IMPLEMENTASI MODIFIKASI ENHANCED CONFIX STRIPPING STEMMER UNTUK BAHASA INDONESIA DENGAN METODE CORPUS BASED STEMMING

Andita Dwiyoga Tahitoe - Diana Purwitasari

Jurusan Teknik Informatika, Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember (ITS) – Surabaya, 60111, Indonesia

email: anditadt06@cs.its.ac.id

***Abstract* — Algoritma stemming kata pada Bahasa Indonesia dengan performa yang paling baik (memiliki jenis kesalahan stemming yang paling sedikit) adalah algoritma Enhanced Confix Stripping (ECS) Stemmer. Meskipun terdapat peningkatan performa stemming kata, masih terdapat kesalahan yang dilakukan oleh ECS Stemmer. Selain itu, algoritma ECS Stemmer juga tidak mengajukan perbaikan terhadap permasalahan overstemming dan understemming.**

Dalam tugas akhir ini, diajukan perbaikan terhadap algoritma ECS Stemmer. Selain perbaikan terhadap aturan pada tabel acuan pemenggalan imbuhan, juga dilakukan implementasi metode corpus based stemming untuk melakukan penyelesaian terhadap problem overstemming dan understemming.

Proses evaluasi sistem Information Retrieval (IR) menggunakan relevansi set dokumen terhadap query yang dibentuk secara otomatis menggunakan teknik data fusion dan metode condorcet. Karena tidak dibentuk secara manual, relevan set tersebut dinamakan pseudo relevant documents (pseudorels).

***Index Terms* — Enhanced Confix Stripping Stemmer, Confix Stripping Stemmer, corpus based stemming, data fusion, metode condorcet, pseudo relevant documents (pseudorels)**

I. PENDAHULUAN

TEKNIK *stemming* adalah suatu teknik pencarian bentuk dasar dari suatu *term*. Yang dimaksud dengan *term* itu sendiri adalah tiap kata yang berada pada suatu dokumen teks. *Stemming* dilakukan pada saat pembuatan indeks dari suatu dokumen. Pembuatan indeks dilakukan karena suatu dokumen tidak dapat dikenali langsung oleh suatu sistem temu kembali informasi atau *information retrieval (IR)* system. Oleh karena itu, dokumen tersebut terlebih dahulu perlu dipetakan ke dalam suatu representasi dengan menggunakan teks yang berada di dalamnya. Teknik *stemming* diperlukan selain untuk memperkecil jumlah indeks yang berbeda dari suatu dokumen, juga untuk melakukan pengelompokan kata-kata lain yang memiliki kata dasar dan arti yang serupa namun memiliki bentuk atau *form* yang berbeda karena mendapatkan imbuhan yang berbeda.

Teknik *stemming* terdiri dari berbagai macam metode. Metode pertama yakni *stemming* dengan acuan tabel pemenggalan imbuhan. Proses *stemming* suatu *term* dengan metode ini dilakukan dengan cara menghilangkan imbuhan dari *term* tersebut sesuai dengan tabel acuan pemenggalan imbuhan yang digunakan. Metode kedua merupakan pengembangan dari metode pertama. Metode kedua ini selain menggunakan tabel acuan pemenggalan imbuhan, juga menggunakan suatu kamus kata dasar. Kamus kata dasar ini digunakan sebagai acuan hasil *stemming* saat proses pemenggalan imbuhan selesai dilakukan. Hasil dari proses *stemming* dengan metode ini harus ada pada kamus kata dasar, jika tidak maka *term* yang diinputkan dianggap sebagai bentuk dasar. Metode ketiga dinamakan metode *stemming* berbasis *corpus* [6] (koleksi dokumen) karena hasil *stemming* menggunakan metode ini dipengaruhi oleh koleksi dokumen yang digunakan dalam proses uji coba. Kelas stem yang terbentuk dipengaruhi oleh nilai statistik *co-occurence* dari tiap *term* pada kelas stem tersebut. Metode ini dikembangkan dari hipotesis awal bahwa dua buah *term* dengan bentuk dasar yang sama akan sering muncul pada koleksi dokumen yang digunakan pada uji coba. Nilai keseringan muncul secara bersamaan inilah yang dihitung menggunakan statistik *co-occurence*.

Metode ketiga dilatarbelakangi dari masalah *overstemming* dan *understemming*. Inti dari masalah tersebut yakni kemungkinan hasil *stemming* yang dapat berjumlah lebih dari satu. Kemungkinan hasil *stemming* yang lebih dari satu ini diakibatkan oleh algoritma *stemming* yang digunakan. Teknik *hard stemming*, *stemming* dilakukan hingga seluruh imbuhan berhasil dihilangkan, tentunya akan memiliki hasil *stemming* yang berbeda dengan teknik *soft stemming*, proses penghilangan imbuhan langsung dihentikan saat kata dasar dari *term* tersebut ditemukan. Selain itu, ambiguitas pada suatu bahasa juga dapat menyebakan hasil *stemming* memiliki kemungkinan berjumlah lebih dari satu.

Algoritma *stemming* kata pada Bahasa Indonesia dengan performa yang paling baik (memiliki jenis kesalahan *stemming* yang paling sedikit) adalah algoritma *Enhanced Confix Stripping (ECS) Stemmer* [2]. Algoritma *ECS Stemmer* ini merupakan algoritma perbaikan dari algoritma *Confix Stripping (CS) Stemmer*. Perbaikan yang dilakukan oleh *ECS Stemmer* adalah perbaikan beberapa aturan pada tabel acuan pemenggalan imbuhan. Selain itu, algoritma *ECS Stemmer*

juga menambahkan langkah pengembalian akhiran jika terjadi penghilangan akhiran yang seharusnya tidak dilakukan.

Meskipun terdapat peningkatan performa (peningkatan keberhasilan melakukan *stemming* kata), masih terdapat kesalahan *stemming* kata yang dilakukan oleh *ECS Stemmer*. Selain itu, algoritma *ECS Stemmer* juga tidak mengajukan perbaikan terhadap permasalahan *overstemming* dan *understemming*. Oleh sebab-sebab itulah dalam Tugas Akhir ini, dilakukan diajukan perbaikan terhadap algoritma *ECS Stemmer*. Selain perbaikan terhadap aturan pada tabel acuan pemenggalan imbuhan, juga dilakukan implementasi metode *stemming* berbasis *corpus* untuk melakukan penyelesaian terhadap problem *overstemming* dan *undertstemming*.

Evaluasi hasil *stemming* dilakukan secara manual dengan melakukan pengamatan secara langsung terhadap hasil *stemming*. Untuk menilai apakah hasil *stemming* yang dilakukan benar atau salah, digunakan Kamus Besar Bahasa Indonesia (KBBI). KBBI berbeda dengan kamus kata dasar yang digunakan sebagai acuan proses *stemming*. Pada KBBI, setiap kata yang terdapat di dalamnya tidak hanya berupa kata dasar. Selain kata dasar, pada KBBI juga disertakan berbagai variasi bentuk kata dasar tersebut dengan berbagai macam imbuhan.

Selain melakukan evaluasi terhadap hasil *stemming*, juga dilakukan evaluasi terhadap sistem *IR*. Sistem *IR* yang digunakan di dalam uji coba adalah suatu sistem pencarian dokumen berdasarkan input *query* dari user. Evaluasi dilakukan terhadap nilai efektifitas sistem *IR* yang menggunakan algoritma *ECS Stemmer* sebelum dan sesudah perbaikan.

Untuk melakukan proses evaluasi sistem *IR* dibutuhkan beberapa buah set. *Dokumen set* yang berisi dokumen-dokumen yang akan digunakan dalam uji coba. *Query set* yang berisi daftar *query* yang akan digunakan dalam proses pencarian dokumen. Serta yang terakhir yakni *relevan set* dokumen terhadap *query* yang berisi daftar dokumen-dokumen yang dinilai relevan untuk tiap *query* pada *query set*.

