa Indonesia dari kolom kata menggunakan Google terjemahan, kolom $Kamus_1$ – $Kamus_6$ merupakan terjemahan dari kolom kata ke bahasa Indonesia menggunakan *database* kamus.

2.2.2 Jaccard Similarity

Jaccard similarity adalah statistik yang digunakan untuk membandingkan kesamaan sampel set. *Jaccard* koefisien menghitung kemiripan antara set sampel, dan didefinisikan sebagai ukuran potongan (*intersection*) dibagi dengan gabungan (*union*) set sampel. Rumus Penghitungan *Jaccard similarity* ditunjukan pada Persamaan (*2.4)*.

$$sim_{jacc}(t_1, t_2) = \frac{|t_1 \cap t_2|}{|t_1 \cup t_2|} \tag{2.4}$$

Dimana $t_1$dan t_{12} adalah set sampel yang akan dihitung similaritasnya. Sebagai contoh pada kalimat 1 "Hitchhike Application Based on Android Device Using Google Maps and GPS" dan terjemahannya pada contoh sebelumnya. Serta kalimat 2 "Aplikasi Tumpangan Kendaraan Berbasis Android dengan Google Maps dan GPS".

Tabel 2.5 Daftar Perbandingan Kata

Kata(Inggris)	t_1 (tejemahan Inggris)	$t_{1,n}$ (tejemahan Inggris)	t_2 (Indonasia)
android	android	-	Android
application	aplikasi	-	Aplikasi
device	alat	-	Basis
google	google	-	Google
gps	gps	-	Gps
hitchhike	membonceng	tumpang	Tumpeng
maps	peta	-	Maps
	-	-	Kendara

Pada Tabel 2.5 ditunjukan bahwa, $t_{1,1}$,…..,$t_{1,n}$ merupakan kemungkinan terjemahan bahasa Indonesia dari kata bahasa

Inggris, kata asli dalam bahasa Inggris juga ikut dibandingkan karena banyaknya kata berbahasa Inggris dalam judul dokumen bahasa Indonesia, sedangkan t_2 adalah kalimat dalam bahasa Indonesia yang akan di hitung similaritasnya dengan kemungkinan terjemahan bahasa Indonesia dari judul bahasa Inggris.

Pada Tabel 2.5 berdasarkan Persamaan *(2.4)* maka didapatkan $sim_{jacc}(t_1, t_2) = \frac{6}{7+8-6} = 0{,}667$.

Selanjutnya diambil dokumen bahasa Inggris dengan nilai kesamaan tertinggi dari setiap dokumen bahasa Indonesia, yang selanjutnya pasangan tersebut disebut *parallel corpus*. Hasil perhitungan yang dihasilkan dalam melakukan perhitungan dan pengambilan *parallel corpus* yang terbentuk dengan nilai persamaan tertinggi ditujukan pada ditunjukan pada Tabel 2.6.

Tabel 2.6 Pengambilan Similaritas Terbesar

No.	Judul	Title	Jaccard Similarity
1	Aplikasi Tumpangan Kendaraan Berbasis Android dengan Google Maps dan GPS	Hitchhike Application Based on Android Device Using Google Maps and GPS	0,67
2	Aplikasi Tumpangan Kendaraan Berbasis Android dengan Google Maps dan GPS	Monitoring Applications of Object Existence using Android Mobile Devices	0,34
3	Aplikasi Monitoring Keberadaan Objek Melalui Perangkat Bergerak Berbasis Android	Monitoring Applications of Object Existence using Android Mobile Devices	0,78

No.	Judul	Title	Jaccard Similarity
4	Aplikasi Monitoring Keberadaan Objek Melalui Perangkat Bergerak Berbasis Android	Hitchhike Application Based on Android Device Using Google Maps and GPS	0,36

2.3 Pembentukan Informasi *Co-occurrence*

Proses selanjutnya adalah pembentukan informasi *co-occurrence* kata dari *parallel corpus* yang telah terbentuk. Elemen penyusun informasi *co-occurrence* adalah kata yang muncul terlebih dahulu, misalnya kata X, kata *co-occurrence* Y (kata yang selanjutnya muncul) dan *relevance weight* (w) yang dihitung berdasarkan Persamaan (*2.5)* sampai Persamaan (*2.6)* dari kemunculan dua kata tersebut.

Proses pembentukan Informasi *co-occurrence* terdiri dari tiga tahap yaitu *pairing* kata, pembobotan kata, perhitungan *relevance weight*. Proses ini dilakukan hingga dokumen diproses secara keseluruhan dan masing-masing kata pada *parallel corpus* tersebut mempunyai informasi *co-occurrence*.

2.3.1 Pairing Kata

Proses ini membentuk kombinasi semua kata dari hasil pada proses pembentukan *parallel* corpus. Kombinasi dilakukan menggunakan *parallel* corpus yang berarti akan ada kombinasi kata bahasa Indonesia dan bahasa Inggris. Kombinasi kata-kata ini nantinya akan digunakan untuk membentuk informasi *co-occurrence*. Contoh pembentukan *pairing* kata ditunjukan pada Gambar 2.1

Gambar 2.1 Contoh Pairing Kata

2.3.2 Pembobotan Pairing Term

Tahap selanjutnya hasil pembentukan *pairing term* dari proses sebelumnya akan dihitung nilai bobot tiap pasangan kata yang nantinya akan digunakan untuk perhitungan *relevance weight*. Perhitungan bobot yang digunakan *inverse document frequency*, *tf-idf* . Perhitungan pada tahap ini menggunakan pasangan. Rumus perhitungan bobot *pairing term* terdapat pada Persamaan (*2.5)*, dan contoh hasil perhitungan ditampilkan pada Tabel 2.8 kolom *idf*.

$$d_{kij} = tf_{kij} \times \log\left(\frac{N}{df_{ij}}\right) \tag{2.5}$$

dimana :

tf_{kij} = *term frequency* minimal dari *term i* dan *term frequency term j* pada pasangan dokumen *k*

df_{ij} = *document frequency* dari *term j* dan *term k*.

2.3.3 Relevance Weight

Relevance Weight digunakan untuk mengukur kedekatan antar kata dalam pasangan kata yang dibentuk. Setiap *pair* (T_j, T_k)

akan dihitung *relevance weight*-nya dengan hasil bagi antara penjumlahan TF-IDF dari *pair* T_j, T_k (d_{ijk}) pada dokumen ke-*i* sampai dokumen ke-*n* (jumlah dokumen dalam *parallel corpus*) dan penjumlahan TF-IDF dari *term* ke-*j* (pertama) pada dokumen ke-*i* sampai dokumen ke-*n* (d_{ij}) dikalikan dengan *weighting factor* dari *term k* (kedua) [6]

Untuk menghitung *weighting factor* dengan cara pembagian antara log jumlah dokumen dalam *corpus* (*N*) dibagi dokumen frequensi dari *term k* (df_k) dan log jumlah dokumen dalam *corpus* (*N*). Perhitungan *weighting factor* sesuai Persamaan (*2.6)*, contoh hasil perhitungan ditampilkan pada Tabel 2.7 kolom weighting_factor.

$$WeightingFactor\,(Tk) = \frac{\log \frac{N}{\mathrm{dfk}}}{\log N} \tag{2.6}$$

Setelah itu lakukan dilakukan penghitungan *relevance weights* antara *term j* dan *term k* [5]. Perhitungan *relevance weight* sesuai Persamaan

(*2.7)*, contoh hasil perhitungan ditampilkan pada Tabel 2.8 kolom relevance _weight.

$$RelevanceWeight(Tj, Tk) = \frac{\sum_{i=1}^{n} dijk}{\sum_{i=1}^{n} dij} xWeightingFactor(Tk) \tag{2.7}$$

Tabel 2.7 Hasil Perhitugan Pembobotan dan Weighting Factor

term	tf	df	idf	tfidf	weighting_factor
android	2	1	0,69	1,39	0,43
aplikasi	1	1	0,69	0,69	0,43
Application	1	1	0,69	0,69	0,43
Bas	1	1	0,69	0,69	0,43
device	1	1	0,69	0,69	0,43

term	tf	df	idf	tfidf	weighting_factor
google	2	1	0,69	1,39	0,43
gps	2	1	0,69	1,39	0,43
hitchhike	1	1	0,69	0,69	0,43
kendara	1	1	0,69	0,69	0,43
maps	2	1	0,69	1,39	0,43
tumpang	1	1	0,69	0,69	0,43

Hasil inilah yang akan dijadikan bobot pembentukan tesaurus sebuah kata. Semakin tinggi nilai *relevance weight* maka semakin mirip kata tersebut.

Tabel 2.8 Perhitungan Relevance Weight

No	term1	term2	tf	idf	relevance _weight
1	android	aplikasi	2	0,47	0,195
2	android	application	1	0,77	0.94
3	android	basis	2	0,47	0,52
4	android	device	1	0,77	0,52
5	android	google	1	0,77	0,52
6	android	gps	1	0,77	0,52
7	android	hitchhike	1	0,77	0,52
8	android	kendara	1	0,77	0,52
9	android	maps	1	0,77	0,52
10	android	tumpang	1	0,77	0,52
11	aplikasi	application	1	0,77	1, 08
12	aplikasi	bas	1	0,77	1, 08
13	aplikasi	device	1	0,77	1, 08
14	aplikasi	google	1	0,77	1, 08

No	term1	term2	tf	idf	relevance _weight
15	aplikasi	gps	1	0,77	1, 08
16	aplikasi	hitchhike	1	0,77	1, 08
17	aplikasi	kendara	1	0,77	1, 08

$$idk_{android,aplikasi} = log_{10}\left(\frac{5+1}{2}\right) = 0{,}47$$

Perhitungan diatas adalah contoh pencarian nilai *idf* pada kata "aplikasi", berdasarkan Persamaan (*2.5)* dan diketahui *N* (total dokumen) adalah 5, maka diperolah hasil idf = log(3) yaitu 0,47.

$$WeightingFactor\,(android) = \frac{0.69897}{log\,(5)} = \frac{0.69897}{1{,}6} = 0{,}43$$

$$W_{android,aplikasi} = \frac{0{,}47}{0{,}69} \times 0{,}4 = 0{,}195$$

Pada contoh perhitungan diatas, nilai *weighting factor* diperoleh dari Persamaan (*2.6),* nilai *weighting factor* pada android adalah nilai *idf* android pada Tabel 2.7 dibagi dengan log(jumlah pair dokumen) pada contoh, diketahui jumlah pair dokumen adalah 5 sehingga nilai *weighting factor* dari kata android adalah 0,4.

Sedangkan perhitungan *relevance weight* diperoleh dari Persamaan (*2.7),relevance weight* pada pasangan kata *term* 1 yaitu android dan *term* 2 yaitu aplikasi adalah nilai *idf* pada Tabel 2.8 dibagi nilai *idf* dari term 2 (aplikasi) pada Tabel 2.7 dikalikan nilai *weighting_factor* dari *term* 1 (android) sehingga nilai relevance weight pada pasangan term android dan aplikasi adalah 0,26.

2.4 Algoritma Constraint Satisfaction Problem

Constraint Satisfaction Problem merupakan sebuah pendekatan untuk menyelesaikan suatu masalah dengan tujuan menemukan keadaan atau objek yang memenuhi sejumlah persyaratan atau kriteria. *Constraint Satisfaction Problem* adalah

suatu permasalahan yang mencari nilai untuk set variabel (*finite*) yang memenuhi set *constraint*.

Constraint Satisfaction Problem (CSP) dimulai dari solusi kosong dan diakhiri dengan sebuah solusi yang memenuhi semua *constraint* (*consistent*). Pencarian solusi dilakukan dengan mencoba mengisi nilai domain pada setiap variabel satu demi satu tanpa melanggar *constraint*, sampai solusi ditemukan [6].

Constraint Satisfaction Problem memiliki komponen yaitu:

- Variabel
  Set *term co-occurence*
- Domain
  Term yang memenuhi *constraint*
- *Constraint*
  - *Node consistency*

x_j konsisten jika dan hanya jika c_j sesuai pada *associate constraint network*.

$$c_j = \begin{cases} 1 \; if \; \sum_{i=0}^{n-1} w_{i,j} x_i \geq threshold \\ 0, if \; \sum_{i=0}^{n-1} w_{i,j} x_i < threshold \end{cases}$$

  - *Associate constraint network*

Associate constraint network sesuai jika dan hanya jika semua node dalam *associate constraint network* konsisten dan $\sum_j x_j < C$, dimana C adalah *threshold* dan ditentukan secara statistik berdasarkan distribusi konsep masukan dan asosiasi terkait. C adalah *threshold* yang digunakan untuk membatasi jumlah keluaran *term* tesaurus

Solusi untuk *Constraint Satisfaction Problem* dapat ditemukan secara efisien dan efektif dengan mencari secara sistematis melalui algoritma pencarian. Sebuah pendekatan alternatif adalah *backtracking*. *Backtracking* adalah algoritma yang berbasis pada DFS untuk mencari solusi persoalan secara lebih ringkas. *Backtracking* merupakan perbaikan dari algoritma *brute-force*, secara sistematis mencari solusi persoalan di antara

semua kemungkinan solusi yang ada [7]. Dengan metode *backtracking*, kita tidak perlu memeriksa semua kemungkinan solusi yang ada. Hanya pencarian yang mengarah ke solusi saja yang dipertimbangkan. Sehingga waktu pencarian dapat dihemat.

Algoritma Depth First Search (DFS) adalah salah satu algoritma pencarian solusi yang digunakan di dalam kecerdasan buatan. Algoritma ini termasuk salah satu algoritma yang melakukan pencarian dalam urutan tertentu tetapi tidak memiliki informasi apa-apa sebagai dasar pencarian kecuali hanya mengikuti pola yang diberikan. Di dalam DFS, pencarian dilakukan pada suatu struktur pohon yaitu kumpulan semua kondisi yang mungkin yang diimplementasikan dalam sebuah struktur pohon. Paling atas adalah akar (root) yang berisi kondisi awal pencarian (initial state) dan di bawahnya adalah kondisi-kondisi berikutnya sampai kepada kondisi tujuan (goal state). Gambar 2.2 merupakan algoritma *backtracking*. Pada Tugas akhir ini *backtracking* hanya terbentuk hanya jika dilakukan pencarian kata.

Algoritma *backtracking*:

1. Inisialisasi. Nilai dari semua node di inisialisasikan sebagai 0.
2. Cari kemungkinan solusi.

$$v_j = \sum_{i=0,\, i \neq j}^{n} W_{ij}\, x_i$$

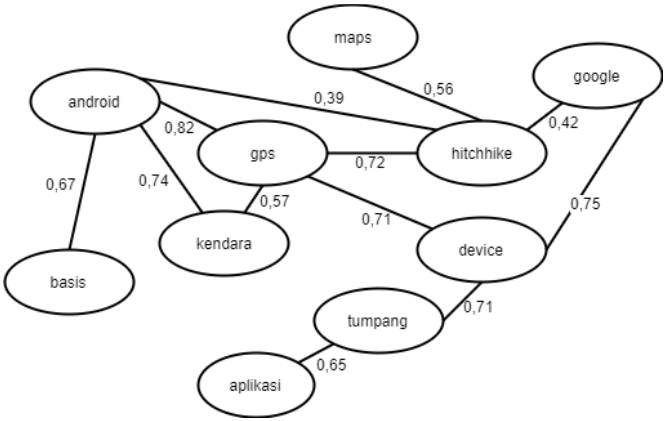
W_{ij} : *relevance weight* antara node *i* ke node

3. Menentukan jika solusi ditemukan
   Periksa *node consistency* pada semua node.
   IF semua *node consistent*
      Masuk pada himpunan solusi
   ELSE IF $\sum_j x_j < C$
      THEN solusi ditemukan
         ELSE // solusi *infeasible* ditemukan
         x(t) = 0 // *x(t)* penyebab *infeasible solution*
            Kembali ke langkah 2 // *backtracking*
   ELSE // iterasi selanjutnya
      Kembali ke langkah 2.

Gambar 2.2 Psudocode CSP Menggunakan Backtracking

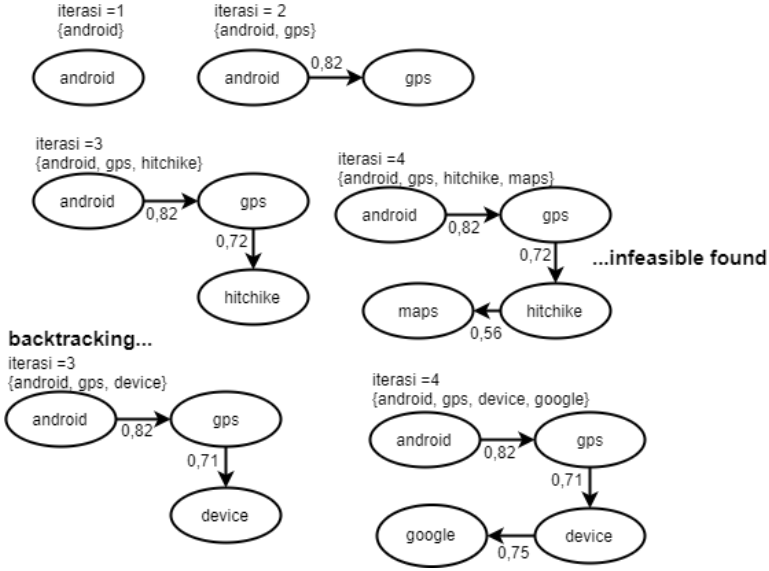
Tabel 2.9 Relevance Weight pada Pairing Term

No	w	Term1	Term2	Relevance Weight
1	$w_{1,2}$	android	gps	0,82
2	$w_{1,7}$	android	kendara	0,74
3	$w_{1,8}$	android	basis	0,67
4	$w_{2,3}$	gps	hitchihke	0,72
5	$w_{2,4}$	gps	device	0,71
6	$w_{2,7}$	gps	kendara	0,57
7	$w_{3,4}$	hitchihke	maps	0,56
8	$w_{3,6}$	hitchihke	google	0,42
9	$w_{3,1}$	hitchihke	android	0,39
10	$w_{5,6}$	device	google	0,75
11	$w_{5,9}$	device	tumpang	0,71
12	$w_{5,10}$	device	aplikasi	0,65



Gambar 2.3 Contoh Graf yang Terbentuk dari Ekstraksi

Gambar 2.3 merupakan graf yang terbentuk dari proses ekstraksi kata dan pembentukan *co-ocurrence* serta *relevance weight* yang terdapat pada Tabel 2.9 node yang terbentuk memiliki bobot yang merupakan *relevance weight*. Dari Gambar 2.3 kemudian dilakukan pencarian *constraint satisfaction problem* menggunakan *backtracking*.



Gambar 2.4 Contoh Backtracking

Misalnya, seperti yang diilustrasikan pada Gambar 2.4. Diketahui x_1 = android, x_2 = gps, x_3 = hitchhike, x_4 = maps, x_5 = device, x_6 = google. x_1 akan menjadi istilah masukan dan dimasukan pada himpunan solusi. *Relevance weight* adalah nilai kesamaan antara *term*1 dan *term*2. Nilai *relevance weight* pada pasangan kata dapat dilihat pada Tabel 2.9. Misal, nilai *threshold* untuk jumlah *relevance weight* = 2,25 dan *threshold* C (jumlah kata yang terbentuk) = 5.