Pembuatan *relevan set* membutuhkan penilaian secara manual oleh manusia untuk menilai apakah suatu dokumen mengandung informasi yang dibutuhkan sesuai *input query* yang dimasukkan. Hal inilah yang membedakan *query* informasi dengan *query database*. Pada *query* informasi, selain *term* pada *query* terdapat pada dokumen, dokumen tersebut dinilai relevan jika informasi yang dikehendaki untuk diketahui dari *query* terdapat pada dokumen tersebut. Sedangkan, proses *query database* hanyalah mencari dokumen-dokumen yang mengandung *term-term* pada *query* yang di-*input*-kan.

Penilaian relevansi menimbulkan beberapa masalah. Masalah pertama yakni terkadang muncul perbedaan penilaian relevan atau tidaknya suatu dokumen terhadap *query* jika penilaian dilakukan oleh lebih dari satu orang. Masalah kedua adalah banyaknya waktu yang dibutuhkan jika koleksi dokumen yang digunakan dalam uji coba jumlahnya sangat banyak.

Permasalahan pembuatan *relevansi set* secara manual mendorong dikembangkannya proses pembuatan *relevansi set* secara otomatis. Pembuatan *relevansi set* secara otomatis dilakukan menggunakan teknik *data fusion* dan *metode condorcet* [5]. Teknik *data fusion* bekerja dengan menggabungkan menjadi satu *top-N* dokumen hasil pencarian oleh beberapa buah sistem terhadap suatu *query*. Setelah dilakukan penggabungan, dilakukan pemberian rangking terhadap tiap dokumen pada hasil penggabungan menggunakan *metode condorcet*. Setelah rangking diberikan, dokumen-dokumen yang memiliki *rank* pada sekian % dari total penggabungan dokumen ditetapkan sebagai *relevan set* dokumen terhadap *query* atau dapat disebut sebagai *pseudo relevant documents (pseudorels)*.

II. Teknik Stemming Pada Bahasa Indonesia

A. Algoritma Stemming Nazief dan Adriani

Algoritma *stemming* Nazief dan Adriani (1996) dikembangkan berdasarkan aturan morfologi Bahasa Indonesia yang mengelompokkan imbuhan menjadi awalan (*prefix*), sisipan (*infix*), akhiran (*suffix*) dan gabungan awalan-akhiran (*confixes*). Algoritma ini menggunakan kamus kata dasar dan mendukung *recoding*, yakni penyusunan kembali kata-kata yang mengalami proses *stemming* berlebih.

Aturan *morfologi* Bahasa Indonesia mengelompokkan imbuhan ke dalam beberapa kategori sebagai berikut :

1) *Inflection suffixes* yakni kelompok akhiran yang tidak merubah bentuk kata dasar. Sebagai contoh, kata “duduk” yang diberikan akhiran “-lah” akan menjadi “duduklah”. Kelompok ini dapat dibagi menjadi dua :
   a. *Particle* (P) atau partikel, yakni termasuk di dalamnya “-lah”, “-kah”, “-tah”, dan “-pun”.
   b. *Possessive Pronoun* (PP) atau kata ganti kepunyaan, termasuk di dalamnya adalah “-ku” , “-mu”, dan “-nya”.
2) *Derivation Suffixes* (DS) yakni kumpulan akhiran asli Bahasa Indonesia yang secara langsung ditambahkan pada kata dasar yaitu akhiran “-i”, “-kan”, dan “-an”.
3) *Derivation Prefixes* (DP) yakni kumpulan awalan yang dapat langsung diberikan pada kata dasar murni, atau pada kata dasar yang sudah mendapatkan penambahan sampai dengan 2 awalan. Termasuk di dalamnya adalah :
   a. Awalan yang dapat ber*morfologi* (“me-”, “be-”, “pe-”, dan “te-”)
   b. Awalan yang tidak ber*morfologi* (“di-”, “ke-” dan “se-”).

Berdasarkan pengklasifikasian imbuhan-imbuhan di atas, maka bentuk kata berimbuhan dalam Bahasa Indonesia dapat dimodelkan sebagai berikut :

[DP+ [DP+ [DP+]]] Kata Dasar [[+DS] [+PP] [+P]]

Dengan model Bahasa Indonesia di atas serta aturan-aturan dasar *morfologi* Bahasa Indonesia, aturan yang dipergunakan dalam proses *stemming* algoritma Nazief-Adriani sebagai berikut :

1) Tidak semua kombinasi awalan dan akhiran diperbolehkan. Kombinasi-kombinasi imbuhan yang tidak diperbolehkan, yaitu ‘be-i’, ‘di-an’, ‘ke-i’, ‘ke-kan’, ‘me-an’, ‘se-i’, ‘se-kan’, dan yang terakhir ‘te-an’.

TABEL 2.1
ATURAN PEMENGGALAN AWALAN *STEMMER* NAZIEF DAN ADRIANI

Aturan	Format Kata	Pemenggalan
1	berV...	ber-V... | be-rV...
2	berCAP...	ber-CAP... dimana C!='r' & P!='er'
3	berCAerV...	ber-CaerV... dimana C!='r'
4	belajar	bel-ajar
5	beC1erC2...	be-C1erC2... dimana C1!={'r'|'l'}
6	terV...	ter-V... | te-rV...
7	terCerV...	ter-CerV... dimana C!='r'
8	terCP...	ter-CP... dimana C!='r' dan P!='er'
9	teC_1erC_2...	te-C_1erC_2... dimana C_1!='r'
10	me{l|r|w|y}V...	me-{l|r|w|y}V...
11	mem{b|f|v}...	mem-{b|f|v}...
12	mempe{r|l}...	mem-pe...
13	mem{rV|V}...	me-m{rV|V}... | me-p{rV|V}...
14	men{c|d|j|z}...	men-{c|d|j|z}...
15	menV...	me-nV... | me-tV
16	meng{g|h|q}...	meng-{g|h|q}...
17	mengV...	meng-V... | meng-kV...
18	menyV...	meny-sV…
19	mempV...	mem-pV... dimana V!='e'
20	pe{w|y}V...	pe-{w|y}V...
21	perV...	per-V... | pe-rV...
23	perCAP	per-CAP... dimana C!='r' dan P!='er'
24	perCAerV...	per-CAerV... dimana C!='r'
25	pem{b|f|V}...	pem-{b|f|V}...
26	pem{rV|V}...	pe-m{rV|V}... | pe-p{rV|V}...
27	pen{c|d|j|z}...	pen-{c|d|j|z}...
28	penV...	pe-nV... | pe-tV...
29	peng{g|h|q}...	peng-{g|h|q}...
30	pengV...	peng-V... | peng-kV...
31	penyV...	peny-sV…
32	pelV...	pe-lV... kecuali "pelajar" yang menghasilkan "ajar"
33	peCerV...	per-erV... dimana C!={r|w|y|l|m|n}
34	peCP...	pe-CP... dimana C!={r|w|y|l|m|n} dan P!='er'

Keterangan simbol huruf :
C : huruf konsonan
V : huruf vokal
A : huruf vokal atau konsonan
P : partikel atau fragmen dari suatu kata, misalnya "er"

TABEL 2.2
MODIFIKASI DAN TAMBAHAN ATURAN PADA TABEL 2.1
OLEH ALGORITMA *CS STEMMER*

Aturan	Format Kata	Pemenggalan
12	mempe...	mem-pe...
16	meng{g|h|q|k}...	meng-{g|h|q|k}...
35	$terC_1erC_2$...	ter-C_1erC_2... dimana C_1!= 'r'
36	peC_1erC_2...	pe-C_1erC_2... dimana C_1!={r|w|y|l|m|n}

TABEL 2.3
MODIFIKASI ATURAN UNTUK TABEL 2.1
OLEH ALGORITMA *ECS STEMMER*

Aturan	Format Kata	Pemenggalan
14	men{c|d|j|s|z}...	men-{c|d|j|s|z}...
17	mengV...	meng-V... | meng-kV... | (mengV-... jika V='e')
19	mempA...	mem-pA... dengan A!='e'
29	pengC...	peng-C...
30	pengV...	peng-V... | peng-kV... | (pengV-... jika V='e')

2) Penggunaan imbuhan yang sama secara berulang tidak diperkenankan.
3) Jika suatu kata hanya terdiri dari satu atau dua huruf, maka proses *stemming* tidak dilakukan.
4) Penambahan suatu awalan tertentu dapat mengubah bentuk asli kata dasar, ataupun awalan yang telah diberikan sebelumnya pada kata dasar bersangkutan (ber*morfologi*). Sebagai contoh, awalan "me-" dapat berubah menjadi "meng- ", "men-", "meny-", dan "mem-". Oleh karena itu, diperlukan suatu aturan yang mampu mengatasi masalah *morfologi* ini.

Algoritma *stemmer* yang diperkenalkan Nazief dan Adriani didefinisikan sebagai berikut :

1) Di awal proses stemming dan setiap langkah yang selanjutnya dilakukan, lakukan pengecekan hasil proses *stemming* kata yang di-*input*-kan pada langkah tersebut ke kamus kata dasar. Jika kata ditemukan, berarti kata tersebut sudah berbentuk kata dasar dan proses *stemming* dihentikan. Jika tidak ditemukan, maka langkah selanjutnya dilakukan.
2) Hilangkan *inflectional suffixes*. Dimulai dari *inflectional particle*, kemudian *possessive pronoun*.
3) Hilangkan *derivation suffixes*.
4) Hilangkan *derivation prefixes*.
   a. Langkah 4 berhenti jika :
      i. Terjadi kombinasi awalan dan akhiran yang terlarang.
      ii. Awalan yang dideteksi saat ini sama dengan awalan yang dihilangkan sebelumnya.
      iii. Tiga awalan telah dihilangkan.
   b. Identifikasikan tipe awalan dan hilangkan. Awalan terdiri dari dua tipe :
      i. Standar ("di-", "ke-", "se-") yang dapat langsung dihilangkan dari kata.
      ii. Kompleks ("me-", "be-", "pe", "te-") adalah tipe-tipe awalan yang dapat bermorfologi sesuai kata dasar yang mengikutinya. Oleh karena itu, gunakan aturan pada Tabel 2.1 untuk mendapatkan hasil pemenggalan yang tepat.
   c. Cari kata yang telah dihilangkan awalannya ini di dalam kamus kata dasar. Apabila tidak ditemukan, maka langkah 4 diulangi kembali. Apabila ditemukan, maka keseluruhan proses dihentikan.
5) Apabila setelah langkah 4 kata dasar masih belum ditemukan, maka proses *recoding* dilakukan dengan mengacu pada aturan pada Tabel 2.1. *Recoding* dilakukan dengan menambahkan karakter *recoding* di awal kata yang dipenggal. Pada Tabel 2.1, karakter *recoding* adalah huruf kecil setelah tanda hubung ('-') dan terkadang berada sebelum tanda kurung. Sebagai contoh, kata "menangkap" (aturan 15), setelah dipenggal menjadi "nangkap". Karena tidak *valid*, maka *recoding* dilakukan dan menghasilkan kata "tangkap".

Catatan :

Disini ditemukan kejanggalan pada aturan pemenggalan awalan pada Tabel 2.1, dimana tidak tercantum aturan ke-22. Hingga tulisan ini selesai dibuat, belum ada konfirmasi atas kekurangan ini.

6) Jika semua langkah gagal, maka input kata yang diuji pada algoritma ini dianggap sebagai kata dasar.

B. Algoritma Confix Stripping Stemmer

Algoritma *Confix Stripping (CS) Stemmer* dikembangkan oleh Jelita Asian [3] dengan referensi dari algoritma *stemming* Nazief-Adriani dan Arifin-Setiono [1], untuk memperbaiki kesalahan-kesalahan *stemming* yang masih dilakukan. Kesalahan-kesalahan tersebut adalah sebagai berikut :

1) Tidak terdapat aturan pemenggalan awalan untuk kata-kata dengan format "mempeng...", "mengk...", "terC_1erC_2...", "peC_1erC_2...", misalnya pada kata "mempengaruhi", "mengkritik", "terpercaya", dan "pekerja".
2) Tidak dapat untuk melakukan proses pemenggalan kata-kata dengan bentuk perulangan, misalnya pada kata "buku-buku".
3) Algoritma Nazief-Adriani melakukan proses *stemming* dengan menghilangkan akhiran terlebih dahulu, kemudian diikuti dengan penghilangan awalan. Langkah ini akan menghasilkan hasil *stemming* yang tidak tepat pada beberapa kata, misalnya pada kata "dimulai". Hasil *stemming* algoritma Nazief-Adriani akan menghasilkan kata "mula". Padahal kata dasar yang tepat dari kata "dimulai" adalah "mulai". Hal ini dikarenakan algoritma bekerja dengan melakukan penghilangan akhiran terlebih dahulu. Pada kata "dimulai", langkah yang seharusnya dilakukan adalah penghilangan awalan terlebih dahulu.

Untuk memperbaiki kesalahan-kesalahan di atas, dilakukan beberapa buah perbaikan terhadap algoritma Nazief-Adriani oleh algoritma *CS Stemmer* sebagai berikut :

1) Penggunaan kamus kata dasar yang lebih lengkap. Sebagai contoh, kata "alasan-alasan" salah di-*stem* oleh *stemmer* Nazief-Adriani menjadi "alas" karena kata "alasan" tidak terdapat di dalam kamus kata dasar.
2) Melakukan modifikasi dan penambahan aturan Tabel 2.1 yang dapat dilihat pada Tabel 2.3.
3) Menambahkan aturan *stemming* untuk kata ulang. Caranya, adalah dengan melakukan pemisahan menjadi dua sub-kata, yakni sub-kata sebelum dan sesudah tanda penghubung "-". Setelah dilakukan pemisahan, masing-masing sub-kata mengalami proses *stemming*. Apabila *stemming* memberikan kata dasar yang sama, maka *output* kata dasarnya adalah hasil *stemming* tersebut. Namun apabila hasil *stemming* 2 sub-kata ini berbeda, maka dapat disimpulkan bahwa input adalah kata ulang semu, dan tidak memiliki bentuk kata dasar lagi.
4) Menambahkan proses pengecekan *rulePrecedence* dalam tahapan dalam proses *stemming*. Proses pengecekan *rulePrecedence* akan menentukan proses *stemming* akan melakukan penghilangan akhiran atau awalan dahulu.

C. Algoritma Enhanded Confix Stripping (ECS) Stemmer

Setelah dilakukan beberapa percobaan dan analisis, ditemukan beberapa kata yang tidak dapat di-*stemming* menggunakan *Confix Stripping Stemmer*. Analisis terhadap kata-kata yang gagal di-*stemming* tersebut sebagai berikut :

1) Kurangnya aturan pemenggalan awalan untuk kata-kata dengan format "mem+p...", "men+s...", dan "peng+k...". Hal ini terjadi pada kata "mempromosikan", "memproteksi", "mensyaratkan", "mensyukuri", dan "pengkajian".
2) Kurang relevannya aturan 17 dan 30 untuk pemenggalan awalan pada kata-kata dengan format "menge+kata dasar" dan "penge+kata dasar", seperti pada kata "mengerem" dan "pengeboman".
3) Adanya elemen pada beberapa kata dasar yang menyerupai suatu imbuhan. Kata-kata seperti "pelanggan", "perpolitikan", dan "pelaku" gagal di-stemming karena akhiran "-an", "-kan" dan "-ku" seharusnya tidak dihilangkan.