Pada Langkah 2, x_2 dimasukan ke himpunan solusi dan y_1 = 2 karena $w_{1,2}$ adalah yang terbesar. {android, gps} adalah solusi parsial. Solusi belum memenuhi *constraint satisfaction* tetapi

solusi saat ini adalah *feasible* solusi parsial. Algoritma kembali ke Langkah 2. Pada iterasi ini, x_3 adalah dimasukan dalam himpunan solusi, dan $y_2 = 2$ karena $v_3 = w_{1,2} + w_{2,3}$ adalah yang terbesar. Solusi parsial sekarang menjadi {android, gps} *feasible* namun tidak memenuhi *constraint satisfaction*. Solusi parsial dibeberapa iterasi berikutnya adalah {android, gps, hitchhike}, {android, gps, hitchhike, maps}, dan akhirnya solusi *infeasible* ditemukan.

Selanjutnya dilakukan *backtracking* mundur ke solusi parsial {android, gps, hitchhike}. Karena solusi parsial yang dihasilkan dari {android, gps, hitchhike} tidak layak, kita backtrack ke {android, gps} dan mendapatkan {android, gps, device} Dalam satu iterasi selanjutnya, solusi ditemukan sebagai {android, gps, device, google}. Sebagai ilustrasi, algoritma backtracking terus mencari solusi sampai suatu *infeasible* solusi ditemukan. Kemudian menghapus node yang menyebabkan *infeasibility* dan melakukan *backtrack* solusi parsial sebelumnya.

Masalah dan Solusi dalam pembentukan CSP:

1. Jika user memasukan *threshold* yang terlalu besar dan C yang terlalu kecil. Maka hal tersebut tidak akan memenuhi Constraint Satisfaction
2. Untuk itu pengulangan dilakukan hanya sebanyak (C/2) +1
3. Dalam setiap perulangan himpunan kata disimpan untuk dibandingkan apabila perulangan sudah terlalu banyak
4. Selanjutnya himpunan penyelesaian itu akan dibandingkan dan diambil himpunan kata yang memiliki nilai tertingi dan diambil sebagai solusi

2.5 Flask

Suatu *web application framework* yang terdapat pada bahasa pemrograman Python. *Flask* digunakan sebagai *server* yang melakukan perhitungan *backtracking*. Flask menghubungkan Python dengan PHP, sehingga Flask dapat memparsing masukan dari PHP dan mengirimkannya ke fungsi Python, sehingga Flask dapat melakukan perhitungan fungsi Python tersebut berdasarkan

masukan melalui PHP. Eksperimen ini memilih *Flask*, sebab *Flask* memiliki tingkat fleksibel yang sangat tinggi untuk digunakan bersamaan dengan *library* Python pada umumnya. *Flask* memiliki *routes* yang dapat dipanggil. Ketika *routes* dalam *Flask* dipanggil, maka *Flask* akan melakukan kompilasi sesuai dengan *routes* yang bersangkutan.

```
1. @app.route("/backtrack/")
2. def backtrack():
3. //code
```

Kode Sumber 2.2 Potongan dari Flask

Kode Sumber 2.1 merupakan penggalan kode dari aplikasi *Flask*. Pada baris 1 dapat diamati bahwa penggalan kode tersebut memiliki *routes* "*/backtrack/*". Sehingga ketika "*/backtrack/*" dijalankan akan menjalankan fungsi "*backtrack()*" untuk melakukan kompilasi.

2.6 Sigma JavaScript Graph Drawing Library

Sigma JavaScript Graph Drawing Library merupakan *JavaScript Library* yang khusus digunakan untuk mevisualisasikan graf ke dalam sebuah aplikasi berbasis web. *SigmaJs* dikembangkan oleh Alexis Jacomy dengan bantuan Guillaume Plique [8]. Beberapa keunggulannya diantara lain pengembang diberikan pilihan dalam proses *rendering* (proses akhir pembangunan) untuk memodelkan graf yang divisualisasikan, yaitu dapat berupa *Canvas* biasa ataupun WebGL (*Web Graphics Library*) sesuai dukungan yang disediakan *browser*. Kemudian keunggulan lainnya yaitu mendukung untuk menerima masukan dari mouse, *rescaling*, dan *refreshing*. Beberapa hal tersebut akan membuat graf yang divisualisasikan berjalan lancar serta memiliki

performa yang baik ketika dijalankan. *SigmaJs* memberikan kemudahan cara memvisualisasikan graf dibuat, data graf dapat berupa DOM (*Document Object Model*), bisa juga diambil dari *file* terpisah berupa JSON (*JavaScript Object Notation*) atau GEXF (*Graph Exchange* XML *Format*). Untuk keperluan tertentu dapat juga ditambahkan API (*Application Programming Interface*) publik untuk memodifikasi data, mengubah posisi kamera, *refresh rendering* serta menerima event tertentu. Visualisasi *basic* pada SigmaJs bisa dilihat pada Gambar 2.5. Pada Kode Sumber 8.1 merupakan view dari *framework* codeigniter untuk membentuk graf menggunakan sigma.js, dan Kode Sumber 8.2 merupakan controller dari framework codeigniter untuk membentuk graf menggunakan sigma.js.

Gambar 2.5 Sigma.js

2.7 Multiprocessing

Multiprocessing adalah pustaka Python yang mendukung proses pemisahan proses menggunakan API yang mirip dengan modul *threading*. Pustaka *multiprocessing* menawarkan *concurrency* lokal dan *remote*, secara efektif mengimbangi *Global Interpreter Lock* dengan menggunakan subproses. Karena ini, modul multiprocessing memungkinkan pemrogram untuk sepenuhnya memanfaatkan beberapa prosesor pada mesin tertentu. Ini berjalan pada Unix dan Windows.
Modul *multiprocessing* juga mengenalkan API yang tidak memiliki analog pada modul *threading*. Kode Sumber 2.3 menunjukkan praktik umum untuk menentukan fungsi semacam itu dalam modul sehingga proses *child* berhasil mengimpor modul itu. Kode Sumber 2.3 adalah contoh penggumaan *multiprocessing* menggunakan pustaka Python.

`start()` merupakan fungsi untuk memulai aktivitas proses. `join()` merupakan fungsi untuk menggabungkan proses. Sebuah proses tidak bisa bergabung sendiri karena ini akan menyebabkan *deadlock*.

```
1.  from multiprocessing import Process
2.  ....
3.  paired = list(sorted(cooc))
4.      count = len(paired)/numprocess
5.      bawah = 0
6.      minn = 0
7.      maxx = len(paired)
8.      jobs=[]
9.      for i in range(numprocess):
10.         atas = bawah+count
11.         if i>=numprocess-1:
12.             atas = maxx
13.
14.         p = Process(target=commiter, args=(i, bawah
    , paired[bawah:atas], len(paired), ))
15.         jobs.append(p)
```

```
16.         p.start()
17.         bawah = atas
18.
19.     for job in jobs:
20.         job.join()
21. ....
```

Kode Sumber 2.3 Contoh *Multiprocessing*

(Halaman ini sengaja dikosongkan)

BAB III
ANALISIS PERANCANGAN SISTEM

Bab ini membahas tahap analisis dan perancangan sistem yang akan dibangun. Analisis membahas semua persiapan yang akan menjadi pokok pikiran pembuatan aplikasi ini. Mulai dari masalah yang melatar belakangi, hingga analisis gambaran awal sistem yang akan dibuat. Perancangan sistem membahas hal-hal yang berkaitan dengan pondasi atau dasar pembuatan aplikasi, yang meliputi perancangan basis data, tampilan antar muka halaman aplikasi, hingga perancangan alur proses yang akan diimplementasikan di dalam aplikasi.

3.1 Analisis Sistem

Tahap analisis meliputi analisis masalah, deskripsi umum sistem, dan kasus penggunaan sistem yang dibuat.

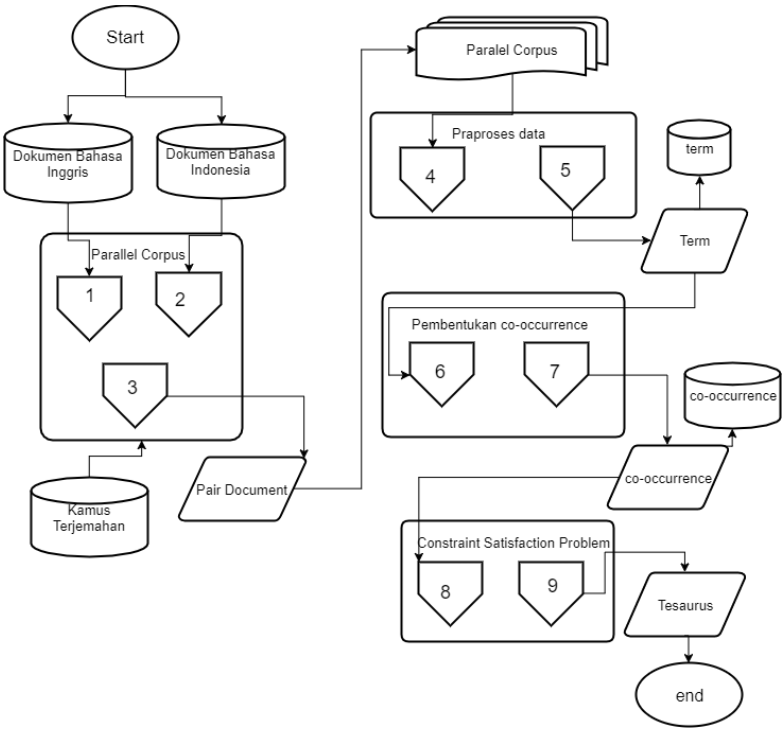
3.1.1 Analisis Permasalahan

Permasalahan yang diangkat pada tugas akhir ini adalah dalam pencarian, setiap mahasiswa memiliki kecenderungan mencari dokumen dengan menggunakan kata kunci dengan bahasa tertentu.

Permasalahan tersebut dapat diatasi dengan membangun *Cross-lingual* tesaurus memungkinkan pengguna mendapatkan dokumen yang relevan dengan input bahasa yang berbeda dengan bahasa pada dokumen.

3.1.2 Deskripsi Umum Sistem

Sistem atau *Cross-lingual* tesaurus yang dibangun pada tugas akhir ini terbagi menjadi empat proses utama, yaitu praproses data, *document alignment*, pembentukan *co-occurrence* dan *constraint satisfaction problem*[6]. Gambar 3.1 adalah bentuk dari gambaran umum sistem.



Gambar 3.1 Deskripsi Sistem

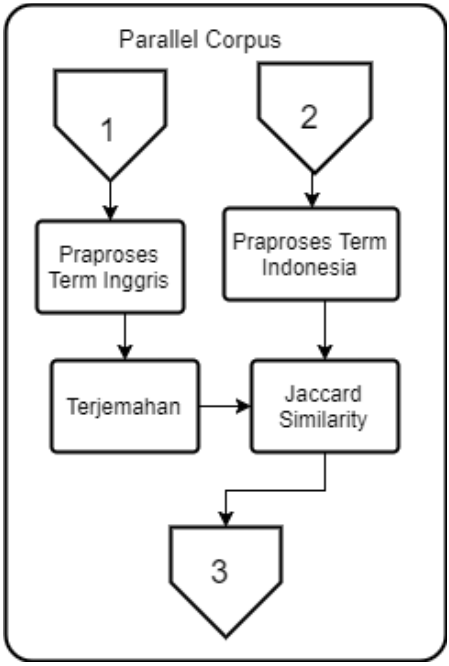
Pada Gambar 3.1 di atas, akan dijelaskan dengan rinci mulai dari *Document Alignment*, praproses data, pembentukan informasi *co-occurrence*, dan *Constraint Satisfaction Problem*. Detail runtutan aplikasi dari *input* hingga *output* akan dijelaskan pada subbab berikut:

3.2 Perancangan Sistem

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

3.2.1 Pembentukan Parallel Corpus

Dalam melakukan pembentukan *parallel corpus* pada data Tugas Akhir dan Tesis ada beberapa metode yang harus dilakukan. Alur dari pembentukan *parallel corpus* bisa dilihat pada Gambar 3.2.



Gambar 3.2 Alur Pembentukan Parallel Corpus

3.2.1.1. Terjemahan Kata

Langkah-langkah yang dilakukan untuk melakukan *document alignment* dari setiap judul dokumen bahasa Indonesia dan bahasa Inggris dilakukan sebagai berikut:

1. Impor pustaka python untuk menghubungkan python dengan basis data MYSQL, mysql.connector.
2. Impor pustaka urllib2 untuk mengakses URL translete.google.com

3. Lakukan praproses data pada dokumen seperti yang telah dijelaskan pada subbab 3.2.1 pada dokumen bahasa Inggris.
4. Simpan pada *database*.
5. Lakukan terjemahan kata bahasa Inggris kedalam bahasa Indonesia menggunakan API google *translate*
6. Selanjutnya dari term bahasa Inggris dilakukan pembandingan arti dari *database* gkamus bahasa Inggris ke bahasa Indonesia.
7. Simpan ke *database*.

Contoh kata "classification of nutritional status in children aged 6-12 years in indonesia using ordinal logistic regression and support vector mechine (svm)" yang telah dilakukan praproses dan terjemahan akan menjadi

$$Doc_1 = \begin{bmatrix} "classification, klasifikasi" \\ "nutritional, nutrisi" \\ "status, pangkat" \\ "\text{children, anak}" \\ "aged, umur" \\ "ordinal, urut" \\ "\text{logistic, logistik}" \\ "support, dukung" \\ "vector\ , vektor" \\ "mechine, mesin" \\ "svm, svm" \end{bmatrix}$$

3.2.1.2. Jaccard Similarity

Jaccard similarity adalah statistik yang digunakan untuk membandingkan kesamaan sampel set. Rumus Penghitungan *Jaccard similarity* terdapat pada Persamaan (*3.1)*. Dimana doc_1 adalah term dari kalimat 1 (bahasa inggris) dan juga kemungkinan terjemahan kata dalam bahasa Indonesia, doc_2 adalah term dari

kalimat 2 (bahasa Indonesia). Sedangkan $t_1 \cup t_2$ adalah jumlah term dalam kalimat 1 (yang asli) ditambah dengan jumlah term pada kalimat 2 dikurangi dengan $doc_1 \cap doc_2$

$$sim_{jacc}(doc_1, doc_2) = \frac{|doc_1 \cap t_2|}{\lfloor t_1 \cup t_2 \rfloor} \tag{3.1}$$

Sebagai contoh pada kalimat 1 "Classification of Nutritional Status in Children Aged 6-12 Years in Indonesia using Ordinal Logistic Regression and Support Vector Mechine (SVM)" dan terjemahannya pada contoh sebelumnya. Serta kalimat 2 "Klasifikasi Status Gizi pada Anak Usia 6-12 Tahun di Indonesia dengan Menggunakan Regresi Logistik Ordinal dan Support Vector Mechine (SVM)".

didapatkan $sim_{jacc}(Doc_1, Doc_2) = \frac{10}{13+14-10} = 0{,}58$

$$t_1 = \begin{bmatrix} "classification" \\ "nutritional" \\ "status" \\ "children" \\ "aged" \\ "years" \\ "ordinal" \\ "logistic" \\ "regression" \\ "support" \\ "vector" \\ "mechine" \\ "svm" \end{bmatrix}$$

$$Doc_1 = \begin{bmatrix} "classification" \\ "klasifikasi" \\ "nutritional" \\ "nutrisi" \\ "status" \\ "pangkat" \\ "children" \\ "anak" \\ "aged" \\ "umur" \\ "years" \\ "tahun" \\ "ordinal" \\ "urut" \\ "logistic" \\ "logistik" \\ "support" \\ "dukung" \\ "vector" \\ "vektor" \\ "mechine" \\ "mesin" \\ "svm" \end{bmatrix}$$

$$t_2 = \begin{bmatrix} "klasifikasi" \\ "status" \\ "gizi" \\ "anak" \\ "usia" \\ "tahun" \\ "guna" \\ "regresi" \\ "logistik" \\ "ordinal" \\ "support" \\ "vector" \\ "mechine" \\ "svm" \end{bmatrix}$$

Algoritma untuk menghitung *jaccard similarity* sebagai berikut :

Input : semua dokumen D yang telah dilakukan tokenisasi, remove puctuation, remove stopword, stemming.

Output : nilai kemiripan antar dokumen bahasa Inggris (dengan terjemahan bahasa Indonesia) dan dokumen bahasa Indonesia.

Pseudocode :

a) Impor pustaka Python untuk menghubungkan Python dengan basis data MYSQL, mysql.connector
a) Inisiasi array dengan isi *query* mendapatkan semua *term*
b) Semua item1 dalam variabel bahasa Indonesia
c) Semua item2 dalam variabel bahasa Inggris
d) Jaccard(item1,item2) = intersection(item1,item2) / union(item1,item2)
e) Iterasi sampai masing masing data (b) memiliki pasangan pada data (c)
f) Jika semua item pada perulangan (e) sudah diiterasi ambil nilai maksimal pada masing – masing pasangan.
g) Simpan hasil *Jaccard* dan pasangan item ke dalam basis data.

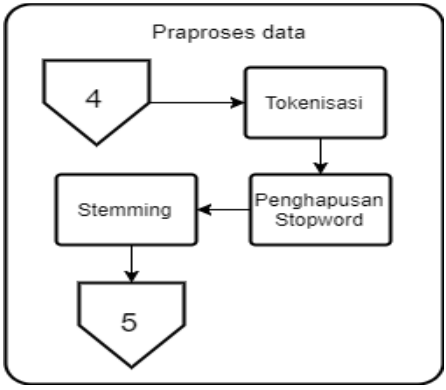
Sehingga dari pencarian pasangan dokumen menggunakan Jaccard akan dihasilkan nilai sebagai berikut :

$$\boldsymbol{PairDoc} = \begin{bmatrix} Doc1, Doc2, 0.667 \\ Doc3, Doc4, 0{,}5 \end{bmatrix}$$

$$\boldsymbol{Parallel\ Corpus} = [Doc1, Doc2]$$

3.2.2 Praproses Data

Praproses data adalah proses awal yang dilakukan untuk melakukan ekstraksi kata-kata dari kumpulan dokumen yang ada dan membentuk informasi *co-occurrence*. Gambar 3.3 merupakan alur praproses data.



Gambar 3.3 Alur Praproses Data

3.2.2.1. Tokenisasi

Proses tokenisasi dilakukan pada setiap judul dokumen. Tokenisasi adalah suatu proses untuk membagi suatu teks berupa kalimat atau paragraf menjadi unit-unit kecil berupa kumpulan kata atau token [4].

Sebelum melakukan tokenisasi, setiap kalimat dilakukan *case folding* dengan *convert* setiap kalimat ke *lowercase* sehingga tidak ada huruf kapital kalimat. *Case Folding* dilakukan dengan menggunakan fungsi pada bahasa pemrograman Python, yaitu ***.lower()***. Tokenisasi dilakukan dengan menggunakan fungsi pada bahasa pemrograman Python, yaitu ***.split()***. Fungsi ***.split()*** digunakan untuk memisahkan kata pada suatu kalimat berdasarkan tanda baca dan spasi.

Sehingga kalimat 1 "Klasifikasi Status Gizi pada Anak Usia 6-12 Tahun di Indonesia dengan menggunakan Regresi Logistik Ordinal Dan Support Vector Mechine (SVM)" apabila diterapkan metode tokenisasi menjadi token.

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

$$Doc_1 = \begin{bmatrix} "Klasifikasi " \\ "Status " \\ "Gizi " \\ "pada " \\ "Anak " \\ "Usia " \\ "6" \\ "12" \\ "Tahun " \\ "di " \\ "Indonesia" \\ "dengan" \\ "menggunakan" \\ "Regresi" \\ "Logistik" \\ "Ordinal" \\ "Dan" \\ "Support " \\ "Vector" \\ "Mechine" \\ "SVM" \end{bmatrix}$$

3.2.2.2. Penghapusan *Stopword*

Proses selanjutnya adalah melakukan penghapusan *stopword* yang terdapat pada kumpulan kata-kata hasil tokenisasi. *Stopword* adalah kumpulan kata trivial atau kata yang sering muncul. Penghapusan *Stopword* dilakukan dengan menggunakan *library* pada bahasa pemrograman Python, yaitu *Natural Language Toolkit (nltk)*. Pada *library* nltk terdapat dokumen *stopword* dalam beberapa bahasa, namun belum tersedia dokumen *stopword* dalam bahasa Indonesia, sehingga harus dilakukan penambahan secara manual dokumen *stopword* bahasa Indonesia. Dokumen kumpulan *stopword* Bahasa Indonesia didapatkan dari penelitian yang

dilakukan oleh F. Z. Tala [9] tentang efek *stemming* untuk temu kembali informasi pada bahasa Indonesia.

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 dokumen.

Sebagai contoh dalam token kata diatas kata-kata yang masuk kedalam list stop word ada 3 yaitu kata "pada", "di" dan "dan". Sehingga token kata menjadi:

$$Doc_2 = \begin{bmatrix} "klasifikasi" \\ "status" \\ "gizi" \\ "anak" \\ "usia" \\ "tahun" \\ "menggunakan" \\ "regresi" \\ "logistik" \\ "ordinal" \\ "support" \\ "vector" \\ "mechine" \\ "svm" \end{bmatrix}$$

3.2.2.3. Stemming

Stemming adalah proses untuk mengembalikan bentuk dari suatu kata pada bentuk dasar kata tersebut [9]. Proses *stemming* pada dokumen Bahasa Indonesia dilakukan dengan menggunakan library Python sastrawi, sedangkan pada dokumen Bahasa Inggris digunakan library Python nltk dengan menunduh *package* punkt pada nltk.download(). Sehingga apabila contoh kasus diatas diterapkan metode

$$Doc_2 = \begin{bmatrix} "klasifikasi" \\ "status" \\ "gizi" \\ "anak" \\ "usia" \\ "tahun" \\ "guna" \\ "regresi" \\ "logistik" \\ "ordinal" \\ "support" \\ "vector" \\ "mechine" \\ "svm" \end{bmatrix}$$

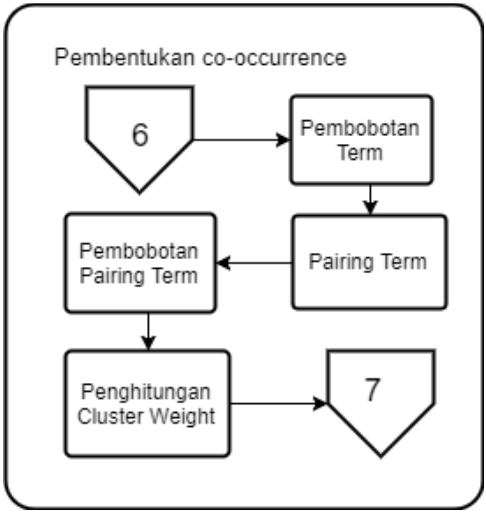
3.2.3 Pembentukan Informasi *Co-occurrence*

Proses ini membentuk kombinasi kata-kata dari hasil praproses data dari pasangan dokumen pada proses sebelumnya. Kombinasi kata-kata ini nantinya akan digunakan untuk membentuk informasi *co-occurrence* dengan nilai *relevance weight*. Gambar 3.4 merupakan alur pembentukan *co-occurrence*.

Berikut langkah-langkah yang dilakukan untuk Pembentukan Informasi *co-occurence*:

a) Impor pustaka python untuk menghuungkan python dengan basis data MYSQL, mysql.connector.
b) Inisiasi array kata dengan index *parallel corpus* dan isi dengan semua token yang telah di *stemming*.
c) Ulangi untuk semua item dalam variabel
d) Hitung tf_{ij} pada masing masing *term* j
e) $d_{ij=} tf_{ij} \times \log(N/df_j)$
f) Lakukan pairing term, sehingga terbentuk term j, term k
g) $d_{kij} = tf_{kij} \times \log(N/df_{ij})$
h) $WF(Tk) = \log(N/df_k) / log\ (N)$
i) $ClusterWeight(Tj, Tk) = d_{ij}/d_{kij} \times WF(Tk)$

j) Simpan hasil *Relevance Weight* ke dalam basis data.



Gambar 3.4 Alur Pembentukan Co-occurrence

$$\boldsymbol{pairkata} = \begin{bmatrix} klasifikasi\,,clasification \\ status,gizi \\ anak\,,umur \\ children,anak \\ . \\ . \\ . \\ . \end{bmatrix}$$

$$\boldsymbol{RelevanceWeight} = \begin{bmatrix} klasifikasi, clasification, 0.22 \\ status, gizi, 1.00 \\ anak, umur, 1.57 \\ children, anak, 1.57 \\ . \\ . \\ . \\ . \end{bmatrix}$$

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

3.2.4 Constraint Satisfaction Problem

Constraint Satisfaction Problem(CSP) dimulai dari solusi kosong dan diakhiri dengan sebuah solusi yang memenuhi semua *constraint*. Pencarian solusi dilakukan dengan mencoba mengisi nilai domain pada setiap variabel satu demi satu tanpa melanggar *constraint*, sampai solusi ditemukan.

- Variabel
  Set *term co-occurence*
- Domain
  Term yang memenuhi *constraint*
- *Constraint*
  - *Node consistency*
    x_j konsisten jika dan hanya jika c_j sesuai pada *associate constraint network*.

$$c_j = \begin{cases} 1 \ if \ \sum_{i=0}^{n-1} w_{i,j} x_i \geq threshold \\ 0, \ if \ \sum_{i=0}^{n-1} w_{i,j} x_i < threshold \end{cases}$$

  - *Associate constraint network*
    Associate constraint network sesuai jika dan hanya jika semua node dalam *associate constraint network* konsisten dan $\sum_j x_j < C$, dimana C adalah *threshold*

dan ditentukan secara statistik berdasarkan distribusi konsep masukan dan asosiasi terkait. *C* adalah *threshold* yang digunakan untuk membatasi jumlah keluaran *term* tesaurus

Solusi untuk CSP dapat ditemukan secara efisien dan efektif dengan mencari secara sistematis melalui algoritma pencarian. Sebuah pendekatan alternatif adalah *backtracking*. *Backtracking* merupakan perbaikan dari algoritma *brute-force*, secara sistematis mencari solusi persoalan di antara semua kemungkinan solusi yang ada.

Berikut langkah – langkah untuk *Constraint satisfaction Problem*

1. Inisialisasi. Nilai dari semua node di inisialisasikan sebagai 0.
2. Cari kemungkinan solusi.

$$v_j = \sum_{i=0, i \neq j}^{n} W_{ij} x_i$$

W_{ij} : *Relevance weight* antara node *i* ke node *j*

3. Menentukan jika solusi ditemukan
   Periksa *node consistency* pada semua node.
   IF semua *node consistent*
   Masuk pada himpunan solusi
   IF $\sum_j x_j < C$
   THEN solusi ditemukan
   ELSE // *infeasible solution* ditemukan
   $X_k(t)$ = 0 //karena x(t) *infeasible solution*
   Kembali ke proses 2 // *backtracking* dibutuhkan
   ELSE // iterasi selanjutnya
   t=t+ 1
   S=himpunanSolusi
   Kembali ke langkah 2.

Contoh kasus penyelesaian *constraint satisfaction problem*:

Input = [gizi]
Threshold C (banyak kata yang terbentuk) = [4]
Threshold (jumlah *relevance weight*) = [4,5]

Cek apakah $\sum_j x_j < 4$
Iterasi 1
HimpunanSolusi = [nutritional]
$\boldsymbol{v_j} = [1{,}5]$
Iterasi 2
HimpunanSolusi = [nutritional, logistik]
$\boldsymbol{v_j} = [1{,}5 + 1{,}4]$
Iterasi 3
HimpunanSolusi = [nutritional, logistik, status]
$\boldsymbol{v_j} = [1{,}5 + 1{,}4 + 0{,}8]$
Iterasi 4
HimpunanSolusi = [nutritional, logistik, status, ordinal]
$\boldsymbol{v_j} = [1{,}5 + 1{,}4 + 1{,}1 + 0{,}3]$

Kondisi = $\sum_j x_j < 5 \ \ tapi \ \sum_{i=0}^{n-1} v_j < threshold$

Backtrack
HimpunanSolusi = [nutritional, logistik]
$\boldsymbol{v_j} = [1{,}5 + 1{,}4]$
Iterasi 1
HimpunanSolusi = [nutritional, logistik, support]
$\boldsymbol{v_j} = [1{,}5 + 1{,}4 + 0{,}9]$
Iterasi 2
HimpunanSolusi = [nutritional, logistic, support, logistic]
$\boldsymbol{v_j} = [1{,}5 + 1{,}4 + 0{,}9 + 0{,}8]$
Kondisi =$\sum_j x_j < 5 \ \ dan \ \sum_{i=0}^{n-1} v_j \geq threshold$

HimpunanSolusi = [nutritional, logistic, support, logistic]

3.2.5 Perancangan Basis Data

Setiap proses yang dilakukan pada tugas akhir ini menghasilkan suatu keluaran seperti kata pada proses ekstraksi kata dokumen, hasil terjemahan, hasil *document alignment*, hasil pembentukan *co-ocurrence*

Perancangan basis data untuk menyimpan hasil keluaran penghitungan maupun hasil ekstraksi kata yang akan digunakan adalah sebagai berikut, terdapat delapan tabel yang akan digunakan selama proses pembangunan tesaurus, yaitu tabel doc_ind, doc_ing, *term*_ing, *term*_ind, pair_doc, *term*s, *term*_dist, coocurrence. *Physical Data Model database* dapat dilihat pada Gambar 8.1 masing - masing kegunaan tabel adalah sebagai berikut:

1. Tabel doc_ind digunakan untuk menyimpan data dokumen tugas akhir dan tesis dalam bahasa Indonesia. Memiliki sembilan atribut, yaitu id_dokumen, kode, judul, abstrak, keyword, author, contributor1, contributor2, dan flag. Contoh data Tabel doc_ind dapat dilihat pada Tabel 8.1
2. Tabel doc_ing digunakan untuk menyimpan data dokumen tugas akhir dan tesis dalam bahasa Inggris. Memiliki sembilan atribut, yaitu id_dokumen, kode, title, abstract, keyword, author, contributor1, contributor2, dan flag. Contoh data Tabel doc_ing dapat dilihat pada Tabel 8.2.
3. Tabel *term*_ing digunakan untuk menyimpan hasil ekstraksi kata dari table doc_ing. Memiliki duabelas atribut, yaitu id_*term* id_dokumen, *term*, t0, t1, t2, t3, t4, t5, t6, t7, dan flag. Atribut t0 sampai t7 merupakan terjemahan Bahasa Indonesia dari atribut *term*. Contoh data Tabel term_ing dapat dilihat pada Tabel 8.4.
4. Tabel *term*_ind digunakan untuk menyimpan kata hasil ekstraksi kata dari table doc_ind. Memiliki enam atribut, yaitu id_*term*, id_dokumen, *term*, tf, idf, tfidf. Contoh data Tabel term_ind dapat dilihat pada Tabel 8.3

5. Tabel pair_doc digunakan untuk menyimpan hasil *document alignment*. Memiliki empat atribut, yaitu id, id_ind, id_ing dan score. Contoh data Tabel pair_doc dapat dilihat pada Tabel 8.7
6. Tabel *term*s digunakan untuk menyimpan hasil ekstraksi kata dari *document alignment*. Memiliki enam atribut, yaitu id_*term*, id_pair, *term*, tf, idf, tfidf. Contoh data Tabel terms dapat dilihat pada Tabel 8.8.
7. Tabel *term*_dist digunakan untuk menyimpan hasil penghitungan *weight factor*. Memiliki tujuh atribut, yaitu id_*term*d, *term*, tf, df, idf, tfidf dan weight_factor. Contoh data Tabel term_dist dapat dilihat pada Tabel 8.6.
8. Tabel coocurrence digunakan untuk menyimpan informasi *co-ocurrence*. Memiliki enam atribut, yaitu id_co, *term*1, *term*2, tf, idf dan relevance _weight. Contoh data Tabel coocurrence dapat dilihat pada Tabel 8.5

3.2.6 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 tesaurus menggunakan sigma.js.

3.2.6.1. Antar Muka Pencarian

Antar muka ini adalah antar muka awal pencarian dokumen dan graf, dari tesaurus saat tesaurus pertama kali dijalankan oleh pengguna. Hal yang dapat dilakukan oleh pengguna pada halaman ini adalah melakukan pencarian kata dan memilih tipe pencaian yang diinginkan.

Terdapat tombol pilihan pencarian dan tombol pencarian yang dapat digunakan untuk memasukkan kata, dan *threshold* pada tesaurus kemudian kata dan *threshold* tersebut digunakan sebagai

parameter pencarian sehingga tesaurus dapat menampilkan kata-kata yang memenuhi *constraint*.

Selain kotak pencarian, juga terdapat tautan untuk menuju antar muka lain yaitu antar muka visualisasi hasil. Antar muka pencarian dapat dilihat pada Gambar 3.5.

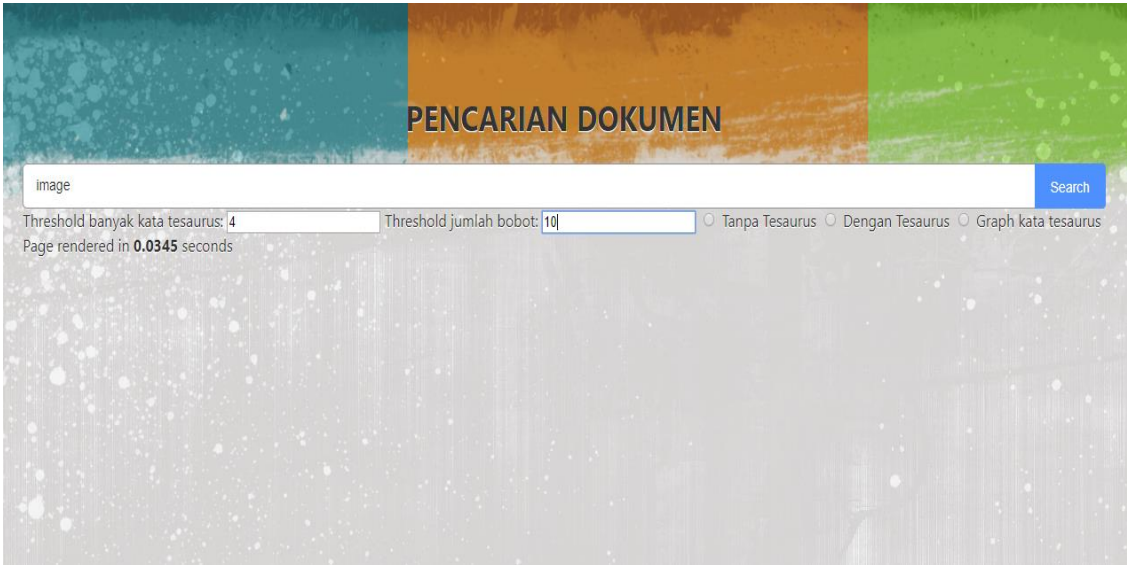
3.2.6.2. Antar Muka Hasil Pencarian Kata

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 pencarian dengan tesaurus atau tanpa menggunakan tesaurus. Antar muka hasil pencarian dokumen dapat dilihat pada Gambar 3.6.

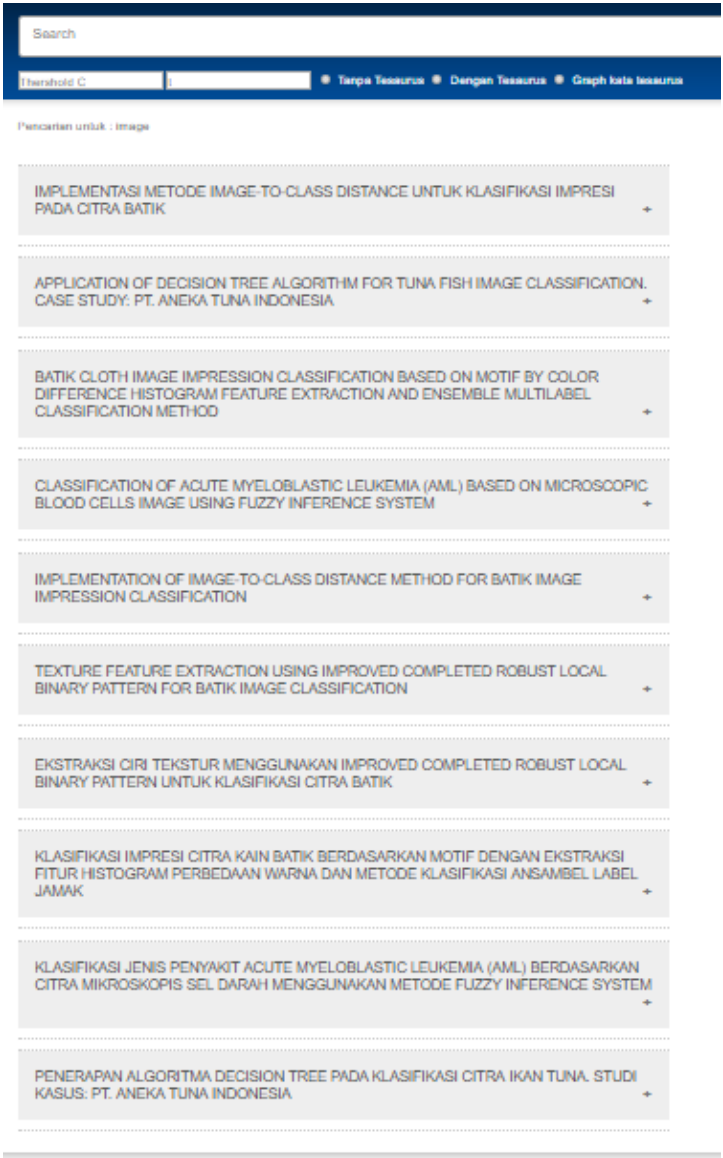
3.2.6.3. Antar Muka Visualisasi Hasil Tesaurus

Antar muka ini adalah antar muka yang digunakan untuk menampilkan hasil dari proses pembentukan tesaurus yang terbangun. Hal yang dapat dilakukan oleh pengguna pada halaman ini adalah melihat hasil visualisasi yang terbentuk. Antar muka visualisasi hasil tesaurus dapat dilihat pada Gambar 3.7.