Untuk memperbaiki kesalahan-kesalahan di atas, algoritma *ECS Stemmer* [2] melakukan beberapa buah perbaikan sebagai berikut :

1) Melakukan modifikasi dan penambahan aturan Tabel 2.1 yang dapat dilihat pada Tabel 2.3.
2) Menambahkan suatu algoritma tambahan untuk mengatasi kesalahan pemenggalan akhiran yang seharusnya tidak dilakukan. Algoritma ini disebut *loopPengembalianAkhiran*, dan dilakukan apabila proses *recoding* gagal. Algoritma *loopPengembalianAkhiran* dideskripsikan sebagai berikut:
   a. Kembalikan seluruh awalan yang telah dihilangkan sebelumnya, sehingga menghasilkan model kata seperti berikut: [DP+[DP+[DP]]] + Kata Dasar. Pemenggalan awalan dilanjutkan dengan proses pencarian di kamus kemudian dilakukan pada kata yang telah dikembalikan menjadi model tersebut.
   b. Kembalikan akhiran sesuai dengan urutan model pada bahasa Indonesia. Ini berarti bahwa pengembalian dimulai dari DS ("-i", "-kan", "-an"), lalu PP("-ku", "-mu", "-nya"), dan terakhir adalah P ("-lah", "-kah", "-tah", "-pun"). Untuk setiap pengembalian, lakukan langkah 3) hingga 5) berikut. Khusus untuk akhiran "-kan", pengembalian pertama dimulai dengan "k", baru kemudian dilanjutkan dengan "an".
   c. Lakukan pengecekan di kamus kata dasar. Apabila ditemukan, proses dihentikan. Apabila gagal, maka lakukan proses pemenggalan awalan berdasarkan aturan pada Tabel 2.1 (dengan revisi Tabel 2.3).
   d. Lakukan *recoding* apabila diperlukan.
   e. Apabila pengecekan di kamus kata dasar tetap gagal setelah *recoding*, maka awalan-awalan yang telah dihilangkan dikembalikan lagi.

D. Algoritma Connected Component

Permasalahan *overstemming* dan *understemming* tidak dapat diselesaikan dengan melakukan *stemming* dengan hanya melihat kata per kata atau melakukan modifikasi tabel aturan pemenggalan. Penyebabnya adalah hasil dari proses *stemming*

yang dapat berjumlah lebih dari satu kata. Jika menggunakan teknik *stemming* kata per kata, maka hasil akhir dari *stemming* bergantung dari algoritma *stemming* yang digunakan apakah menggunakan pemenggalan semaksimal mungkin atau sebaliknya, yaitu seminimal mungkin.

Contoh kasusnya terdapat proses *stemming* dari kata "mengalami". Hasil dari pemenggalan awalan "meng-" akan menghasilkan "alami". Kata "alami" merupakan sebuah kata dasar. Tetapi jika kita teruskan proses pemenggalan lebih lanjut, penghilangan akhiran "-i" dapat menghasilkan kata dasar "alam". Kedua hasil stemming, "alami" dan "alam", merupakan kata dasar yang *valid*. Namun, hasil *stemming* yang paling tepat tidak dapat ditentukan jika hanya memperhatikan kata yang akan di-*stemming* saja.

Kata-kata lain yang berada di sekitar kata yang akan di-*stem* dapat mempengaruhi hasil *stemming*. Begitu pula dengan corpus atau koleksi dokumen yang digunakan dalam percobaan. Proses *stemming* yang demikian dinamakan dengan *corpus based stemming* [6].

Kata-kata berimbuhan yang memiliki hasil *stemming* yang sama dikatakan berada pada satu kelas yang sama. Pada *corpus based stemming*, kata-kata yang berada pada satu kelas tidak hanya memiliki hasil *stemming* yang sama, namun juga tiap pasangan kata yang berada pada kelas tersebut haruslah memiliki nilai *em* yang lebih besar dari *threshold* yang telah ditentukan sebelumnya.

Variabel *em* merupakan variabel yang mengukur kedekatan hubungan antara satu kata dengan kata yang lain berdasarkan sering atau tidaknya kedua buah kata tersebut muncul secara bersamaan dan berdekatan di dalam *corpus* yang digunakan pada percobaan. Jika terdapat sebuah pasangan kata yang memiliki nilai *em* yang lebih besar dari *threshold*, maka kedua buah kata tersebut memiliki hubungan yang dekat. Dengan demikian, kedua buah kata tersebut harus diletakkan di dalam satu kelas yang sama. Hasil akhirnya adalah kelas kata yang terbentuk memiliki anggota dengan nilai *em* pasangan kata anggotanya lebih besar dari *threshold*.

Untuk mendapatkan nilai *em* dari suatu pasang kata, kita menghitung proporsi kemunculan secara bersamaan kedua kata tersebut yang melebihi nilai ekspektasi kemunculan kedua kata tersebut secara bersamaan. Nilai *em* di antara dua buah kata dihitung dengan persamaan di bawah ini :

$$em(a,b) = max(\frac{n_{ab} - En(a,b)}{n_a + n_b}, 0)$$

Variabel n_a dan n_b adalah jumlah frekuensi kata *a* dan *b* pada koleksi dokumen dan n_{ab} merupakan frekuensi kedua kata tersebut muncul secara bersamaan di dalam jendela teks yang sama. Kata *a* dan *b* dikatakan berada pada jendela teks yang sama jika jarak keduanya di dalam suatu dokumen kurang dari batas jendela teks yang telah ditentukan sebelumnya. Lebih jelasnya, n_{ab} adalah jumlah elemen di dalam set $\{ < a_i, b_j > | dist(a_i, b_j) < win \}$, di mana a_i dan b_j merupakan *distinct occurence* dari a dan b di dalam *corpus*, sedangkan $dist(a_i, b_j)$ adalah jarak kata antara a_i dan b_j pada tiap dokumen. Dan *win* adalah ukuran jendela kata yang telah ditentukan sebelumnya.

TABEL 2.4
CONTOH PENGHITUNGAN NILAI EM UNTUK TIAP PASANGAN KATA

stem	term 1	frek 1	term 2	frek 2	co-oc	em
segel	segel	4	menyegel	2	4	0,66
pungut	pungut	2	pemungutan	6	5	0,62
serap	terserap	16	menyerap	24	19	0,45
kemas	kemasan	26	dikemas	1	9	0,33
selat	selatan	199	selat	16	5	0

Sedangkan *En(a,b)* merupakan nilai ekspektasi munculnya kata *a* dan *b* secara bersama-sama, dengan asumsi awal bahwa kedua kata tersebut tidak saling mempengaruhi (*statictically independent*). Untuk menghitung nilai *En(a,b)* digunakan persamaan sebagai berikut :

$$En(a,b) = kn_a n_b$$

Nilai *k* merupakan faktor konstan yang diperoleh, berdasarkan *corpus* dan ukuran jendela teks yang dipergunakan dalam percobaan. Nilai *k* selanjutnya dilakukan estimasi dengan sample berukuran besar yang dipilih secara *random* atau acak dari pasangan kata yang terdapat pada corpus yang digunakan. Persamaan untuk menghitung nilai *k* adalah sebagai berikut :

$$k = \frac{\sum n_{ab}}{\sum n_a n_b}$$

Pada Tabel 2.4 ditunjukkan penghitungan nilai *em* untuk beberapa buah pasangan kata dalam *corpus* atau koleksi dokumen. 5000 pasangan kata yang dipilih secara acak dengan nilai $k = 0.0022716$ dan ukuran jendela kata = 100.

Berdasarkan rumus ekspektasi En(a,b), *term* 'selatan' dan 'selat' yang memiliki hasil *stem* yang sama yakni 'selat', memiliki ekspektasi muncul secara bersamaan (*co-occur*) berjumlah 8. Oleh karena itu, nilai *em* dari *term* 'selatan' dan 'selat' adalah 0. Yang berarti keduanya tidak memiliki hubungan satu sama lain.

Setelah nilai *em* dari tiap pasangan kata di dalam sebuah kelas diketahui, maka langkah selanjutnya adalah melakukan *class refinement* atau pembentukan kelas-kelas baru dari sebuah kelas awal dengan penggunaan *connected component algorithm* untuk melakukan partisi terhadap kelas *stem* yang dibentuk oleh *stemmer* kata per kata.

Connected component algorithm dilakukan dengan cara menghubungkan kata-kata yang memiliki nilai *em* lebih besar daripada nilai *threshold* untuk em yakni 0,01 sesuai dengan yang digunakan oleh Larkey, Ballesteros, dan Cornell dalam percobaannya [4]. Tiap-tiap *graph* yang terbentuk selanjutnya akan membentuk sebuah kelas tersendiri.