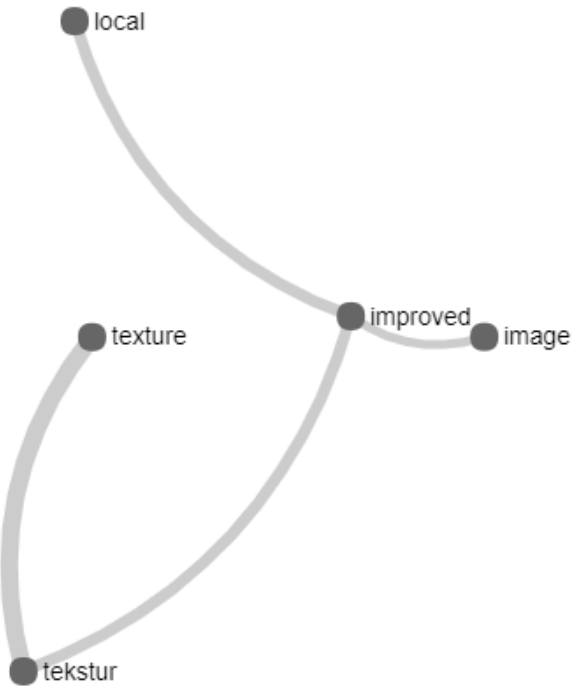
Visualisasi tesaurus dilakukan menggunakan pustaka *javascript* sigma.js.



Gambar 3.5 Antar Muka Pencarian Kata



Gambar 3.6 Hasil Pencarian Dokumen



Gambar 3.7 Antar Muka Visualisasi Hasil Tesaurus

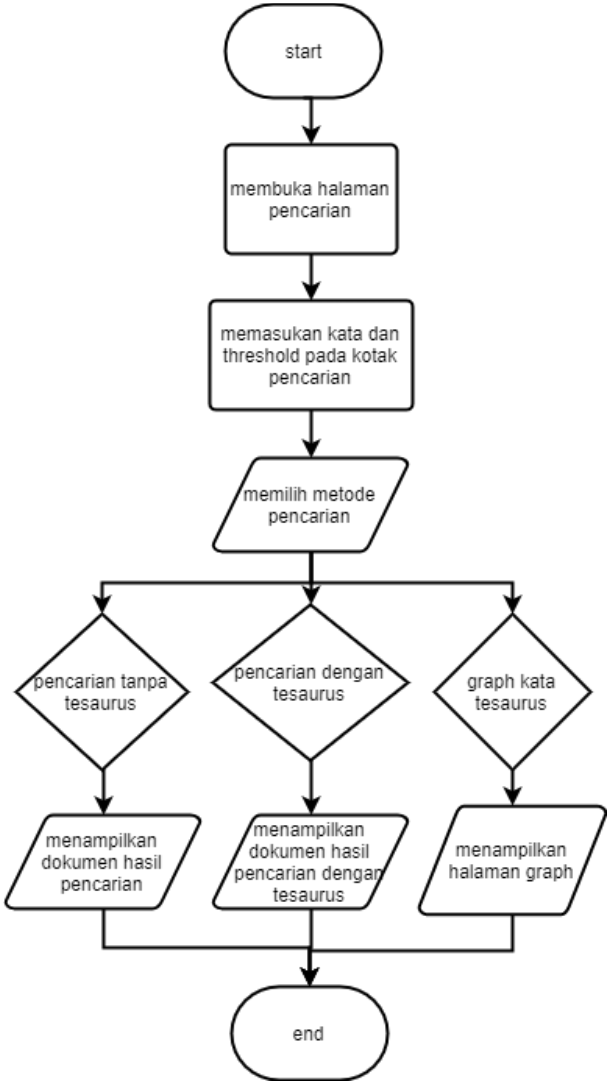
3.2.7 Perancangan Metode Pencarian Dokumen

Metode pencarian dokumen adalah metode yang digunakan untuk mengetahui hasil keluaran dari tesaurus yang telah dibangun. Hasil keluaran tersebut berupa daftar dokumen dari kata kunci yang di masukan oleh pengguna. Pencarian dokumen dilakukan dengan mencari pada dokumen mana saja *term* inputan itu berada, dan kemudian sistem akan menampilkan dokumen dengan urutan berdasarkan nilai tfidf tertinggi ke yang terendah. Pada pencarian dokumen dengan tesaurus, kata inputan pengguna akan secara otomatis ter-ekspansi dengan kata kata lain dari tesaurus yang terbentuk dari kata inputan pengguna. Diagram alur metode pencarian dokumen dapat dilihat pada Gambar 3.8.

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 dokumen terhadap kata inputan, pencarian kata dengan tesaurus dari kata inputan dan menampilkan kata kata tesaurus yang terbentuk dari kata inputan.

Metode pencarian dokumen memiliki tiga proses, yaitu:

1. Membuka halaman pencarian, untuk membuka antar muka pencarian kata tesaurus. Antar muka halaman pencarian dokumen dapat dilihat pada Gambar 3.85.
2. Masukkan kata pencarian ke dalam kotak pencarian, sehingga kata masukan dapat diproses dalam tesaurus
3. Pilih metode pencarian kata untuk menentukan keluaran yang diharapkan.
4. Pengguna dapat memilih melakukan pencarian tanpa menggunakan tesaurus dan dengan menggunakan tesaurus. Antar muka hasil pencarian dokumen tersebut dapat dilihat pada Gambar 3.6.
5. Apabila pengguna memilih melihat graf kata tesaurus yang terbentuk dari kata masukan, maka antar muka visualisasi graf dapat dilihat pada Gambar 3.7.



Gambar 3.8 Diagram Alur Metode Pencarian Dokumen

(Halaman ini sengaja dikosongkan)

BAB IV
IMPLEMENTASI

Bab ini membahas implementasi dari analisis dan perancangan sistem yang telah dibahas pada Bab III. Namun dalam penerapannya, rancangan tersebut dapat mengalami perubahan minor sewaktu-waktu apabila dibutuhkan.

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 : LEOVO G400S
- Prosesor : Inter® Core(TM) i5-3230M CPU (4 CPUs) @ 2.6GHz
- Memori (RAM): 4096 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
- Kerangka Kerja : CodeIgniter 3.0.3
- *Web Server* : Apache 2.4.4
- Basis Data : MySQL 5.5.32
- Kakas Bantu Basis Data: SQLyog Ultimate

Dalam penyusunan tugas akhir ini data diambil dari beberapa sumber basis data. Seperti daftar kata stop word pada tugas akhir ini menggunakan 2 stop word, data tugas akhir di its, dan basis data gkamus untuk terjemahan.

4.2 Daftar Stopword

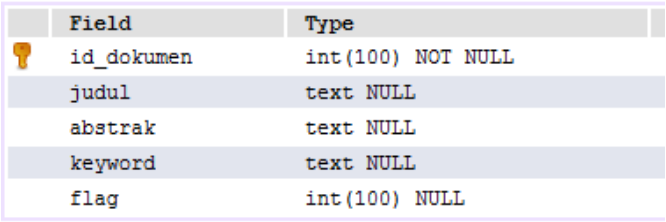
Pada pengerjaan tugas akhir ini terdapat dua proses yang menggunakan daftar *stopword* yaitu penghapusan *stopword* dari judul dokumen tugas akhir. Pada judul dokumen tugas akhir berbahasa Indonesia dan bahasa Inggris menggunakan daftar *stopword* yang berbeda dan dilakukan penambahan kata dalam *stopword* dari kata trivial yang sering muncul pada judul dokumen. Keterangan *stopword* tersebut dapat dilihat dalam Tabel 4.1

Tabel 4.1 Keterangan *Stopword*

Keterangan	Stop Word Dokumen Bahasa Indonesia	Stop Word Dokumen Bahasa Inggris
Sumber	A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia [9]	Stop word library Python nltk
Jumlah	359 kata	157 kata
Bahasa	Bahasa Indonesia	Bahasa Inggris

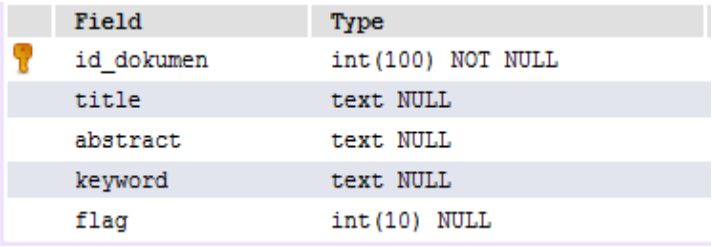
4.3 Basis Data Tugas Akhir ITS

Basis data sistem informasi repositori tugas akhir menggunakan MySQL 5.5.32. Berikut tabel pada basis data Tugas Akhir di ITS yang digunakan dalam tugas akhir ini. Gambar 4.1 – Gambar 4.2 dan Tabel 4.2 merupakan informasi *database* yang digunakan.



	Field	Type
🔑	id_dokumen	int(100) NOT NULL
	judul	text NULL
	abstrak	text NULL
	keyword	text NULL
	flag	int(100) NULL

Gambar 4.1 Tabel doc_ind



	Field	Type
🔑	id_dokumen	int(100) NOT NULL
	title	text NULL
	abstract	text NULL
	keyword	text NULL
	flag	int(10) NULL

Gambar 4.2 Tabel doc_ing

Tabel 4.2 Keterangan Tabel Hasil Duplikasi Data dari Basis Data Tugas Akhir ITS

doc_ind	
Keterangan	Tabel doc_ind merupakan tabel yang menyimpan data tugas akhir berbahasa Indonesia ITS
Jumlah Data	25.261
doc_ing	
Keterangan	Tabel doc_ing merupakan tabel yang menyimpan data tugas akhir berbahasa Ingris ITS
Jumlah Data	983

4.4 Gkamus

Pada basis data gkamus terdapat 2 tabel yaitu ind_ing dan ing_ind. ind_ing merupakan kamus terjemahan Bahasa Inggris ke Bahasa Indonesia, sedangkan ing_ind merupakan data terjemahan Bahasa inggris ke Bahasa Indonesia. Basisdata yang digunakan disini adalah table ing_ind. Implementasi basis data pada basis data

gkamus sebagai berikut. Gambar 4.4 dan tabel merupakan struktur dan informasi tabel yang dimiliki oleh gkamus.

Field	Type
kata	varchar(54) NULL
arti	varchar(268) NULL

Gambar 4.3 Tabel eng_ind

Tabel 4.3 Keterangan Tabel gkamus

Tabel	Keterangan	Jumlah Data
eng_ind	Tabel eng_ind menyimpan kata Bahasa inggris dan terjemahan dalam Bahasa indonesia	33.189

4.5 Implementasi Document Alignment

Tahapan implementasi dari hasil proses perancangan *Document Alignment* yang telah dilakukan pada Bab III. Seluruh tahapan implementasi Dokument *Alignment* sesuai dengan perancangan pada Bab III menggunakan bahasa pemrograman Python.

4.6 Praproses Data

Implementasi praproses data menggunakan bahasa pemrograman Python dan pustaka sesuai dengan perancangan pada Bab III. Praproses data yang dilakukan pada judul dokumen Bahasa Indonesia dan Bahasa Inggris adalah tokenisasi, penghapusan tanda baca kalimat, penghapusan stop word, dan stemming kata. Koneksi ke basis data menggunakan pustaka mysql.connector karena basis data Tugas Akhir menggunakan MySQL sebagai DBMS. Berikut implementasi setiap langkah pada praproses data.

Praproses data dilakukan pada judul dokumen Bahasa Indonesa dan judul dokumen Bahasa Inggris.

A. Tokenisasi

```
1. import mysql.connector
2. import re
3. def preprocess(text):
4. raw = text.split(' ')
5. cleaner = re.compile('[^a-zA-Z-]')
6. cleaned = []
7. ........
```

Kode Sumber 4.1 Implementasi Tokenisasi

Tokenisasi diimplementasikan pada awal setiap kali pembacaan data judul dokumen Bahasa Inggris dan Bahasa Indonesia. Implementasi tokenisasi menggunakan pustaka Python dari re dengan melakukan pemisahan kata terhadap tanda baca dan spasi. Setelah dilakukan tokenisasi pada kata judul dokumen, maka hasil dari tokenisasi disimpan dalam variabel cleaned untuk diproses pada proses selanjutnya. Kode Sumber 4.1 merupakan potongan kode untuk implementasi tokensasi pada dokumen.

B. Penghapusan Stop Word

```
1. import mysql.connector
2. import re
3. from nltk.corpus import stopwords
4. ...
5. row = [word for word in row if word not in stopword
   s.words('english')]
6. new_row = []
7. ...
```

Kode Sumber 4.2 Implementasi Penghapusan Stop Word

Implementasi penghapusan *stopword* menggunakan pustaka Python dari nltk dengan melakukan pengecekan kata *stopword* pada dokumen *stopword* di dalam dokumen C:\Users\[namauser]\AppData\Roaming\nltk_data\corpora\stopwords, hasil dari penghapusan *stopword* disimpan dalam variabel new_row untuk diproses pada proses selanjutnya. Kode Sumber 4.2 merupakan potongan kode untuk implementasi tokensasi pada dokumen.

C. Stemming Kata

```
1.  import mysql.connector
2.  from Sastrawi.StopWordRemover.StopWordRemoverFactor
    y
3.  import StopWordRemoverFactory
4.  ...
5.  for i in raw:
6.  cleaned.append(cleaner.sub('', i).lower())
7.  row = filter(None,cleaned)
8.  factory_stemmer = StemmerFactory()
9.  stemmer = factory_stemmer.create_stemmer()
10.
11. ....
```

Kode Sumber 4.3 Implementasi Stemming Kata Bahasa Indonesia

Implementasi stemming kata menggunakan pustaka Python Sastrawi dan nltk. Sastrawi merupakan pustaka pengolahan bahasa indonesia yang ada dalam bahasa Python. Kata yang diolah dalam tahap stemming adalah kata dalam dokumen bahasa Indonesia yang sudah dihilangkan stop wordnya. Kata yang telah dilakukan *stemmer* dimasukan pada variabel stemmer. Kode Sumber 4.3 merupakan potongan kode untuk stemming kata Bahasa Indonesia

4.6.1 Penterjemahan Kata Bahasa Inggris

A. Terjemahan Menggunakan Kamus

```
1.  import mysql.connector
2.  from nltk.corpus import stopwords
3.  from Sastrawi.Stemmer.StemmerFactory import Stem
    merFactor
4.  ....
5.  cursor = db.cursor()
6.  cursor.execute("SELECT distinct datafinal.term_i
    ng.term, kamus.kamuseng.arti FROM  datafinal.ter
    m_ing INNER JOIN kamus.kamuseng ON datafinal.ter
    m_ing.term = kamus.kamuseng.kata where datafinal
    .term_ing.flag =1 ")
7.  data = cursor.fetchall()
8.  ....
9.  word= []
10. word.append(str(data[index][0]))
11. for i in range(1,8):
12.     word.append(str(sorted_x[i-1][0]))
13.     cursor = db.cursor()
14.     for i in range(1,8):
15.         sqlQuery = "UPDATE datafinal.term_ing se
    t t"+str(i)+"='"+str(asem[i])+"' WHERE term = '"
    +str(asem[0])+"'""'"
16.                 cursor.execute(sqlQuery)
17.                 query = "UPDATE datafinal.term_i
    ng set flag = 2 WHERE term = '"+str(asem[0])+"'"

18.                 cursor.execute(query)
19.                 db.commit()
20.  ....
```

Kode Sumber 4.4 Terjemahan Menggunakan Kamus

Implementasi terjemahan menggunakan kamus, dilakukan dengan mencari terjemahan setiap kata bahasa inggris di dalam *database*, sehingga didapat terjemahan kata. Terjemahan kata yang didapat dilakukan penghapusan *stopword* dan *stemming* seperti pada Kode Sumber 4.2 – Kode Sumber 4.3, setelah itu hasilnya

disimpan dalam basis data dalam bentuk baris *record* untuk masing-masing kata. Kode Sumber 4.4 merupakan potongan kode untuk terjemahan menggunakan kamus.

B. Terjemahan Menggunakan Google Terjemahan

```
1.  import urllib2
2.  import mysql.connector
3.  from multiprocessing import Process
4.
5.  def translate(to_translate, to_langage="id", lan
    gage="eng"):
6.      proxy_support = urllib2.ProxyHandler({})
7.      opener = urllib2.build_opener(proxy_support)

8.      urllib2.install_opener(opener)
9.      agents = {'User-
    Agent':"Mozilla/4.0 (compatible; MSIE 6.0; Windo
    ws NT 5.1; SV1; .NET CLR 1.1.4322; .NET CLR 2.0.
    50727; .NET CLR 3.0.04506.30)"}
10.     before_trans = 'class="t0">'
11.     link = "http://translate.google.com/m?hl=%s&
    sl=%s&q=%s" % (to_langage, langage, to_translate
    .replace(" ", "+"))
12.     request = urllib2.Request(link, headers=agen
    ts)
13.     page = urllib2.urlopen(request).read()
14.     result = page[page.find(before_trans)+len(be
    fore_trans):]
15.     result = result.split("<")[0]
16.     return result
17. ...
```

Kode Sumber 4.5 Terjemahan Menggunakan Google Terjemahan

Implementasi terjemahan menggunakan google terjemahan, dilakukan dengan menggunakan pustaka Python urlib2, untuk mensiasati limit pada google terjemahan. Karena apabila menggunakan API google terjemahan akan dikenakan limit 1000

kta per hari. Kode Sumber 4.5 merupakan potongan kode untuk terjemahan menggunakan google terjemahan.

`urllib2.ProxyHandler` untuk menangai permintaan melalui *proxy*. Jika *proxy* dibutuhkan, diperlukan nama protokol untuk URL *proxy*. Jika tidak ada pengaturan *proxy*, pada *Windows* pengaturan *proxy* diperoleh dari bagian pengaturan Internet yang digunakan.

`urllib2.build_opener`, merupakan fungsi untuk membuat pembuka objek dengan fungsi tunggal. `urllib2.build_opener` menambahkan beberapa penangan secara *default*, tetapi menyediakan cara cepat untuk menambah dan / atau mengganti penangan *default* seperti *proxy* dan lain sebagainya.

`urllib2.Request`, untuk melakukan *request* terhadap suatu URL.

4.6.2 Pairing Dokumen

```
....
def lcs(x, y):
    n = len(x) #panjang list 1
    m = len(y) #panjang list 2
    himkata = dict()

    for i in range(n+1):     # i=0,1,...,n
        for j in range(m+1):  # j=0,1,...,m
            if i == 0 or j == 0:
                himkata[i, j] = 0
            elif x[i-1] == y[j-1]:
                himkata[i, j] = himkata[i-1, j-1] + 1
            else:
                himkata[i, j] = max(himkata[i-1, j], himkata[i, j-1])
    #himkata[n, m] LCS dari list 1 dan list 2
def final(i, j):
        if i == 0 or j == 0:
            return []
        elif x[i-1] == y[j-1]:
            return final(i-1, j-1) + [x[i-1]]
        elif himkata[i-1, j] > himkata[i, j-1]:
```

```
22.             return final(i-1, j)
23.         else:
24.             return final(i, j-1)
25.     return final(n, m)
26. #hitung jaccard simmilarity
27. for j in range(0,len(datae)):
28.     lists = final(data1, data2)
29.     n = len(lists[0])
30.     bagi1 =len(data1)+len(data3)
31.     bagi = bagi1 - n
32.     score = float(n) / float(bagi)
33. listData.append([datai[i][0],datae[j][0],score])
34. classifier = sorted(listData, key=lambda x:x[2], re
    verse= True)
35. listFinal.append([classifier[0][0],classifier[0][1]
    ,classifier[0][2]])
36. ...
```

Kode Sumber 4.6 Pembentukan Pairing Dokumen

Implementasi *Pairing* Dokumen pada tugas akhir ini menggunakan bahasa Python, pairing dokumen dilakukan antara judul dokumen Bahasa Indonesia dengan kemungkinan terjemahan judul bahasa Inggris ke Bahasa Indonesia. Penghitungan dilakukan dengan menggunakan *Jaccard simmilarity*. Pertama, setiap term pada tiap dokumen bahasa Indonesia dan kemungkinan terjemahan judul bahasa Inggris ke Bahasa Indonesia dilakukan pencaarian kesamaan kata yang dilakukan pada fungsi final(). Setelah mendapatkan nilai pasangan, kemudian dilakukan penghitungan *jaccard similarity*, sehingga masing masing pasangan antara dokumen bahasa Indonesia dan bahasa Inggris memiliki nilai kesamaan. Nilai pasangan tertinggi masing – masing dokumen seluruh data pada listFinal dimasukkan ke dalam basis datad. Kode Sumber 4.7 merupakan potongan kode untuk pembentukan pairing dokumen.

4.7 Pembentukan Informasi *Co-occurrence*

Sebelumnya akukan praproses data pada dokumen seperti yang telah dijelaskan pada subbab 4.7 dari *document alignment* yang terbentuk.

4.7.1 Pembobotan *Term*

```
1.  import mysql.connector
2.  import math
3.  ...
4.  def compute_idf(ndoc, data):
5.      for i in data:
6.          data[i].append(math.log( float(ndoc[0]) / d
    ata[i][1], 10))
7.      return data
8.
9.  def compute_tfidf(data):
10.     for i in data:
11.         data[i].append( data[i][0] * data[i][2])
12.     return data
13. ...
```

Kode Sumber 4.7 Pembobotan Term

Implementasi pembobotan term menggunakan pustaka Python math untuk melakukan penghitungan menggunakan log. Kode Sumber 4.8 merupakan potongan kode untuk implementasi pembobotan term idf dan tfidf.

4.7.2 Penghitungan Weighting Factor

```
1.  import mysql.connector
2.  import math
3.  ...
4.  weight_factor = 0.0
5.  for i in data:
6.      weight_f = i[1]
7.      weight_factor = weight_f/math.log(lendoc)
```

```
8.      print i
9.      cursor = db.cursor()
10.     cursor.execute("update term_dist set weight_fac
    tor = %s where id_termd = %s", (weight_factor, i[0]
    ))
11.     db.commit()
12. ...
```

Kode Sumber 4.8 Penghitungan *Weight Factor*

Implementasi penghiungan weight factor menggunakan pustaka Python math untuk melakukan penghitungan menggunakan log. Kode Sumber 4.9 merupakan potongan kode untuk melakukan penghitungan *weight factor*.

4.7.3 Pembentukan Pair Kata

```
1.  import mysql.connector
2.  import math
3.  ...
4.  kam = {}
5.      kamdf= {}
6.      for i in doc:
7.          gen = (x for x in daftar if x[1]==i[0])
8.          la.write("-
    "+ str(c)+" of "+str(len(doc))+"\n")
9.          c=c+1
10.         for j in gen:
11.             gen1 = (x for x in daftar if x[0]>j[0]
    and x[1]==i[0])
12.             for k in gen1:
13.                 args=(j[2], k[2])
14.                 df = 0
15.                 tf = min(j[3], k[3])
16.                 if tf > 0:
17.                     df = 1
18.                 if not args in kam:
19.                     kam[args]=1
20.                 else:
```

```
                    kam[args]+=tf
                if not args in kamdf:
                    kamdf[args]=df
                else:
                    kamdf[args]+=df

                cooc.add(args)
                c1=c1+1;

    que = ''
    for i in list(sorted(cooc)):
        #print kam[i]
        if(kamdf[i]==0):
            idf = 0
        else:
            idf = math.log10(len(doc)/kamdf[i])
        que=que+"INSERT INTO cooccurence VALUES(NUL
L,'"+str(i[0])+"','"+str(i[1])+"', "+str(kam[i])+",
 "+str(idf)+", 0);\n"
        la.write("inserting: "+str(i)+" tf: "+str(k
am[i]) + " df: "+ str(idf)+"\n")
   la.close()
...
```

Kode Sumber 4.9 Pembentukan Pair Kata

Implementasi pembentukan pair kata dillakukan agar mendapatkan pasangan *co-occurrence* setiap kata.. Kode Sumber 4.10 merupakan potongan kode untuk melakukan pembentukan pair kata.

4.7.4 Penghitungan Relevance Weight

```
import mysql.connector
import math
...
rows = []
    for i in data:
        d1 = terms[i[1]]
        d2 = terms[i[2]]
```

```
8.          tf = d1[0]
9.          idf = d1[1]
10.         weight_factor2 = d1[2]
11.         tf2 = d2[0]
12.         idf2 = d2[1]
13.         weight_factor = d2[2]
14.         atas = i[3]*i[4]
15.         relevance
    _weight = atas/tf*idf*weight_factor
16.         relevance
    _weight_2 = atas/tf2*idf2*weight_factor2
17.         cursor = db.cursor()
18.         rows = rows + [[str(relevance
    _weight), str(relevance _weight_2), str(i[0])]]
19. ...
```

Kode Sumber 4.10 Pembentukan *Relevance Weight*

Implementasi penghiungan *relevance weight* dicari untuk mendapatkan nilai kedekatan tiap pasangan kata dari *co-occurrence*. Kode Sumber 4.11 merupakan potongan kode untuk melakukan penghitungan *relevance weight.*

4.8 Constraint Satisfaction Problem

```
1.  ...
2.  def backTrack():
3.      kata = request.args.get('i')
4.      c =  request.args.get('c')
5.      threesold =  request.args.get('t')
6.      threshold = float(threesold)
7.      C = int(c)
8.      input = kata.split('_')
9.      himKata = []
10.     for i in range(0,len(input)):
11.         query = "select id_co, term2, relevance
    _weight from coocurrence where term1 ='"+input[i].s
    trip()+""
12.         cursor.execute(query)
13.         data = cursor.fetchall()
```

```
14.         sortBesar = sorted(data, key=lambda x:x[2],
     reverse= True)
15.         if(len(sortBesar) > 0):
16.             himKata.append([sortBesar[0][0],sortBes
    ar[0][1],sortBesar[0][2]]) #ini yang baru
17.     tmp = []
18.     kataBanned = []
19.     batasC = C/2
20.     back = 1
21.     if(len(himKata) == 0): #kalo yang tidak ditemuk
    an
22.         stringFinal = '[{'+'"id":"notFound"'+'}]'
23.         return stringFinal
24.     nInput = len(input)
25.     while True:
26.         listSementara = []
27.         for i in range(0,len(himKata)+nInput):
28.             query = ""
29.             if(i<nInput):
30.                 for z in range(0,nInput):
31.                     query = "select id_co, term2, r
    elevance
    _weight from coocurrence where term1 ='"+input[z]+"
    '"
32.             else:
33.                 query = "select id_co, term2, relev
    ance
    _weight from coocurrence where term1 ='"+himKata[i-
    nInput][1]+"'"
34.             cursor.execute(query)
35.             data = cursor.fetchall()
36.             listSementara = listSementara + data
37.         sortBesar = sorted(listSementara, key=lambd
    a x:x[2], reverse= True)
38.         if(len(kataBanned) != 0):
39.             flagBesar = True
40.             for x in range(0,len(sortBesar)):
41.                 flag = True
42.                 for y in range(0,len(kataBanned)):

43.                     if(kataBanned[y] == sortBesar[x
    ][0]):
44.                         flag = False
```

```
                        break
                if(flag):
                    if(any(sortBesar[x][0] in subli
 for subli in himKata)):
                        pass
                    else:
                        himKata.append([sortBesar[x
][0],sortBesar[x][1],sortBesar[x][2]])
                        break
                elif(x==len(sortBesar)-1):
                    tmp.append(himKata)
                    back = back + 1
                    batasAkhir = 1 + back
                    kataBanned = []
                    kataBanned.append(himKata[jumla
h-batasAkhir][0])
                    batasAkhir = -1 * batasAkhir
                    himKata = himKata[:(back-1)*-
1]
        else:
            for i in range(0,len(sortBesar)):
                if(any(sortBesar[i][0] in sublist f
or sublist in himKata)):
                    pass
                else:
                    himKata.append([sortBesar[i][0]
,sortBesar[i][1],sortBesar[i][2]])
                    break
        maxValue = sum(row[2] for row in himKata)
        jumlah = len(himKata)
        if(maxValue >= threshold and jumlah <= C):

            break
        elif (jumlah >= C):
            back = back + 1
            tmp.append(himKata)
            batasAkhir = 2
            kataBanned.append(himKata[jumlah-
batasAkhir][0])
            batasAkhir = -1 * batasAkhir
            himKata = himKata[:batasAkhir]
        elif (back==batasC+1):
            listJumlah = []
```

```
80.             for o in range(0,len(tmp)):
81.                 jumlahAkhir = 0.0
82.                 for p in range(0,len(tmp[o])):
83.                     jumlahAkhir = jumlahAkhir + tmp
    [o][p][2]
84.                 listJumlah.append([o,jumlahAkhir])

85.             sortTmp = sorted(listJumlah, key=lambda
     x:x[1], reverse= True)
86.             himKata = tmp[sortTmp[0][0]]
87.             break
88.             return himKata
89.
```

Kode Sumber 4.11 *Constraint Satisfaction Problem*

Implementasi *Constraint Satisfaction Problem* pada tugas akhir ini menggunakan bahasa Python. *Constraint Satisfaction Problem* dilakukan untuk mendapatkan solusi sesuai dengan kebutuhan. Solusi untuk pencarian solusi digunakan *backtracking* sebagai perbaikan dari metode *brute-force*. Kode Sumber 4.8 merupakan potongan kode untuk *Constraint Satisfaction Problem*. Setiap *term* yang memenuhi *constraint* akan dimasukan pada variable himKata[], sedangkan *term* yang mengakibatkan infeasible solution akan dimasukan pada variable kataBanned[]

Kode Sumber 4.11 merupakan potongan kode untuk Backtracking untuk mendapatkan solusi dari CSP secara maksimal, terkadang terdapat beberapa kasus yang menyebabkan infinite loop, karena tidak mendapat solusi maksimal. Hal itu diakibatkan karena besarnya *threshold* yang ditentukan. Untuk mengatasi hal tersebut, maka setiap proses backtracking akan disimpan dalam suatu array, dan looping akan berhenti pada $\frac{Treshold\ C}{2} + 1$. Lalu array yang telah menyimpan kandidat solusi diurutkan dan diambil nilai yang paling maksimal.

(Halaman ini sengaja dikosongkan)

BAB V
UJI COBA DAN IMPLEMENTASI

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 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 : LEOVO G400S
   Prosesor : Inter® Core(TM) i5-3230M CPU (4 CPUs) @ 2.6GHz
   Memori (RAM): 4096 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 data Tugas Akhir serta Tesis mahasiswa FTIF, Jaringan Cerdas Multimedia, Statistika, Teknik Multimedia Jaringan di Institut Teknologi Sepuluh Nopember dengan bahasa Inggris dan Bahasa Indonesia.

Dokumen dibagi ke dalam beberapa jumlah tertentu yaitu yaitu 35 dokumen bahasa Indonesia dan 32 dokumen bahasa Inggris dengan topik yang sama, serta 865 dokumen bahasa Indonesia dan 637 dokumen bahasa Inggris.

Pembagian banyak jumlah data dan pembagian dokumen berdasarkan jurusan 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 jurusan digunakan untuk mengetahui keluaran tesaurus dipengaruhi oleh data jurusan yang digunakan untuk membangun tesaurus.

5.2 Kuesioner Uji Coba

Kuesioner uji coba perlu diterapkan pada tugas akhir ini karena tesaurus yang terbentuk tidak memilki data kebenaran yang pasti. Kuisioner berfungsi untuk menilai kepuasan pengguna mengenai hasil tesaurus yang terbentuk serta untuk melakukan pembentukan *ground truth*. Target responden dari kuesioner ini 25 adalah mahasiswa aktif dan alumni teknik informatika, mahasiswa statistika, mahasiswa matematika. Target responden adalah mahasiswa atau alumni yang mengerti bahasa Indonesia serta bahasa Inggris.

Skenario pengisian kuesioner dilakukan dalam dua tahap yaitu:

a) Kuesioner tahap pertama bertujuan untuk membandingkan hasil kata yang dibentuk berdasakan jumlah data pembentukan tesaurus. Setelah melihat hasil tesaurus yang terbentuk dari jumlah data yang berbeda, responden diminta untuk memberikan opini tentang apakah jumlah data serta memperbanyak dokumen dengan kesamaan topik yang serupa mempengaruhi hasil tesaurus yang terbentuk. Hasil dari kuesioner tahap pertama akan digunakan untuk menarik kesimpulan bawa data yang digunakan akan berpengaruh dengan kualitas tesaurus

yang terbentuk. Contoh hasil jawaban dari kuisioner 1 ditunjukan pada Tabel 8.11

b) Kuesioner tahap kedua memiliki tujuan untuk membandingkan pembentukan tesaurus menggunakan *constraint satisfaction problem* dan tesaurus yang dibentuk dengan tidak menggunakan *constraint satisfaction problem* (*co-occureence*). Responden akan diminta untuk menentukan *ground truth* kata-kata apa saja yang relevan dengan *keyword* dan membandingkan hasil dari kedua metode tesaurus tersebut. Dari *ground truth* tersebut dapat dihitung nilai *pecision* dari kata tesaurus yang terbentuk. Contoh gambar tesaurus yang terbentuk dapat dilihat pada Gambar 8.2 sampai dengan Gambar 8.3. Selanjutnya responden akan diminta untuk menentukan suatu *ground truth* dari hasil pencarian dengan suatu dokumen. Lalu *ground truth* tersebut dibandingkan dengan keluaran pencarian dengan menggunakan tesaurus. Terakhir pengguna diminta untuk melakukan pencarian dokumen menggunakan pencarian biasa dan pencarian menggunakan tesaurus, perbandingan pencarian dapat dilihat pada Tabel 8.9 dan kemudian responden diminta untuk menjawab pertanyaan. Contoh hasil jawaban dari kuisioner 2 ditunjukan pada Tabel 8.12

5.3 Uji Coba 1

Uji coba 1 bertujuan untuk membandingkan hasil kata yang dibentuk berdasakan jumlah data pembentukan tesaurus. Uji coba 1 dilakukan dengan meminta responden mengisi kuisioner, setelah melihat hasil tesaurus yang terbentuk dari jumlah data yang berbeda, responden diminta untuk memberikan opini tentang apakah jumlah data serta memperbanyak dokumen dengan kesamaan topik yang serupa mempengaruhi hasil tesaurus yang terbentuk. Hasil dari kuesioner tahap pertama akan digunakan

untuk menarik kesimpulan bahwa data yang digunakan akan berpengaruh dengan kualitas tesaurus yang terbentuk.

5.3.1 Skenario Uji Coba

Tahapan-tahapan uji coba yang dilakukan untuk mendapatkan hasil kepuasan responden dari skenario uji coba 1 adalah sebagai berikut:

1. Menampilkan tesaurus yang akan diuji dengan jumlah data yang pertama yaitu 35 dokumen bahasa Indonesia dan 32 dokumen bahasa Inggris dengan topik yang serupa.
2. Menampilkan tesaurus yang akan diuji dengan jumlah data yang pertama yaitu 865 dokumen bahasa Indonesia dan 637 dokumen bahasa Inggris dengan topik yang berbeda-beda.
3. Selanjutnya responden diminta untuk menjawab pertanyaan nomor 1 yaitu basis data manakah yang dilakukan uji coba yang memiliki hasil terbaik
4. Selanjutnya responden diminta untuk menjawab pertanyaan nomor 2 yaitu apakah jumlah data mempengaruhi pembentukan tesaurus
5. Selanjutnya responden diminta untuk menjawab pertanyaan nomor 3 yaitu apakah data yang memiliki topik serupa dapat membentuk tesaurus yang lebih baik.

5.3.2 Hasil Uji Coba

- Data 1 = data berjumlah lebih sedikit namun dengan dokumen yang memiliki topik serupa (35 dokumen bahasa Indonesia dan 32 dokumen bahasa Inggris)
- Data 2 = data berjumlah lebih banyak dan dokumen memiliki topik yang bervariatif (865 dokumen bahasa Indonesia dan 637 dokumen bahasa Inggris)

Tabel 5.1 Hasil Uji Coba 1

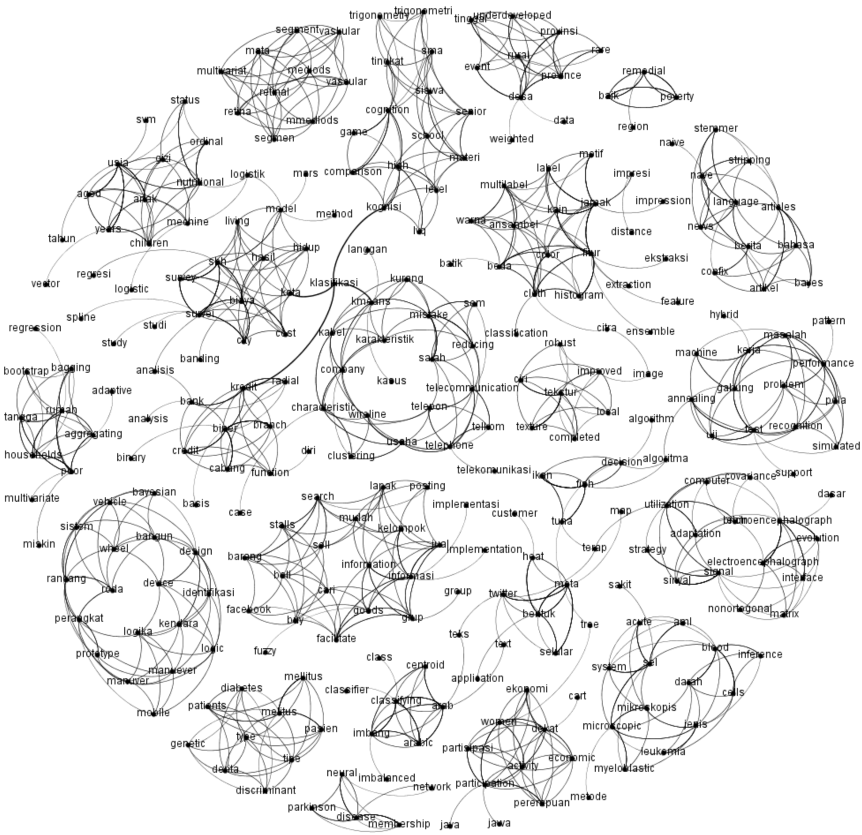
No	Item	Informasi
1	Berapa jumlah data dari basis data yang dilakukan uji coba yang memiliki hasil terbaik?	Data 1= 20 Data 2 = 5
2	Apakah jumlah data mempengaruhi pembentukan tesaurus?	Ya = 23 Tidak = 2
3	Apakah data yang memiliki topik serupa dapat membentuk tesaurus yang lebih baik?	Ya = 24 Tidak = 1
4	Total jumlah responden	25 mahasiswa

Gambar 5.1 merupakan graf CSP yang terbentuk dari data 1, sedangkan Gambar 5.2 merupakan graf CSP yang terbentuk dari data 2.