Pada Tabel 2.5 diberikan contoh hasil pemrosesan pada kelas stem 'desa' menggunakan algoritma *connected component*. Proses pertama yang dilakukan yakni penghitungan nilai *em* untuk tiap pasangan kata pada kelas *stem* 'desa'.

Setelah nilai *em* untuk setiap pasangan kata selesai dihitung, maka langkah selanjutnya adalah menghubungkan pasangan kata dengan nilai *em* > *threshold* yang ditentukan (*threshold* = 0,01). Perhatikan Gambar 2.1.

TABEL 2.5
PENGHITUNGAN NILAI *EM* PADA ALGORITMA CC

stem	*term1*	*frek1*	*term2*	*frek2*	*cooc*	*em*
desa	desa-desa	1	desa	45	1	0,022
	desakan	16	desa	45	0	0
	desakan-desakan	1	desa	45	0	0
	pedesaan	3	desa	45	0	0
	desakan	16	desa-desa	1	0	0
	desakan-desakan	1	desa-desa	1	0	0
	pedesaan	3	desa-desa	1	0	0
	desakan-desakan	1	desakan	16	0	0
	pedesaan	3	desakan	16	0	0
	pedesaan	3	desakan-desakan	1	0	0

TABEL 2.6
HASIL PEMBENTUKAN KELAS BARU OLEH ALGORITMA CC

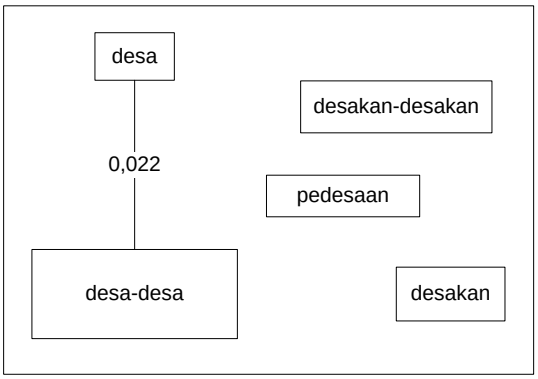
class	*term*
1	desa
	desa-desa
2	desakan
3	desakan-desakan
4	pedesaan

Jumlah kelas baru yang terbentuk dari kelas *stem* awal ‘desa’ ditentukan oleh jumlah *graf* yang terbentuk. Dari pembentukan *graf* di atas, maka jumlah kelas yang terbentuk adalah 4 buah kelas. Perhatikan Tabel 2.6, tampak bahwa *term* ‘desa’ dan ‘desa-desa’ berada pada satu kelas yang terpisah dari *term* ‘desakan’ dan ‘desakan-desakan’. Meskipun demikian, *term* ‘pedesaan’ karena tidak memiliki nilai em yang cukup, *term* tersebut tidak berada satu kelas dengan *term* ‘desa’ dan ‘desa’. Begitu juga dengan *term* ‘desakan’ dan ‘desakan-desakan’ yang meskipun seharusnya berada pada satu kelas, karena tidak memiliki nilai *em* yang cukup, keduanya berada pada kelas yang berbeda.

E. Algoritma pemilihan hasil stemming dengan nilai em terbesar

Permasalahan *overstemming* dan *understemming* tidak dapat diatasi jika proses *stemming* hanya dilakukan secara kata per kata dengan mengacu pada aturan pembentukan kata pada suatu bahasa dan kamus kata dasar. Hal ini disebabkan oleh hasil proses *stemming* tersebut yang dapat berjumlah dua hingga tiga buah kata dasar, yakni berdasarkan aturan yang terdapat pada tabel acuan pemengggalan awalan pada nomor 1, 6, 13, 15, 17, 21, 26, 28, 30, 31 dan 32. Pada aturan-aturan tersebut, proses pencarian bentuk dasar dari suatu kata tidak hanya berupa penghilangan awalan, tetapi juga adanya proses *recoding* jika proses penghilangan awalan tidak berhasil menemukan bentuk dasar dari kata tersebut.

Sebagai contoh, pemenggalan kata “pengawal” yang hasil *stemming*-nya dapat berupa kata “kawal” atau dapat juga berupa kata “awal” tergantung dari penerapan aturan nomor 30. Jika aturan “peng-V” diterapkan terlebih dahulu, maka *stemming* kata “pengawal” tersebut menghasilkan kata “awal”.



Gambar 2.1 Proses pembentukan *graf* pada algoritma CC

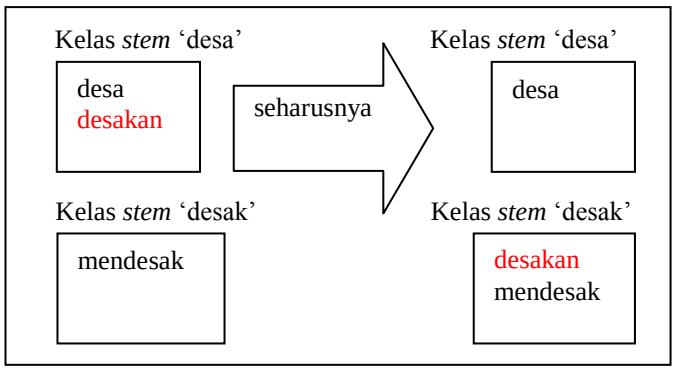
Dan kata “awal” dapat disebut sebagai hasil “*stemming*” yang *valid* karena ditemukan di dalam kamus kata dasar. Sebaliknya, jika aturan recoding “peng-kV” diterapkan terlebih dahulu, maka hasil “*stemming*” akan menghasilkan kata “kawal” yang juga dapat dikatakan sebagai hasil *stemming* yang valid karena juga dapat ditemukan di dalam kamus kata dasar.

Untuk mengatasi problem-problem tersebut, solusi berupa proses *stemming* berbasis *corpus* dapat dipergunakan. Dengan demikian, hasil *stemming* dari kata-kata dengan potensi *overstemming* dan *understemming* tersebut sangat ditentukan oleh *corpus* dokumen yang ditentukan di dalam percobaan.

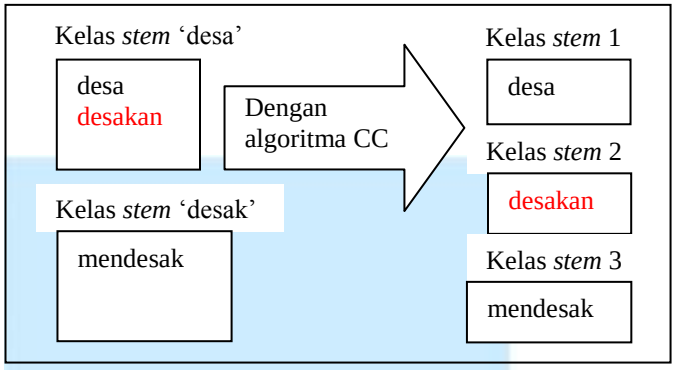
Namun, untuk memperbaiki kesalahan stemming pada nama orang, *overstemming*, dan *understemming*, algoritma *connected component* tidak dapat digunakan. Terdapat 2 alasan tidak dapat digunakannya algoritma *connected component* :

1) Dalam proses algoritma *connected component*, jumlah kelas *stem* yang terbentuk sangat ditentukan jumlah *graf* yang terbentuk dari proses pencarian nilai *em* dari tiap pasangan anggota kelas tersebut.
   Sehingga, ketika ada salah satu *term* dari suatu kelas *stem* yang tidak sesuai (hasil *stemming* dari *term* tersebut tidak benar), maka hasil yang diharapkan adalah *term* yang tidak sesuai tersebut dipisahkan dari kelas *stem* tersebut ke kelas *stem* yang baru. Jadi diharapkan akan terbentuk 2 kelas *stem* yang salah satunya hanya berisi *stem* yang tidak sesuai tersebut, dan kelas satunya berisi *term-term* lainnya. Namun, jumlah kelas *stem* yang baru oleh algoritma *connected component* sangat ditentukan oleh jumlah graf yang terbentuk dari penghubungan nilai *em* dari tiap pasang *term* dari kelas *stem* awal.
2) Saat terbentuk 2 kelas *stem* oleh algoritma *ECS Stemmer*, dan salah satu *term* dari salah satu kelas tersebut salah di-stem oleh *ECS Stemmer*, algoritma *connected component* tidak dapat memindahkan *term* tersebut ke kelas yang lain.
   Yang dapat dilakukan oleh algoritma *connected component* hanyalah memecah kelas *stem* dari *term* yang tidak tepat tersebut menjadi 2. Sehingga, kelas *stem* yang terbentuk adalah 3 kelas *stem*.