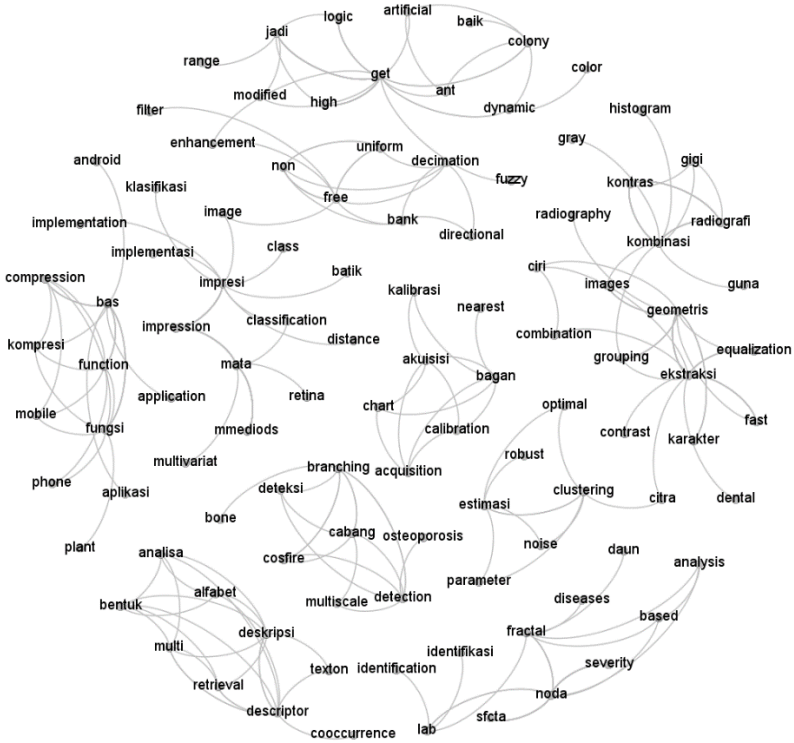
5.3.3 Analisis dan Evaluasi

Pada hasil skenario uji coba 1 didapatkan hasil tesaurus lebih baik ditunjukan pada data 1 yaitu data berjumlah lebih sedikit namun dengan dokumen yang memiliki topik serupa. Beberapa hal yang dapat diambil dari hasil uji coba di atas selain informasi tersebut adalah sebagai berikut:

1. Jumlah data mempengaruhi pembentukan tesaurus, karena apabila data semakin banyak, maka kata – kata yang terbentuk akan semakin bervariatif.
2. Data yang memiliki topik serupa akan membentuk tesaurus yang lebih baik, karena tesaurus yang terbentuk berasal dari topik yang sama, maka tesaurus yang terbentuk akan semakin relevan.
3. Data yang dapat membentuk tesaurus yang lebih baik adalah data yang reletif memiliki jumlah lebih banyak, namun dengan jumlah dokumen masing - masing topik yang lebih banyak.



Gambar 5.1 Graf CSP Data 1



Gambar 5.2 Graf CSP Data 2

5.4 Uji Coba 2

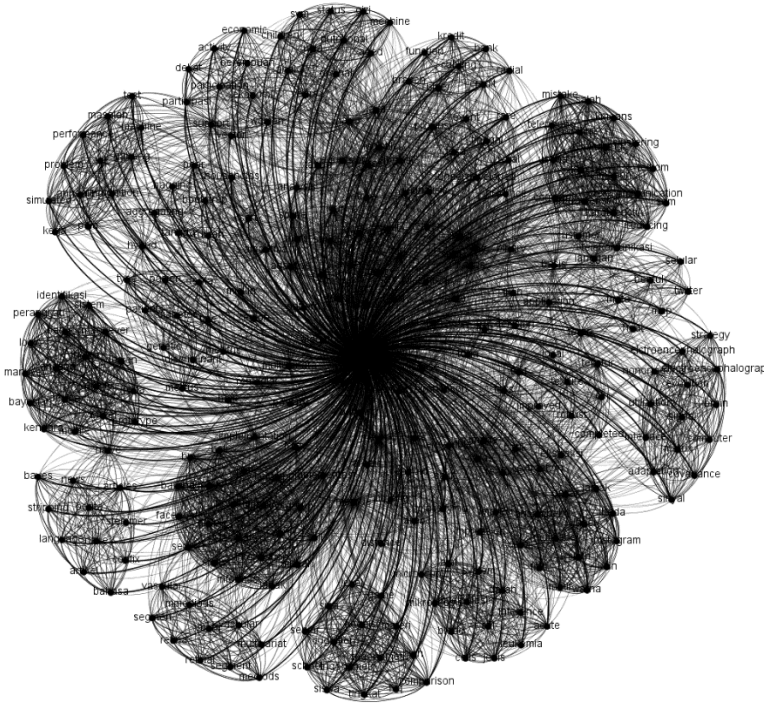
Uji coba 2 bertujuan untuk membandingkan pembentukan tesaurus menggunakan *constraint satisfaction problem* dan tesaurus yang dibentuk dengan tidak menggunakan *constraint satisfaction problem* (*co-occureence*). Uji coba ini menggunakan 865 dokumen bahasa Indonesia dan 637 dokumen bahasa Inggris dengan topik yang tidak serupa. Uji coba dilakukan dengan meminta responden menentukan *ground truth* kata kata apa saja yang relevan dengan *keyword* dan membandingkan hasil dari

kedua metode tesaurus tersebut. Dari hasil opini tersebut dapat dihitung nilai *pecision* dari kata tesaurus yang terbentuk.

5.4.1 Skenario Uji Coba

Tahapan-tahapan uji coba yang dilakukan untuk mendapatkan *precision* dari skenario uji coba 2 adalah sebagai berikut:

1. Menampilkan kata tesaurus dengan menggunakan *constraint satisfaction problem* dari *keyword* masukan.
2. Menampilkan kata tesaurus yang terbentuk yang tanpa menggunakan *constraint satisfaction problem* (*co-occurrence*) dari *keyword* masukan.
3. Selanjutnya responden diminta untuk menilai relevan atau tidaknya masing masing kata keluaran tesaurus dengan menandai kata kata yang relevan. Nilai tersebut yang selanjutnya digunakan untuk menghitung *precision*.
4. Responden diminta untuk menjawab pertanyaan nomor 1 yaitu apakah tesaurus yang dibentuk menggunakan CSP lebih baik daripada non CSP
5. Selanjutnya responden melakukan pencarian dokumen dengan *keyword* tertentu dengan pencarian biasa dan dengan menggunakan tesaurus
6. Selanjutnya responden diminta untuk menjawab pertanyaan nomor 2 yaitu apakah pencarian yang menggunakan tesaurus lebih relevan



Gambar 5.3 Graf Co-occurrence Kata Uji Coba 2

	keyword : GAME	Tesaurus + CSP	responden	Tesaurus tanpa csp	responden
		gameplay	gameplay	gameplay	gameplay
		main	main	versus	vesus
		timestrategy	timestrategy	hajj	
		hierarchical	machine	troops	
		machine	profile	trigonometry	
		profil	profil		
		profile	real		
		real			

Gambar 5.4 Kuisioner Uji Coba 2

Gambar 5.1 merupakan Graf CSP pada data 1 pada uji coba 1 yang digunakan untuk melakukan uji coba 2, uji coba 2 dilakukan dengan membandingkan kata berdasarkan relevansi menurut responden seperti yang terdapat pada Gambar 5.4 dan kemudian akan didapat *precision* dari kata tesaurus yang terbentuk.

5.4.2 Hasil Uji Coba

Tabel 5.2 Hasil Uji Coba 2

No	Item	Informasi
1	Apakah pencarian yang menggunakan tesaurus lebih relevan?	Ya = 16, Tidak = 0
2	Apakah tesaurus yang dibentuk menggunakan CSP lebih baik daripada non CSP?	Ya =16, Tidak = 0
	Total jumlah responden	16 mahasiswa

Dari pendapat responden terebut mengenai tesaurus yang terbentuk, dapat ditarik kesimpulan mengenai *precison* dari pembentukan tesaurus menggunakan *constraint satisfaction problem* dan tanpa menggunakan *constraint satisfaction problem*. Perhitungan *precison* dapat dilihat pada Persamaan *(5.1)*. Hasil rata– rata precision CSP dan Non CSP dapat dilihat dalam Tabel 5.3.

$$Precision = \frac{Jumlah\ tesaurus\ yang\ relevan}{Jumlah\ tesaurus\ yang\ ditampilkam} \times 100\% \quad (5.1)$$

Tabel 5.3 Hasil Perbandingan Precision CSP dan Non CSP Data 1

	CSP (%)	Non CSP (%)
Precision	68,29	45,23

5.4.3 Analisis dan Evaluasi

Pada hasil skenario uji coba 2 yang diberikan tesaurus, didapatkan beberapa hal yang dapat diambil dan dilakukan analisis yaitu:

1. Pembentukan tesaurus menggunakan *constraint satisfaction problem* menghasilkan kata kata tesaurus yang lebih relevan dengan tesaurus yang dibentuk tanpa menggunakan *constraint satisfaction problem.*
2. Ada beberapa kata yang memiliki *precision* rendah dikarenakan data tersebut tidak memiliki topik serupa
3. Tesaurus yang dibentuk dengan menggunakan CSP menghasilkan *precision* 68,29% sedangkan tesaurus yang dibentuk tanpa menggunakan CSP menghasilkan *precision* 45,23%.

5.5 Uji Coba 3

Uji coba 3 bertujuan untuk membandingkan pembentukan tesaurus menggunakan *constraint satisfaction problem* dan tesaurus yang dibentuk dengan tidak menggunakan *constraint satisfaction problem.* Tesaurus yang dibentuk dengan tidak menggunakan *constraint satisfaction problem* (Non CSP) adalah tesaurus yang terbentuk dari informasi *co-occurrence*, dan dilakukan pengambilan lima kata dengan nilai *relevant weight* tertinggi. Uji coba akan dilakukan dengan 35 dokumen bahasa Indonesia dan 32 dokumen bahasa Inggris dengan topik yang serupa. Uji coba dilakukan dengan meminta responden menentukan *ground truth* kata apa saja yang relevan dengan *keyword* dan membandingkan hasil dari kedua metode tesaurus tersebut. Contoh *ground truth* tesaurus ditunjukan pada Tabel 8.10 Dari hasil opini tersebut dapat dihitung nilai *pecision* dari kata tesaurus yang terbentuk.

Selain itu juga responden diminta untuk melakukan pembentukan *ground truth* pencarian dokumen dengan kata kunci tertentu, selanjutnya dilakukan perbandingan hasil pencarian

dokumen dengan menggunakan tesaurus dibandingkan dengan *ground truth* sehingga dapat dilakukan penghitungan akurasi. Contoh *ground truth* pencarian dokumen ditunjukan pada Tabel 8.13 *Ground truth* adalah kunci jawaban hasil pencarian dokumen relevan yang dibuat secara manual.

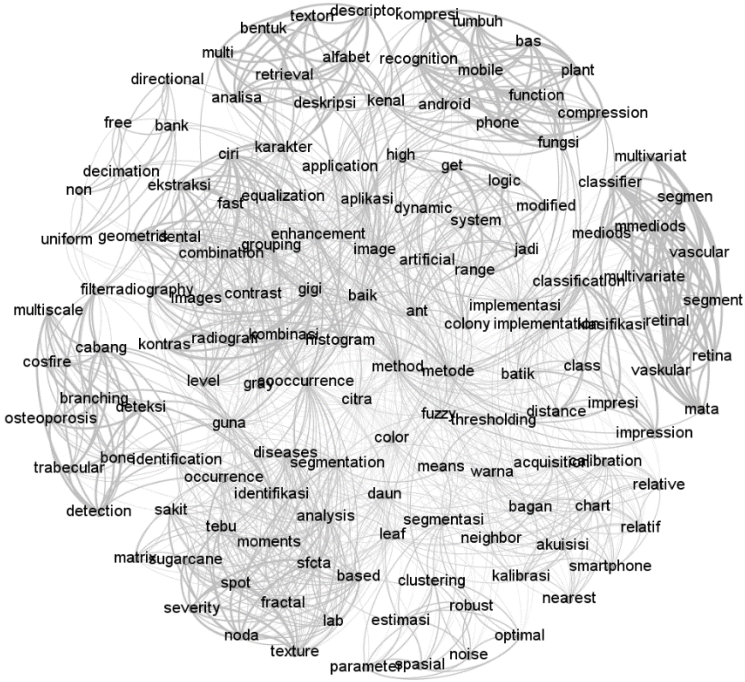
Terakhir pengguna diminta untuk melakukan pencarian dokumen menggunakan pencarian biasa dan pencarian menggunakan tesaurus, perbandingan pencarian dapat dilihat pada Tabel 8.9 dan kemudian responden diminta untuk menjawab pertanyaan.

5.5.1 Skenario Uji Coba

Tahapan-tahapan uji coba yang dilakukan untuk mendapatkan hasil performa dan pendapat responden dari skenario uji coba 3 adalah sebagai berikut:

1. Menampilkan tesaurus dengan menggunakan *constraint satisfaction problem* dari *keyword* masukan
2. Menampilkan tesaurus yang terbentuk yang tanpa menggunakan *constraint satisfaction problem* (*co-occurrence*) dari *keyword* masukan
3. Selanjutnya responden diminta untuk menilai relevan atau tidaknya masing masing kata keluaran tesaurus dengan menandai kata kata yang relevan. Nilai tersebut yang selanjutnya digunakan untuk menghitung *precision*
4. Responden diminta untuk menjawab pertanyaan nomor 1 yaitu apakah tesaurus yang dibentuk menggunakan CSP lebih baik daripada non CSP
5. Selanjutnya responden diminta untuk menentukan *ground truth* pencarian dokumen dari suatu kata.
6. Dilakukan perbandingan hasil pencarian dokumen dengan menggunakan tesaurus dan *ground truth* yang telah ditentukan responden
7. *Precision*, *Recall* dan Akurasi dapat dihitung dari hasil tersebut

8. Selanjutnya responden melakukan pencarian dokumen dengan *keyword* tertentu dengan pencarian biasa dan dengan menggunakan tesaurus
9. Selanjutnya responden diminta untuk menjawab pertanyaan nomor 2 yaitu apakah Pencarian yang menggunakan tesaurus lebih baik



Gambar 5.2 merupakan Graf CSP pada data 2 pada uji coba 1 yang digunakan untuk melakukan uji coba 3, uji coba 2 dilakukan karena hasil tesaurus yang terbentuk tidak memiliki hasil yang relative baik. Maka pada uji coba 3 dilakukan dengan membandingkan kata berdasarkan relevansi menurut responden

seperti yang terdapat pada Gambar 5.4 dan kemudian akan didapat *precision* dari kata tesaurus yang terbentuk.

5.5.2 Hasil Uji Coba

Tabel 5.4 Hasil Uji Coba 3

No	Item	Informasi
1	Apakah pencarian yang menggunakan tesaurus lebih relevan?	Ya = 25, Tidak = 0
2	Apakah tesaurus yang dibentuk menggunakan CSP lebih baik daripada non CSP?	Ya =25, Tidak = 0
	Total jumlah responden	25 mahasiswa

Dari pendapat responden terebut mengenai tesaurus yang terbentuk, dapat ditarik kesimpulan mengenai *precison* dari pembentukan tesaurus menggunakan *constraint satisfaction problem* dan tanpa menggunakan *constraint satisfaction problem.* Perhitungan *precison* dapat dilihat pada Persamaan

(*5.1*). Hasil rata – rata precision CSP dan Non CSP dapat dilihat dalam Tabel 5.3.

Tabel 5.5 Hasil Perbandingan Precision CSP dan Non CSP Data 2

	CSP (%)	Non CSP (%)
Precision	91,38	55,28

Dari percobaan pencarian dengan *query* ekspansi menggunakan tesaurus yang dibandingkan dengan *ground truth*, maka dapat dihitung nilai *recall* sesuai pada Persamaan *(5.2, precision* sesuai pada Persamaan *(5.3* serta akurasi sesuai dengan Persamaan (*5.4*

$$Recall = \frac{TP}{TP + FN} \times 100\% \tag{5.2}$$

$$Precision = \frac{TP}{TP + FP} \times 100\% \quad (5.3)$$

$$Akurasi = \frac{TP + TN}{TP + FP + TN + FN} \times 100\% \quad (5.4)$$

Dimana TP adalah *True Positive* yang berarti data relevan yang diterima dari pencarian yang dicocokan dengan *ground truth*, TN adalah *True Negative* yang berarti data tidak relevan yang diterima dari pencarian yang dicocokan dengan *ground truth*. FN adalah *False Negatif* yang berarti jumlah data yang relevan pada *ground truth* yang tidak ada pada hasil pencarian. FP adalah *False Positive* yang berarti jumah data yang tidak relevan yang terdapat pada hasil pencarian. *Ground truth* adalah kunci jawaban hasil pencarian dokumen relevan yang dibuat secara manual.

Tabel 5.6 Hasil Performa Uji Coba 3

	Pencarian dengan tesaurus menggunakan CSP (%)	Pencarian tanpa tesaurus (%)
Recall	86,67	63,33
Precision	100	100
Akurasi	86,67	63,33

5.5.3 Analisis dan Evaluasi

Pada hasil skenario uji coba 2 yang diberikan tesaurus, didapatkan beberapa hal yang dapat diambil dan dilakukan analisis yaitu:

1. Pembentukan tesaurus menggunakan *constraint satisfaction problem* menghasilkan kata kata tesaurus yang lebih relevan dengan tesaurus yang dibentuk tanpa menggunakan *constraint satisfaction problem*.

2. Ada beberapa kata yang memiliki *precision* rendah dikarenakan data tersebut tidak memiliki topik serupa
3. Tesaurus yang terbentuk dari *parallel corpus* terdiri dari terjemahan dari kata inputan, sehingga pencarian dokumen dapat menghasilkan keluaran dokumen dalam bahasa yang berbeda dari kata inputan.
4. Tesaurus yang dibentuk dengan menggunakan CSP menghasilkan *precision* 91,38% sedangkan tesaurus yang dibentuk tanpa menggunakan CSP menghasilkan *precision* 45,23%.
5. Pencarian tanpa menggunakan tesaurus menghasilkan *recall* 63,33% *precision* 100% dan akurasi 63,33%
6. Pencarian menggunakan tesaurus menghasilkan *recall* 86,67% *precision* 100% dan akurasi 86,67%
7. *Backtracking* dapat menyederhanakan solusi pencarian pada graf, seperti yang terlihat pada Gambar 8.4 setelah dilakukan pencarian menggunakan *backtracking*, maka graf solusi akan menjadi Gambar 8.5, graf yang terbentuk berasal dari data contoh uji coba pada Bab 2.

Masalah dan Solusi dalam pembentukan CSP:

1. Jika user memasukan *threshold* yang terlalu besar dan C yang terlalu kecil. Maka hal tersebut tidak akan memenuhi Constraint Satisfaction
2. Untuk itu pengulangan dilakukan hanya sebanyak (C/2) +1
3. Dalam setiap perulangan himpunan kata disimpan untuk dibandingkan apabila perulangan sudah terlalu banyak
4. Selanjutnya himpunan penyelesaian itu akan dibandingkan dan diambil himpunan kata yang memiliki nilai tertingi dan diambil sebagai solusi

5.6 Uji Coba 4

Skenario uji coba 4 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 tiga kali dengan menggunakan jumlah data yang telah dibagi sebelumnya, yaitu 35 dokumen bahasa Indonesia dan 32 dokumen bahasa Inggris, 155 dokumen bahasa Indonesia dan 155 dokumen bahasa Inggris, serta 865 dokumen bahasa Indonesia dan 637 dokumen bahasa Inggris. Catat setiap waktu yang dibutuhkan pada suatu proses dalam setiap uji coba yang dilakukan. Proses-proses yang dimaksud adalah proses *document alignment*, ekstraksi kata, pembentukan informasi *co-occurrence*, Pencarian *constraint satisfaftion problem* menggunaan *backtracking*.

5.6.1 Skenario 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 35 dokumen bahasa Indonesia dan 32 dokumen bahasa Inggris.
2. Tahap selanjutnya adalah memasukkan data sebanyak yang telah ditentukan, 35 dokumen bahasa Indonesia dan 32 dokumen bahasa Inggris, 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 bahasa Inggris, ekstraksi kata bahasa Indonesia, terjemahan kata, *document alignment*, ekstraksi kata *document alignment*, pembentukan informasi *co-occurrence*, penghitungan *relevance weight*, *constraint satisfaction problem – backtracking* (per-kata)
4. Setelah program berhenti, lakukan kembali pada tahap 1 terhadap jumlah data selanjutnya, yaitu 155 dokumen

bahasa Indonesia dan 155 dokumen bahasa Inggris, serta 865 dokumen bahasa Indonesia dan 637 dokumen bahasa Inggris.

5.6.2 Hasil Uji Coba

Tabel 5.7 Hasil Waktu Eksekusi

Jumlah dokumen	Nama Proses	Waktu Eksekusi (sec)
35 dokumen bahasa Indonesia 32 dokumen bahasa Inggris	Ekstraksi Kata Bahasa Inggris	22,4
	Ekstraksi Kata Bahasa Indonesia	42,4
	Terjemahan Kata	95,6
	Document Alignment	8,7
	Ekstraksi Kata *Document Alignment*	32,8
	Pembentukan Informasi *Co-occurrence*	9,5
	Penghitungan *Relevance Weight*	5,5
	Constraint Satisfaction Problem – Backtracking (per-kata)	12,6
155 dokumen bahasa Indonesia 155 dokumen bahasa Inggris	Ekstraksi Kata Bahasa Inggris	27,5
	Ekstraksi Kata Bahasa Indonesia	124,6
	Terjemahan Kata	367,8
	Document Alignment	369,3
	Ekstraksi Kata *Document Alignment*	63,4
	Pembentukan Informasi *Co-occurrence*	23,4
	Penghitungan *Relevance Weight*	7,4
	Constraint Satisfaction Problem - Backtracking	15,6
865 dokumen bahasa Indonesia 637 dokumen Bahasa Inggris	Ekstraksi Kata Bahasa Inggris	367,7
	Ekstraksi Kata Bahasa Indonesia	500,9
	Terjemahan Kata	1.4
	Document Alignment	10.7
	Ekstraksi Kata *Document Alignment*	78,6
	Pembentukan Informasi *Co-occurrence*	25,3
	Penghitungan *Relevance Weight*	9,3
	Constraint Satisfaction Problem - Backtracking	17,5

Pada data sejumlah 35 dokumen bahasa Indonesia 32 dokumen bahasa Inggris, total waktu yang dibutuhkan untuk membangun tesaurus adalah 282,6 *second* atau 4,71 menit. Proses ekstraksi kata bahasa Inggris memerlukan waktu 22,4 *second*, ekstraksi kata bahasa Indonesia memerlukan waktu 42,4 *second*, terjemahan kata memerlukan waktu 120,6 *second*, *document alignment* memerlukan waktu 8,8 *second*, ekstraksi kata *document alignment* memerlukan waktu 60,8 *second*, pembentukan informasi *co-occurrence* memerlukan waktu 9,5 *second*, penghitungan *relevance weight* memerlukan waktu 5,5 *second*, *constraint satisfaction problem – backtracking* (per-kata) memerlukan waktu 12,6 *second*. Sedangkan data yang dihasilkan oleh tesaurus dari proses pembangunan tesaurus tersebut terdapat pada Tabel 5.2. Pada data sejumlah 35 dokumen bahasa Indonesia 32 dokumen bahasa Inggris, data yang dihasilkan adalah kata bahasa Indonesia 386, kata bahasa Inggris 324, pair dokumen 35 dan data informasi *co-occurrence* 1479. Data yang dihasilkan berbanding lurus dengan banyaknya dokumen yang digunakan.

Tabel 5.8 Jumlah Data yang Dihasilkan

Jumlah Dokumen	Nama Data	Jumlah Data
35 dokumen bahasa Indonesia 32 dokumen bahasa Inggris	Kata Bahasa Indonesia	386
	Kata Bahasa Inggris	324
	Pair Dokumen	35
	Informasi *Co-occurrence*	1530
155 dokumen bahasa Indonesia 155 dokumen bahasa Inggris	Kata Bahasa Indonesia	2206
	Kata Bahasa Inggris	2018
	Pair Dokumen	155
	Informasi *Co-occurrence*	25294
865 dokumen bahasa Indonesia 637 dokumen Bahasa Inggris	Kata Bahasa Indonesia	10.263
	Kata Bahasa Inggris	7137
	Pair Dokumen	865
	Informasi *Co-occurrence*	101688

5.6.3 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 lumayan lama adalah proses ektraksi kata dari dokumen berbahasa Indonesia karena proses tersebut terdiri dari beberapa bagian proses, seperti proses untuk memuat data, kemudian melakukan proses tokenisasi, penghapusan *stopword* dan proses *stemming*. Proses ekstraksi kata bahasa Indonesia membutuhkan waktu yang lebih lama daripada dokumen bahasa Inggris, karena pada dokumen bahasa Inggris tidak dilakukan proses stemming karena apabila dilakukan *stemming* menggunakan pustaka *porter* atau *snowball* dari nltk, kata yang dihasilkan menjadi tidak bagus dan tidak memiliki makna.
2. Proses yang membutuhkan waktu paling cepat adalah proses penghitungan *weight relevance,* karena proses tersebut hanya terdiri dari satu proses saja yaitu untuk penghitungan kata..
3. Proses *document aligment* juga membutuhkan waktu yang lama, karena harus dilakukan pengecekan dari satu dokumen ke dokumen lainnya agar didapatkan dokumen dengan tingkat kemiripan yang paling tinggi dari setiap pasang dokumen.
4. Proses terjemahan kata bahasa Inggris ke bahasa Indonesia dipengaruhi oleh kecepatan internet yang digunakan. Penggunaan pustaka urllib2 dapat mengatasi proses terjemahan tanpa limit, sedangkan penggunaan API google terjemahan dikenakan limit yaitu 1000 kata per hari.

BAB VI
KESIMPULAN

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. Banyaknya dokumen 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
2. Jumlah data mempengaruhi pembentukan tesaurus, karena apabila data semakin banyak, maka kata – kata yang terbentuk akan semakin bervariatif.
3. Data yang memiliki topik serupa akan membentuk tesaurus yang lebih baik, karena tesaurus yang terbentuk berasal dari topik yang sama, maka tesaurus yang terbentuk akan semakin relevan.
4. Tesaurus yang dibentuk dengan menggunakan CSP menghasilkan *precision* 91,38% sedangkan tesaurus yang dibentuk tanpa menggunakan CSP menghasilkan *precision* 45,23%.
5. Pencarian tanpa menggunakan tesaurus menghasilkan *recall* 63,33% *precision* 100% dan akurasi 63,33%
6. Pencarian menggunakan tesaurus menghasilkan *recall* 86,67% *precision* 100% dan akurasi 86,67%

6.2 Saran

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

1. Gunakan data yang lebih banyak agar tesaurus dapat memberikan hasil keluaran yang lebih baik.
2. Perbaiki tampilan hasil tesaurus berupa bentuk *dendogram* atau representasi dari hirarki antar kata yang digunakan untuk membangun tesaurus.
3. Mencari metode dokumen *alignment* yang lebih akurat

DAFTAR PUSTAKA

[1] D. W. Oard, "Alternative approaches for cross-language text retrieval," in *AAAI Symposium on Cross-Language Text and Speech Retrieval. American Association for Artificial Intelligence*, 1997, vol. 16.

[2] K. Wo\lk, "Noisy-parallel and comparable corpora filtering methodology for the extraction of bi-lingual equivalent data at sentence level," *ArXiv Prepr. ArXiv151004500*, 2015.

[3] N. Bel, C. H. Koster, dan M. Villegas, "Cross-lingual text categorization," in *International Conference on Theory and Practice of Digital Libraries*, 2003, hal. 126–139.

[4] C. D. Manning, P. Raghavan, H. Schütze, dan others, *Introduction to information retrieval*, vol. 1. Cambridge university press Cambridge, 2008.

[5] C. C. Yang dan K. W. Li, "An associate constraint network approach to extract multi-lingual information for crime analysis," *Decis. Support Syst.*, vol. 43, no. 4, hal. 1348–1361, 2007.

[6] C. C. Yang, C.-P. Wei, dan K. W. Li, "Cross-lingual thesaurus for multilingual knowledge management," *Decis. Support Syst.*, vol. 45, no. 3, hal. 596–605, 2008.

[7] G. Verfaillie dan T. Schiex, "Dynamic Backtracking for Dynamic Constraint Satisfaction Problems," in *In Proceedings of the ECAI-94 Workshop on Constraint Satisfaction Issues Raised by Practical Applications*, 1994, hal. 1–8.

[8] A. JACOMY, "Sigma js." [Daring]. Tersedia pada: http://sigmajs.org/. [Diakses: 04-Jun-2017].

[9] F. Z. Tala, "A study of stemming effects on information retrieval in Bahasa Indonesia," *Inst. Log. Lang. Comput. Univ. Van Amst. Neth.*, 2003.

(Halaman ini sengaja dikosongkan)

LAMPIRAN

LAMPIRAN KODE SUMBER

```
<script>
var i,
    s,
    g = {
      nodes: [],
      edges: []
    };
<?php

    $js_array = json_encode($hasilAkhir);
    echo "var dataPhp = ". $js_array . ";\n";
?>
var E = dataPhp.length;
node = [dataPhp[0]['term1']];
for(i=0 ; i<E; i++){
  for(j=0; j<node.length;j++){
    if(dataPhp[i]['term1'].trim().localeCompare(nod
e[j].trim())==0){
      break;
    }
    if(j==node.length-1){
      node.push(dataPhp[i]['term1']);
    }
  }
  for(j=0; j<node.length;j++){
    if(dataPhp[i]['term2'].trim().localeCompare(nod
e[j].trim())==0){
      break;
    }
    if(j==node.length-1){
      node.push(dataPhp[i]['term2']);
    }
  }
}

var N = node.length
```

```
35. for (i = 0; i < N; i++)
36.   g.nodes.push({
37.     id: node[i],
38.     label: node[i],
39.     x: Math.random(),
40.     y: Math.random(),
41.     size: 10,
42.     color: '#666'
43.   });
44. for (i = 0; i < E; i++)
45.   g.edges.push({
46.     id: 'edge' + i,
47.     source: dataPhp[i]['term1'].trim(),
48.     target: dataPhp[i]['term2'].trim(),
49.     size: dataPhp[i]['nilai'],
50.     type: 'curve',
51.     color: '#ccc',
52.     hover_color: '#000'
53.   });
54. s = new sigma({
55.   graph: g,
56.   renderer: {
57.     container: document.getElementById('graph-
    container'),
58.     type: 'canvas'
59.   },
60.   settings: {
61.     doubleClickEnabled: false,
62.     minEdgeSize: 0.5,
63.     maxEdgeSize: 10,
64.     enableEdgeHovering: true,
65.     edgeHoverColor: 'edge',
66.     defaultEdgeHoverColor: '#000',
67.     edgeHoverSizeRatio: 1,
68.     edgeHoverExtremities: true,
69.   }
70. });
71.
72. // Bind the events:
73. s.bind('overNode outNode clickNode doubleClickNode
    rightClickNode', function(e) {
74.   console.log(e.type, e.data.node.label, e.data.cap
    tor);
```

```
75. });
76. s.bind('overEdge outEdge clickEdge doubleClickEdge
    rightClickEdge', function(e) {
77.   console.log(e.type, e.data.edge, e.data.captor);

78. });
79. s.bind('clickStage', function(e) {
80.   console.log(e.type, e.data.captor);
81. });
82. s.bind('doubleClickStage rightClickStage', function
    (e) {
83.   console.log(e.type, e.data.captor);
84. });
85. </script>
```

Kode Sumber 8.1 Javascript sigma.js

```
1.  ...
2.  public function hasil()
3.      {
4.
5.          $kata = $_GET['nama'];
6.          $ammount = $_GET['ammount'];
7.          $threshold = $_GET['threshold'];
8.
9.          $hasilFinal = array();
10.
11.         $kata = str_replace(" ", "_", $kata);
12.         $address = 'http://localhost:5000/backTrack
    /?i='.$kata.'&c='.$ammount.'&t='.$threshold;
13.         $hasil = file_get_contents($address);
14.         $data = json_decode($hasil);
15.         $N = sizeof($data);
16.         for($i=0 ; $i<$N; $i++){
17.             $hasilSementara = $this-
    >searchNode((string)$data[$i]->id);
18.             $hasilFinal = array_merge($hasilFinal,$
    hasilSementara);
19.         }
20.         $hasilKirim = array();
21.         for($i=0 ; $i<$N; $i++){
```

```
22.             $hasilKirim[$i]['term1'] = $hasilFinal[
    $i]->term1;
23.             $hasilKirim[$i]['term2'] = $hasilFinal[
    $i]->term2;
24.             $hasilKirim[$i]['nilai'] = $data[$i]-
    >nilai;
25.         }
26.         $data['hasilAkhir'] = $hasilKirim;
27.
28.         $this->load->view('cobaNode',$data);
29.     }
30. ...
```

Kode Sumber 8.2 Controller Visualisasi Graph

LAMPIRAN TABEL

Tabel 8.1 Contoh data doc_ind

id	Kode	judul	abstrak	keyword	author	contributor1	contributor2	flag
329	40349	ANALISIS KLASIFIKASI KREDIT MENGGUNAKAN REGRESI LOGISTIK BINER DAN RADIAL BASIS FUNCTION NETWORK DI BANK 'X' CABANG KEDIRI	Kredit macet merupakan salah satu faktor penyebab terjadinya kebangkrutan pada industri perbankan. …….	Regresi Logistik Biner, Radial Basis Function Network, Credit Scoring, Akurasi, Klasifikasi	DIAPRINA, SISTYA ROSI	Dr. Suhartono, M.Sc		0
409	37937	BOOTSTRAP AGGREGATING MULTIVARIATE ADAPTIVE REGRESSION SPLINES (BAGGING MARS) UNTUK MENGKLASIFIKASIKAN RUMAH TANGGA MISKIN DI KABUPATEN JOMBANG	Kemiskinan merupakan masalah sosial utama setiap negara, terutama negara berkembang termasuk Indonesia. Fokus penelitian ini adalah pada kemiskinan di Kabupaten Jombang …..	Bagging; Jombang; Multivariate Adaptive Regression Spline; Rumah Tangga Miskin	ARLEINA, OKTIVA DHANI	DR. Bambang Widjanarko Otok, S. Si., M. Si		0

Tabel 8.1 merupakan contoh data dari tabel doc_ind dari *database*, kolom id merupakan kolom untuk menyimpan id data dan merupakan *primary key*, kolom kode merupakan kode dari dokumen tugas akhir, kolom judul berisi judul dari tugas akhir, kolom abstrak berisi abstraksi dari tugas akhir, keyword merupakan kata kunci dari tugas akhir, kolom author berisi nama dari penulis tugas akhir, kolom contributor1 merupakan nama dosen pembimbing 1 dan contributor2 merupakan dosen pembibing 2

tugas akhir, kolom flag merupakan penanda data tersebut teah diproses atau belum. Data yang terseimpan dalam table doc_ind merupakan dokumen dalam bahasa Indonesia.

Tabel 8.2 Contoh data doc_ing

id	kode	title	abstract	keyword	author	contributor1	contributor2	flag
63	37526	APPLICATION OF DECISION TREE ALGORITHM FOR TUNA FISH IMAGE CLASSIFICATION. CASE STUDY: PT. ANEKA TUNA INDONESIA	In Indonesia, the production of fish is one of the economic resources of society. …..…	co-occurance matrix, decision tree, ekstraksi, ikan tuna, pengolahan citra, segmentasi.	KALBUADI, MUHAMMAD AKBAR	Dr. AGUS ZAINAL ARIFIN, S.Kom., M.Kom.	WIJAYANTI NURUL KHOTIMAH, S.Kom.,M.Kom.	1
68	38308	APPLICATION OF TEXT CLASSIFICATION FOR MAPPING TWITTER DATA OF PT. TELEKOMUNIKASI SELULAR CUSTOMERS USING HEAT MAP	The tight competition makes companies increasing effort to gain information from various media for the need of service analysis.……	PT. Telekomunikasi selular; twitter API stream; klasifikasi teks; naive bayes; visualisasi; heat map	MAâ€™ADY, MOCHAMAD NIZAR PALEFI	Prof. Ir. Arif Djunaidy M.Sc., Ph.D.	Renny Pradina Kusumawardani S.T., M.T.	1

Tabel 8.2 merupakan contoh data dari tabel doc_ing dari *database*, kolom id merupakan kolom untuk menyimpan id data dan merupakan *primary key*, kolom kode merupakan kode dari dokumen tugas akhir, kolom title berisi judul dari tugas akhir, kolom abstract berisi abstraksi dari tugas akhir, keyword merupakan kata kunci dari tugas akhir, kolom author berisi nama dari penulis tugas akhir, kolom contributor1 merupakan nama dosen pembimbing 1 dan contributor2 merupakan dosen pembibing 2 tugas akhir, kolom flag merupakan penanda data tersebut teah diproses atau belum. Data yang terseimpan dalam table doc_ind merupakan dokumen dalam bahasa Inggris.

Tabel 8.3 Contoh data term_ind

id_term	id_dokumen	term	tf
1	329	analisis	1
2	329	bank	1
3	329	basis	1
4	329	biner	1
5	329	cabang	1
6	329	diri	1
7	329	function	1
8	329	guna	1
10	329	kredit	1
11	329	logistik	1
12	329	network	1
13	329	radial	1
14	329	regresi	1

Tabel 8.3 merupakan contoh dari tabel term_ind, tabel ini berisi hasil dari ekstraksi kata judul dari tabel doc_ind. Kolom id_term merupakan *primary key*, id_dokumen merupakan id dokumen dimana *term* terbentuk dari table doc_ind, tf merupakan nilai *term frequency* dari kolom *term*.

Tabel 8.4 Contoh data term_ing

id_term	id	term	t0	t1	t2	t3	t4	t5	t6	t7	flag
1	63	algorithm	algoritma	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0
2	63	aneka	aneka	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0
3	63	application	aplikasi	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0
4	63	case	kasus	orang	sakit	adu	tas	casing	government	tempat	2
5	63	classification	klasifikasi	NULL	NULL	NULL	NULL	NULL	NULL	NULL	0
6	63	decision	keputusan	kalah	angka	box	the	decisioned	kkt	nyata	2

Tabel 8.4 merupakan contoh dari tabel term_ing, tabel ini berisi hasil dari ekstraksi kata judul dari tabel doc_ing. Kolom id_term merupakan *primary key*, id_dokumen merupakan id dokumen dimana *term* terbentuk dari table doc_ind, t0 merupakan hasil terjemahan *term* mnggunakan google terjemahan ke dalam bahasa Indonesia, kolom t1-t7 merupakan hasil terjemhan *term* menggunakan *database* kamus. Kolom flag merupakan penanda data telah diproses atau belum.