Misalkan, algoritma *ECS Stemmer* menghasilkan 2 buah kelas *stem* ‘desa’ dan ‘desak’ untuk *term* ‘desa’, ‘desakan’, dan ‘mendesak’.



Gambar 2.2 Hasil *stemming* yang tidak tepat



Gambar 2.3 Hasil pemecahan kelas oleh algoritma *connected component*

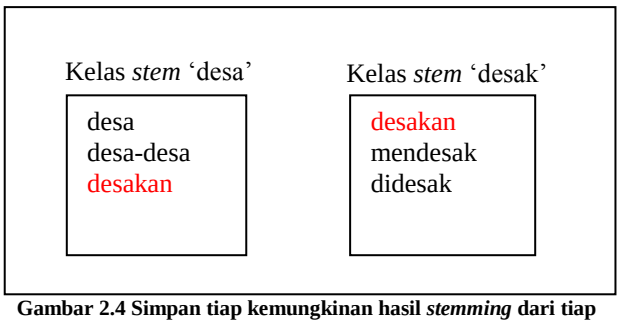
Kelas *stem* 'desa' terdiri dari *term* 'desa' dan 'desakan', sedangkan *term* 'mendesak' *term*asuk ke dalam kelas *stem* 'desak'. Ternyata dalam pengamatan (pengecekan kebenaran hasil *stemming* menggunakan Kamus Besar Bahasa Indonesia), *term* 'desakan' seharusnya berada pada kelas *stem* 'desak'. Perhatikan Gambar 2.2.

Akan tetapi, algoritma connected component tidak dapat melakukan perbaikan *term* 'desakan' dari kelas *stem* 'desa' ke kelas *stem* 'desakan'. Yang dapat dilakukannya adalah memecah kelas *stem* 'desa' menjadi 2. Sehingga, hasil akhirnya akan terbentuk 3 kelas *stem*. Perhatikan Gambar 2.3.

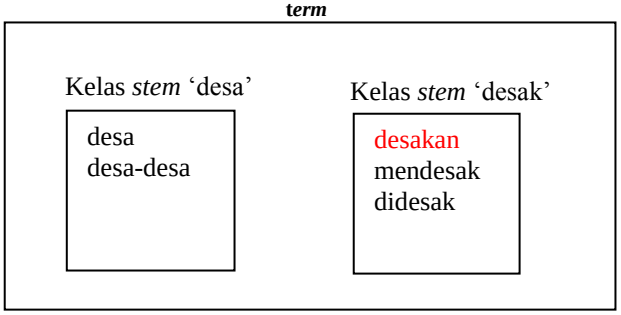
Untuk mengatasi 2 problem di atas, dalam tugas akhir ini dilakukan pengembangan terhadap metode *stemming* berbasis corpus dengan algoritma baru yang berbeda dengan algoritma *connected component*. Seperti telah dijelaskan di atas, algoritma *connected component* menimbulkan 2 problem di atas, yang menyebabkan tidak terselesaikannya hasil *stemming* yang benar dari permasalahan over*stemming*, under*stemming* dan nama-nama orang atau tempat yang tidak seharusnya di-*stemming*.

Untuk suatu *term* yang memiliki hasil *stem* lebih dari 1 (dapat dimasukkan ke dalam beberapa kelas *stem* yang berbeda), algoritma ini bekerja sebagai berikut :

a) Lakukan proses *stemming* terhadap tiap *term-term* indeks menggunakan algoritma ECS Stemmer yang telah diperbaiki. Simpan indeks tidak hanya hasil *stemming* dari perbaikan ECS Stemmer, tetapi juga kemungkinan-kemungkinan hasil stem yang lain (simpan). Gambar 2.4, tampak bahwa proses *stemming* pada *term* 'desakan' menghasilkan 2 buah kemungkinan hasil *stem*, yakni 'desa' dan 'desak'.



Gambar 2.4 Simpan tiap kemungkinan hasil *stemming* dari tiap *term*



Gambar 2.5 Hasil akhir dari algoritma pemilihan hasil *stemming* dengan nilai *em* terbesar

TABEL 2.7
TABEL PENGHITUNGAN NILAI EM SUATU TERM PADA TIAP KELAS *STEM*

Kelas *Stem*	*term* 1	*term* 2	*em*	*max(em)*
desa	desakan	desa	0	0
	desakan	desa-desa	0	
desak	desakan	mendesak	0,078	0,078
	desakan	didesak	0	

b) Lakukan pencarian nilai *em* dengan melakukan *pairing* atau pemasangan *term* yang bermasalah di atas dengan tiap anggota *term* dari tiap kelas *stem*. Namun, proses pairing ini ada satu catatan, yakni proses *pairing* dilakukan hanya dengan *term-term* yang benar-benar hanya berada pada kelas *stem* tersebut (hanya menghasilkan 1 buah hasil *stem*). Dapatkan nilai *em* tertinggi dari tiap kelas *stem*. Setelah didapatkan nilai *em* tertinggi, lakukan perbandingan nilai *em* dari *term* yang bermasalah tersebut pada tiap kelas *stem* terhadap kelas *stem* yang lain. Perhatikan Tabel 2.7.

c) Kelas *stem* yang memiliki nilai *em* tertinggi akan ditetapkan sebagai kelas *stem* untuk *term* yang bermasalah di atas. Dari proses sebelumnya, tampak bahwa proses pairing *term* 'desakan' dengan *term-term* dari kelas *stem* 'desa' dan 'desak' menghasilkan nilai *em* 0 untuk kelas *stem* 'desa' dan 0,078 untuk kelas *stem* 'desak'. Dengan demikian, hasil *stem*ming dari *term* 'desakan' ditetapkan sebagai 'desak'. Perhatikan gambar 2.5.

F. Perbaikan terhadap algoritma ECS Stemmer

Algortima *ECS Stemmer* merupakan perbaikan dari algoritma *CS Stemmer*. Sayangnya, algoritma *ECS Stemmer* masih melakukan beberapa kesalahan *stemming* yang dapat dilihat pada Tabel 2.8.

TABLE 2.8
KLASIFIKASI KEGAGALAN *ECS STEMMER*

Tipe kesalahan	Contoh Kasus		
	Awal	*stemming*	Seharusnya
sisipan	temaram	temaram	taram
Overstemming	penyidikan	sidi	sidik
Understemming	mengalami	alami	alam
Nama orang	Gumai	Guma	gumai
Kesalahan aturan pemenggalan awalan ke-18	menyatakan	menyatakan	nyata
Kesalahan aturan pemenggalan awalan ke-31	penyanyi	penyanyi	nyanyi
Kata gabungan (*compound words)*	diberitahu	diberitahu	beritahu

Setelah melakukan pengelompokkan kesalahan *stemming* yang dilakukan oleh *ECS Stemmer*, selanjutnya dilakukan perbaikan untuk mengatasi setiap kesalahan yang terjadi :

1) Sisipan
   Imbuhan dalam bahasa Indonesia mengenal adanya sisipan, yang terdiri dari "er", "el", "em" dan "in". Aturan yang dibuat sebelumnya hanya mengenal imbuhan yang berupa awalan dan akhiran. Untuk itu perlu ditambahkan aturan reduksi untuk sisipan guna memperbaiki kesalahan *stemming* untuk kata yang memiliki sisipan. Proses reduksi sisipan dilakukan setelah proses reduksi awalan dan akhiran selesai dilakukan.
2) Kesalahan aturan pemenggalan ke-18 dan 31
   Untuk memperbaiki kesalahan *stemming* kata yang terkait dengan aturan 18 dan 31 pada Tabel 2.2, maka dilakukan revisi aturan tersebut.
3) Kata gabungan (*compound word*)
   Untuk dapat melakukan proses *stemming* pada kata gabungan yang merupakan hasil dari penggabungan dua buah kata dasar, perlu ditambahkan langkah untuk melakukan pengecekan keberadaan kata turunan dalam algoritma *ECS Stemmer*. Proses ini dilakukan apabila tidak ditemukan bentuk dasar dari kata yang di-*input*-kan pada kamus kata dasar setelah proses reduksi awalan dan akhiran selesai dilakukan.
   Ketika kata dasar tidak ditemukan, maka proses *stemming* kata yang di-*input*-kan diulangi sekali lagi. Namun kali ini yang dilakukan adalah pengecekan keberadaan kata gabungan. Hal tersebut dilakukan setelah proses reduksi awalan dan akhiran. Masing-masing kata pada kata turunan yang di-*input*-kan tentu saja harus terdapat pada kamus kata dasar yang dipergunakan.
4) Akhiran serapan bahasa asing
   Untuk melakukan reduksi akhiran yang berasal dari serapan bahasa asing, yang perlau dilakukan tentu saja adalah melakukan pendaftaran akhiran serapan bahasa asing ke dalam tabel aturan pemenggalan imbuhan. Akhiran serapan bahasa asing tersebut, yakni '-wan', '-wati', '-is', '-isme', dan '-isasi'.
5) Nama orang, tempat, o*verstemming* dan *understemming*
   Pengecualian proses *stemming* terhadap nama-nama orang, tempat, ataupun organisasi tidak dapat dilakukan. Hal ini disebabkan oleh tidak diketahuinya apakah setiap *input* kata yang dimasukkan ke dalam proses *stemming* tersebut merupakan nama-nama orang, tempat, ataupun organisasi. Pendataan satu per satu terhadap nama-nama tersebut merupakan proses yang tidak dimungkinan dikarenakan jumlahnya yang tidak dapat dihitung dan masih terdapat kemungkinan untuk selalu bertambah. Selain itu, proses *stemming* yang dilakukan berdasarkan aturan pembentukan kata di dalam suatu bahasa akan melakukan proses *stemming* jika kata yang di-*input*-kan mengandungkan komponen imbuhan dan berhasil dilakukan jika menghasilkan kata dasar yang terdapat pada kamus kata dasar yang digunakan.
   Permasalahan *overstemming* dan *understemming* tidak dapat diatasi jika proses *stemming* hanya dilakukan secara kata per kata dengan mengacu pada aturan pembentukan kata pada suatu bahasa dan kamus kata dasar. Hal ini disebabkan oleh hasil proses *stemming* tersebut yang dapat berjumlah dua hingga tiga buah kata dasar, yakni berdasarkan aturan yang terdapat pada tabel acuan pemengggalan awalan pada nomor 1, 6, 13, 15, 17, 21, 26, 28, 30, 31 dan 32. Pada aturan-aturan tersebut, proses pencarian bentuk dasar dari suatu kata tidak hanya berupa penghilangan awalan, tetapi juga adanya proses *recoding* jika proses penghilangan awalan tidak berhasil menemukan bentuk dasar dari kata tersebut.
   Sebagai contoh, pemenggalan kata "pengawal" yang hasil *stemming*-nya dapat berupa kata "kawal" atau dapat juga berupa kata "awal" tergantung dari penerapan aturan nomor 30. Jika aturan "peng-V" diterapkan terlebih dahulu, maka *stemming* kata "pengawal" tersebut akan menghasilkan kata "awal". Dan kata "awal" dapat disebut sebagai hasil "*stemming*" yang *valid* karena ditemukan di dalam kamus kata dasar. Sebaliknya, jika aturan "peng-kV" diterapkan terlebih dahulu, maka hasil "*stemming*" akan menghasilkan kata "kawal" yang juga dapat dikatakan sebagai hasil *stemming* yang valid karena juga dapat ditemukan di dalam kamus kata dasar.
   Untuk mengatasi problem tersebut, solusi berupa proses *stemming* berbasis *corpus* dapat dipergunakan. Menggunakan "algoritma pemilihan hasil stemming dengan nilai em terbesar", maka problem *overstemming* dan *understemming* yang memiliki kemungkinan hasil *stemming* lebih dari satu dapat diselesaikan. Algoritma pemilihan hasil *stemming* nilai nilai em terbesar akan memilih salah satu dari kemungkinan-kemungkinan hasil *stem* tersebut yang memiliki *em* terbesar.

III. EVALUASI STEMMING

A. Penghitungan bobot tf-idf

Metode pembobotan yang paling sederhana terhadap suatu *term* (*term weighting*) adalah dengan menggunakan frekuensi kemunculan *term* (kata) bersangkutan pada suatu dokumen. Eksperimen-eksperimen *preprocessing* dokumen berbasiskan frekuensi *term*, telah banyak dilakukan dalam bidang *information retrieval*. Akan tetapi, dalam kaitannya dengan

performa *recall* dan *precision*, penggunaan frekuensi *term* saja ternyata hanya dapat memenuhi fungsi *recall*.

Fungsi *precision* yang baik, sayangnya tidak dapat dicapai dengan representasi frekuensi *term* yang saja pada suatu dokumen. Disini, *precision* yang tinggi menyiratkan kemampuan untuk membedakan suatu dokumen dengan dokumen lain untuk mencegah *retrieval* yang tidak diinginkan. Frekuensi *term* yang tinggi dapat digunakan untuk *preprocessing*, hanya jika frekuensi kemunculan *term* bersangkutan tidaklah tinggi pada dokumen-dokumen lainnya.

Nilai *precision* yang baik pada kenyataannya dihasilkan oleh *term-term* yang kemunculannya tergolong jarang pada suatu dokumen, karena *term-term* bersangkutan seringkali menjadi pembeda signifikan antara dokumen-dokumen yang memiliki *term-term* tersebut dengan dokumen-dokumen yang tidak memiliki *term-term* bersangkutan.

Untuk meningkatkan *precision*, digunakanlah representasi *Inverse Document Frequency (IDF)* untuk *term-term*, yang didefinisikan sebagai logaritma dari rasio jumlah keseluruhan dokumen yang diproses dengan jumlah dokumen yang memiliki *term* bersangkutan. Ini berarti bahwa *termterm* (kata) yang tingkat kemunculannya jarang akan memiliki nilai *IDF* yang tinggi. Telah dibuktikan melalui eksperimen, bahwa penggunaan *IDF* akan menghasilkan performa *retrieval* yang lebih efektif jika dibandingkan dengan penggunaan *frekuensi term (TF)* saja.

Adanya pembobotan klasik berbasis frekuensi dan *IDF* menjadi inspirasi untuk mengkombinasikan kedua metode pembobotan tersebut, dengan mempertimbangkan frekuensi inter-dokumen dan frekuensi intra-dokumen dari suatu *term*. Dengan menggunakan frekuensi *term* pada suatu dokumen dan distribusinya pada keseluruhan dokumen, yakni kemunculannya pada dokumen-dokumen lain (IDF), dapat ditarik suatu kesimpulan melalui eksperimennya bahwa *term-term* dengan total frekuensi menengah, lebih berguna dalam *retrieval* jika dibandingkan dengan *term-term* yang total frekuensinya terlalu tinggi atau terlalu rendah. Metode pembobotan yang menggabungkan konsep frekuensi intra-dokumen dan inter-dokumen ini kemudian dikenal sebagai metode *TF-IDF*, yang dinyatakan seperti rumus berikut :

$$w_{ij} = tf_{ij}.\log_2(\frac{N}{dfj})$$

dimana *wij* menandakan bobot atau seberapa penting suatu *term j* pada dokumen *i*, *tfij* merupakan frekuensi *term* j pada dokumen *i*, *N* merupakan jumlah total dokumen yang diproses, sedangkan *dfj* adalah jumlah dokumen yang memiliki *term j* didalamnya. Penggunaan metode pembobotan TF-IDF ini telah terbukti mampu memberikan nilai *retrieval* yang efektif, baik dari sisi *recall* maupun *precision*. Selain itu, metode pembobotan *TF-IDF* ini juga tergolong mudah dalam implementasinya.