Tabel 8.5 Contoh data coocurrence

id_co	term1	term2	tf	idf	relevance _weight	relevance _weight2
1	activity	cart	1	3.59	1.217243	0.790181
2	activity	classification	1	3.59	0.084028	1.259169
3	activity	dekat	1	3.59	1.551296	1.551296
4	activity	economic	1	3.59	1.551296	1.551296
5	activity	ekonomi	1	3.59	1.551296	1.551296
6	activity	java	1	3.59	1.217243	1.146067
7	activity	jawa	1	3.59	1.217243	1.146067

Tabel 8.5 merupakan contoh dari tabel coocurrence, tabel ini berisi hasil dari proses pembentukan informasi *co-occurrence*. id_co merupakan *primary key*, term1 merupakan term1 dari pasangan kata, term2 merupakan pasangan kata dari ter1, tf merupakan nilai TF dari pasangan kata, idf merupakan nilai IDF dari pasangan kata, relevance_weight adalah nilai *relevance weight* term1 terhadap term2, relevance_weight2 adalah nilai *relevance weight* term2 terhadap term1.

Tabel 8.6 Contoh data term_dist

id_termd	term	tf	df	tfidf	weight_factor	idf	flag
1	activity	1	1	1.39794	0.434294	1.39794	0
2	acute	2	1	2.79588	0.434294	1.39794	0
3	adaptation	2	1	2.79588	0.434294	1.39794	0
4	adaptive	4	2	4.38764	0.340774	1.09691	0
5	aged	1	1	1.39794	0.434294	1.39794	0
6	aggregating	2	1	2.79588	0.434294	1.39794	0

Tabel 8.6 merupakan contoh data dari term_dist, tabel ini berisi data hasil perhitungan dari kolom term yang telah di-*distinct*. id_termd merupakan *primary key*. Kolom term merupakan *term distinct* dari tabel term, kolom TF merupakan nilai TF dari term, df adalah nilai DF dari *term*, tfidf adalah nilai TFIDF dari *term*, weight_factor merupakan nilai *weighting factor* dari *term*, idf adalah nilai IDF dari *term*, flag merupakan penanda data tersebut telah diproses atau belum.

Tabel 8.7 Contoh data pair_doc

id	id_ind	id_ing	score	flag
1	329	119	0.733333	0
2	409	91	0.6	0
3	412	397	0.5	0
4	454	603	0.6	0
5	535	358	0.545455	0
6	539	351	0.304348	0
7	542	361	0.4375	0
8	545	368	0.230769	0
9	580	604	0.333333	0
10	584	455	0.352941	0
11	585	85	0.304348	0

Tabel 8.7 merupakan contoh data dari pair_doc, tabel ini berisi data hasil pembentukan *parallel corpus*. id merupakan *primary key*. Kolom id_ind merupakan id dokumen bahasa Indonesia, kolom id_ing merupakan id dokumen bahasa Inggris yang merupakan pasangan dokumen dari dokumen pada kolom id_ind , score adalah nilai perhitungan *similarity* antara id_ind dan id_ing, flag merupakan penanda data tersebut telah diproses atau belum.

Tabel 8.8 Contoh data terms

id_term	id_pair	term	tf
1	1	analisis	1
2	1	analysis	1
3	1	bank	2
4	1	basis	2
5	1	binary	1
6	1	biner	1

Tabel 8.8 merupakan hasil ekstrasi term dari parallel corpus yang terbentuk. Id_term merupakan primary key, id_pair merupakan id

pasangan dimana *term* tersebut berasal, TF merupakan nilai TF dari *term*.

Tabel 8.9 Perbandingan Pencarian

Term	Non Tesaurus	Tesaurus
Text	Application Of Text Classification For Mapping Twitter Data Of Pt. Telekomunikasi Selular Customers Using Heat Map	Application Of Text Classification For Mapping Twitter Data Of Pt. Telekomunikasi Selular Customers Using Heat Map
	Class Feature Centroid Classifier Method For Classifying Imbalanced Arabic Text	Class Feature Centroid Classifier Method For Classifying Imbalanced Arabic Text
		Metode Class Feature Centroid Classifier Untuk Mengklasifikasi Data Teks Arab Yang Tidak Seimbang
		Penerapan Klasifikasi Teks Untuk Pemetaan Data Twitter Pelanggan Pt. Telekomunikasi Selular Dalam Bentuk Heat Map
Signal	The Utilization Nonortogonal Basis Of Covariance Matrix Adaptation Evolution Strategy For Electroencephalograph Signal Classification Based Brain Computer Interface	The Utilization Nonortogonal Basis Of Covariance Matrix Adaptation Evolution Strategy For Electroencephalograph Signal Classification Based Brain Computer Interface
		Penerapan Basis Nonortogonal Pada Covariance Matrix Adaptation-Evolution Strategy Untuk Klasifikasi Sinyal Elctroencephalograph Berdasarkan Brain Computer Interface
Retinal	Implementation Of Retinal Vascular Segment Based Classifier Using Multivariate M-Mediods	Implementation Of Retinal Vascular Segment Based Classifier Using Multivariate M-Mediods

Term	Non Tesaurus	Tesaurus
		Implementasi Pengklasifikasi Segmen Vaskular Retina Mata Dengan Metode Mmediods Multivariat
Disease	Implementation Of Neural Network With Weighted Fuzzy Membership Study Case Classification Parkinson Disease	Implementation Of Neural Network With Weighted Fuzzy Membership Study Case Classification Parkinson Disease
		Implementasi Neural Network With Weighted Fuzzy Membership Studi Kasus Klasifikasi Penyakit Parkinson

Tabel 8.9 merupakan perbandingan pencarian menggunakan pencarian biasa dan pencarian menggunakan tesaurus, kolom term merupakan kata kunci pencarian, kolom non tesaurus adalah dokumen yang dihasilkan dari pencarian dokumen dari kata kunci tanpa menggunakan tesaurus, dan kolom tesaurus merupakan dokumen yang dihasilkan dari pencarian dokumen dari kata kunci yang diekspansi menggunakan tesaurus.

Tabel 8.10 Contoh groundtruth tesaurus

No	Keyword	Tesaurus	Kecocokan	Kecocokan	Rata-rata
1	text	twitter	1	1	
		arabic	1	1	
		arab	1	1	
		centroid	1	0	
			100	75	87.5
2	stemmer	stripping	1	0	
		articles	1	1	
		artikel	1	1	
		bahasa	1	1	
			100	75	87.5
3	brain	eeg	1	1	
		covariance	1	1	
		computer	1	1	
		evolution	1	1	
			100	100	100

Tabel 8.11 Contoh Kuisioner 1

Responden	Berapa jumlah data dari basis data yang dilakukan uji coba yang memiliki hasil terbaik?	Apakah jumlah data mempengaruhi pembentukan tesaurus?	Apakah data yang memiliki topik serupa dapat membentuk tesaurus yang lebih baik?
Faris Ponighzwa	D1	Ya	Ya
Asri Mulyasari	D1	Ya	Ya
Andi putra	D1	Ya	Ya
Renanda Agustiantoro	D1	Ya	Ya
Annisa Rizki	D1	Ya	Ya
Herni Anggi	D1	Ya	Ya
Kevin Z	D1	Ya	Ya

Tabel 8.12 Contoh Kuisioner 2

Responden	Apakah pencarian yang menggunakan tesaurus lebih relevan?	Apakah tesaurus yang dibentuk menggunakan CSP lebih baik daripada non CSP?
Faris Ponighzwa	Ya	Ya
Asri Mulyasari	Ya	Ya
Andi putra	Ya	Ya
Renanda Agustiantoro	Ya	Ya
Annisa Rizki	Ya	Ya
Herni Anggi	Ya	Ya
Kevin Z	Ya	Ya
Putu	Ya	Ya
Ayu Pangesti	Ya	Ya
Nidar Luffi	Ya	Ya

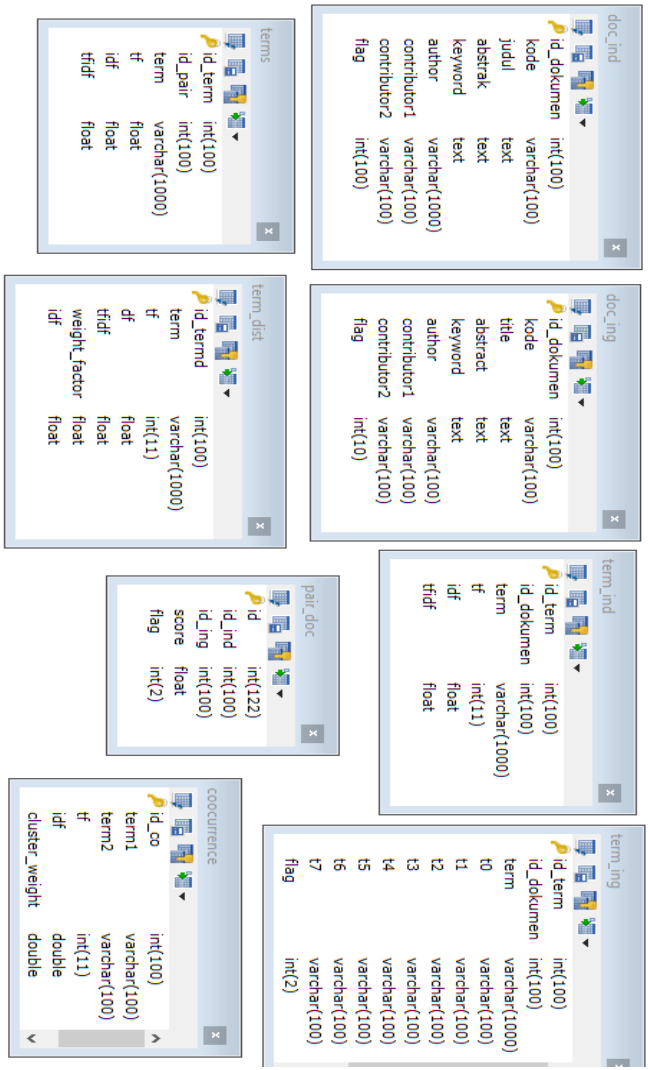
Tabel 8.13 Contoh groundtruth pencarian dokumen

No	Keyword	Hasil yang Ditampilkan	Ground truth	recall	precision	akurasi
1	Text	APPLICATION OF TEXT CLASSIFICATION FOR MAPPING TWITTER DATA OF PT. TELEKOMUNIKASI SELULAR CUSTOMERS USING HEAT MAP	APPLICATION OF TEXT CLASSIFICATION FOR MAPPING TWITTER DATA OF PT. TELEKOMUNIKASI SELULAR CUSTOMERS USING HEAT MAP	100	100	100
		CLASS FEATURE CENTROID CLASSIFIER METHOD FOR CLASSIFYING IMBALANCED ARABIC TEXT	CLASS FEATURE CENTROID CLASSIFIER METHOD FOR CLASSIFYING IMBALANCED ARABIC TEXT			
		METODE CLASS FEATURE CENTROID CLASSIFIER UNTUK MENGKLASIFIKASI DATA TEKS ARAB YANG TIDAK SEIMBANG	METODE CLASS FEATURE CENTROID CLASSIFIER UNTUK MENGKLASIFIKASI DATA TEKS ARAB YANG TIDAK SEIMBANG			

No	Keyword	Hasil yang Ditampilkan	Ground truth	recall	precision	akurasi
		PENERAPAN KLASIFIKASI TEKS UNTUK PEMETAAN DATA TWITTER PELANGGAN PT. TELEKOMUNIKASI SELULAR DALAM BENTUK HEAT MAP	PENERAPAN KLASIFIKASI TEKS UNTUK PEMETAAN DATA TWITTER PELANGGAN PT. TELEKOMUNIKASI SELULAR DALAM BENTUK HEAT MAP			
2	Stemmer	KLASIFIKASI ARTIKEL BERITA BERBAHASA INDONESIA BERBASIS NAÏVE BAYES CLASSIFIER MENGGUNAKAN CONFIX-STRIPPING STEMMER	KLASIFIKASI ARTIKEL BERITA BERBAHASA INDONESIA BERBASIS NAÏVE BAYES CLASSIFIER MENGGUNAKAN CONFIX-STRIPPING STEMMER	100	100	100
		NEWS ARTICLES CLASSIFICATION IN INDONESIAN LANGUAGE BASED ON NAIVE BAYES CLASSIFIER USING CONFIX-STRIPPING STEMMER	NEWS ARTICLES CLASSIFICATION IN INDONESIAN LANGUAGE BASED ON NAIVE BAYES CLASSIFIER USING CONFIX-STRIPPING STEMMER			

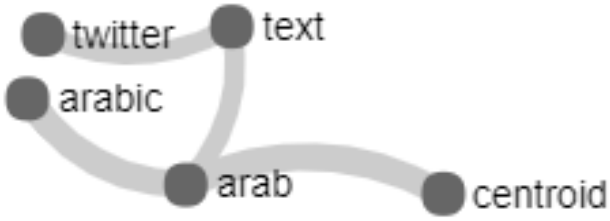
No	Keyword	Hasil yang Ditampilkan	Ground truth	recall	precision	akurasi
3	Signal	PENERAPAN BASIS NONORTOGONAL PADA COVARIANCE MATRIX ADAPTATION-EVOLUTION STRATEGY UNTUK KLASIFIKASI SINYAL ELCTROENCEPHALOGRAPH BERDASARKAN BRAIN COMPUTER INTERFACE	PENERAPAN BASIS NONORTOGONAL PADA COVARIANCE MATRIX ADAPTATION-EVOLUTION STRATEGY UNTUK KLASIFIKASI SINYAL ELCTROENCEPHALOGRAPH BERDASARKAN BRAIN COMPUTER INTERFACE	100	100	100
		THE UTILIZATION NONORTOGONAL BASIS OF COVARIANCE MATRIX ADAPTATION EVOLUTION STRATEGY FOR ELECTROENCEPHALOGRAPH SIGNAL CLASSIFICATION BASED BRAIN COMPUTER INTERFAC	THE UTILIZATION NONORTOGONAL BASIS OF COVARIANCE MATRIX ADAPTATION EVOLUTION STRATEGY FOR ELECTROENCEPHALOGRAPH SIGNAL CLASSIFICATION BASED BRAIN COMPUTER INTERFAC			

LAMPIRAN GAMBAR



Gambar 8.1 Physical Data Mode

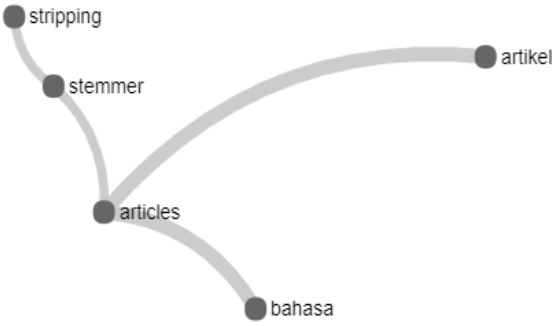
Keyword = text



Gambar 8.2 Tesaurus text

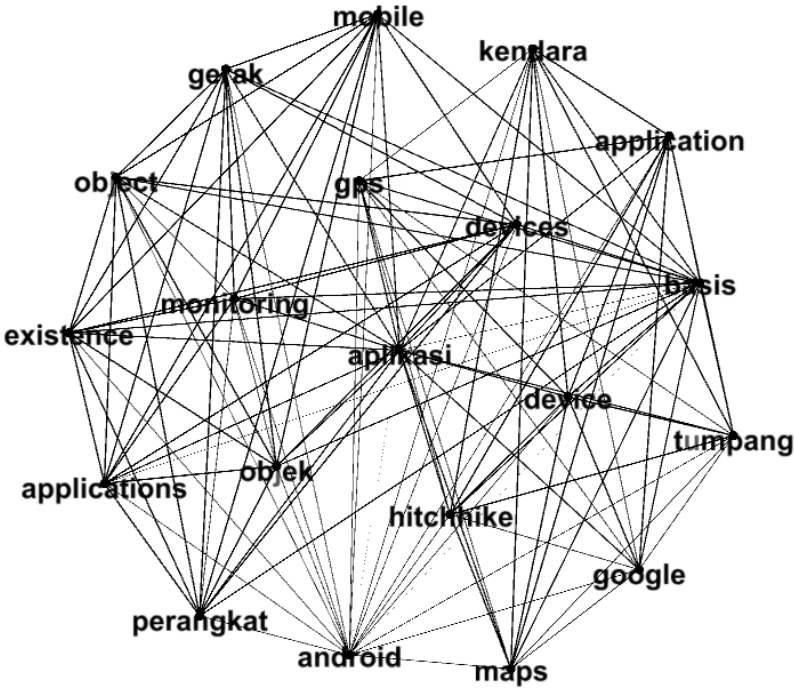
Gambar 8.2 merupakan graf tesaurus yang terbentuk dari kata text.

Keyword = stemmer



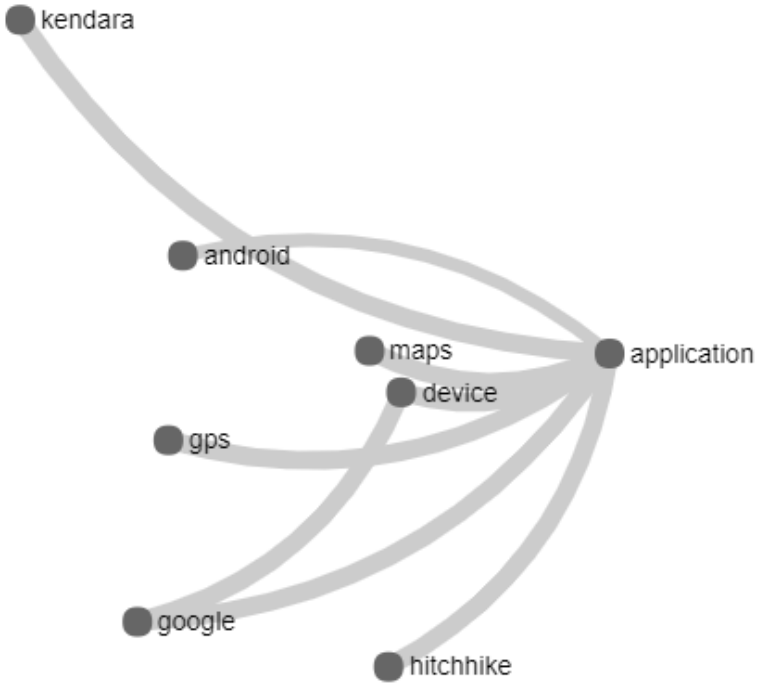
Gambar 8.3 Tesaurus stemmer

Gambar 8.3 merupakan graf tesaurus yang terbentuk dari kata stemmer.



Gambar 8.4 Graf Co-occurrence

Gambar 8.4 merupakan graf yang terbentuk dari pembentukan informasi *co-occurrence* yang berasal dari contoh kasus yang digunakan pada Bab 2.



Gambar 8.5 Graf CSP

Gambar 8.5 merupakan contoh pencarian CSP dengan input "device android" dengan nilai *threshold* C (jumlah kata) = 8 dan *threshold* T (jumlah *relevance weight*) = 18. Gambar 8.4 merupakan graf yang terbentuk dari pembentukan informasi *co-occurrence* yang berasal dari contoh kasus yang digunakan pada Bab 2.

(Halaman ini sengaja dikosongkan)

BIODATA PENULIS

Umy Chasanah Noor Rizqi, Lahir pada tanggal 09 Desember 1996. Anak kedua 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 2013. Pernah mengikuti beberapa organisasi dan beberapa kepanitian diantaranya adalah Staf Hubungan Luar Himpunan Mahasiswa Teknik Computer-Informatika 2014-2015, Staff Web dan Kesektretariatan SCHEMATICS 2014 dan SCHEMATICS 2015, Staff Kesektretariatan ITS EXPO 2014 dan ITS EXPO 2015. **umy.rizqi@gmail.com**