B. Penghitungan tingkat kemiripan

Penghitungan kemiripan (*similarity*) pada saat proses pencarian dokumen dari input query user adalah standar *cosine similarity* dengan rumus :

$$\text{score}(q,d) = \frac{\vec{V}(q) \,.\, \vec{V}(d)}{|\vec{V}(q)|.|\vec{V}(d)|}$$

Pada ujicoba, vektor dari *query* direpresentasikan menggunakan nilai bobot *IDF* untuk tiap *term*-nya. Sedangkan vektor dari dokumen-dokumen yang ada di koleksi direpresentasikan menggunakan nilai bobot *TF* untuk tiap *term* di dalamnya. Dokumen yang berada pada posisi teratas pada saat proses pencarian dokumen adalah dokumen dengan nilai *cosine similarity* tertinggi hasil dari penghitungan *TF-IDF* untuk tiap *term* pada *query*.

C. Evaluasi sistem temu kembali informasi

Untuk melakukan evaluasi terhadap performa suatu sistem temu kembali informasi, pertama-tama tentu saja dibutuhkan koleksi dokumen atau *corpus* yang akan digunakan di dalam proses uji coba. Setelah koleksi dokumen selesai dikumpulkan, selanjutnya adalah pembentukan set *query* dan set dokumen-dokumen yang relevan terhadap *query-query* tersebut. Perhatikan Gambar 3.1 dan Gambar 3.2.

Suatu dokumen dikatakan relevan terhadap *query* tidak hanya jika *term-term* pada *query* terdapat pada dokumen. Dokumen tersebut dikatakan relevan jika informasi yang diinginkan dari *query* tersebut terdapat di dokumen. Contoh, dengan *query* "nama Presiden Indonesia", dokumen-dokumen yang mengandung *term-term* dari *query* tersebut tidak dapat dikatakan relevan jika tidak menyebutkan nama dari Presiden. Jadi, misalkan suatu dokumen mengandung *term* "Presiden Susilo Bambang Yudhoyono", maka dokumen tersebut barulah dapat dikatakan relevan karena mengandung nama dari Presiden Indonesia. Salah satu *term* dari *query* tentu saja harus terdapat pada dokumen. Karena apabila tidak ada *term* dari *query* pada dokumen tersebut, maka otomatis nilai similaritas dari dokumen tersebut adalah nol dan tidak akan keluar pada proses pencarian dokumen.

Setelah m*em*persiapkan *query* set dan *relevan set*, maka tiap *query* dalam *query* set di-*input*-kan ke dalam si*stem*. K*em*udian dilakukan pengecekan hasil pencarian dokumen oleh si*stem* tersebut terhadap *relevan set* yang sesuai dengan *query* yang sebelumnya di-*input*-kan. Suatu dokumen yang berada pada hasil pencarian si*stem* tersebut dikatakan relevan jika terdapat pada relevan set dan begitu pula sebaliknya.

D. Ukuran performa efektifitas sistem temu kembali informasi

Untuk mengukur performa efektifitas suatu sistem temu kembali informasi ada berbagai cara. Ada beberapa ukuran, di antaranya terdapat *recall, precision,* dan *Mean Average Precision (MAP).*

Sebagai contoh, hasil pencarian suatu sistem terhadap *query* tersebut dapat dilihat pada Tabel 2.8.

TABLE 2.8
KLASIFIKASI KEGAGALAN *ECS STEMMER*

Rank	Relevan (R) / Tidak Relevan (T)
1	R
2	T
3	T
4	R
5	R
6	R

Query set :

1. Kerusakan akibat pemadaman listrik bergilir
2. Perombakan kabinet

Dokumen set :

1. IPMI: Pemadaman Listrik Bisa Picu Deindustrialisasi
2. Listrik Byar Pet, Biaya Pengiriman Barang Melonjak 30X Lipat
3. Mal Kurangi Suhu AC, Industri Tambah Pemakaian Genset
4. 19 Program Ekonomi Hatta Rajasa
5. DPR Segera Gelar Pemilihan Gubernur BI
6. Gita Wirjawan Dilantik Senin Pekan Depan

Gambar 3.1 Contoh Query set dan Dokumen set

Relevan Set :

Query : 'Kerusakan akibat pemadaman listrik bergilir' :

Dokumen relevan terhadap query :

1. HIPMI: Pemadaman Listrik Bisa Picu Deindustrialisasi
2. Listrik Byar Pet, Biaya Pengiriman Barang Melonjak 30X Lipat
3. Mal Kurangi Suhu AC, Industri Tambah Pemakaian Genset

Query : 'perombakan kabinet' :

1. 19 Program Ekonomi Hatta Rajasa
2. DPR Segera Gelar Pemilihan Gubernur BI
3. Gita Wirjawan Dilantik Senin Pekan Depan

Gambar 3.2 Contoh relevan set dokumen terhadap query

Sedangkan, total dokumen relevan dalam *relevan set* terhadap *query* tersebut adalah 10. Selanjutnya di bawah ini akan dijelaskan pengertian dari tiap ukuran efektifitas dan cara penghitungannya.

a) *Recall*

Nilai *recall* merupakan perbandingan jumlah dokumen relevan hasil pencarian oleh sistem terhadap jumlah total dokumen relevan dalam relevan set.

$$Recall = \frac{\text{jumlah dokumen relevan pada hasil pencarian}}{\text{jumlah dokumen relevan pada relevan set}}$$

$$Recall = \frac{4}{10} = 0{,}4$$

Kemudian, *recall*(n) merupakan perbandingan jumlah dokumen relevan pada *n* dokumen teratas hasil pencarian oleh sistem terhadap jumlah total dokumen relevan dalam *relevan set*.

$$Recall(5) = \frac{3}{10} = 0{,}3$$

Nilai tertinggi dari *recall* adalah 1,0 yang berarti semua dokumen relevan dari relevan set berhasil dikembalikan seluruhnya oleh hasil pencarian dari sistem tersebut.

b) *Precision*

Nilai *precision* merupkan perbandingan jumlah dokumen relevan hasil pencarian oleh sistem terhadap jumlah total dokumen relevan maupun tidak relevan hasil pencarian sistem tersebut.

$$Precision = \frac{\text{jumlah dokumen relevan pada hasil pencarian}}{\text{jumlah dokumen pada hasil pencarian oleh sistem}}$$

$$Precision = \frac{4}{6} = 0{,}66..$$

Kemudian, *precision*(n) merupakan perbandingan jumlah dokumen relevan pada n dokumen teratas hasil pencarian oleh sistem terhadap jumlah total dokumen relevan maupun tidak relevan hasil pencarian sistem tersebut.

$$Precision(5) = \frac{3}{6} = 0{,}5$$

Nilai tertinggi dari *precision* adalah 1,0 yang berarti semua dokumen yang dikembalikan oleh hasil pencarian dari sistem tersebut adalah relevan.

c) Mean Average Precision (MAP)

Nilai *MAP* didapatkan dengan cara menghitung nilai *precision* setiap dokumen relevan ditemukan pada hasil pencarian suatu sistem.

Pada hasil pencarian tersebut, dokumen relevan yang pertama berada pada ranking 1. Maka, average *precision* dokumen tersebut adalah 1/1. Dokumen relevan yang kedua berada pada posisi 4. Maka, average *precision* untuk dokumen tersebut adalah 2/4. Setelah, dilakukan pengamatan untuk semua rangking dari hasil pencarian sistem tersebut, selanjutnya dihitung *mean* dari setiap average *precision* tiap dokumen.

$$\text{MAP} = \frac{\frac{1}{1} + \frac{2}{4} + \frac{3}{5} + \frac{4}{6}}{\sum \text{jumlah relevan dokumen pada relevan set}}$$

$$\text{MAP} = \frac{\frac{1}{1} + \frac{2}{4} + \frac{3}{5} + \frac{4}{6}}{10} = 0{,}27666..$$

Semakin tinggi nilai dari *MAP*, berarti semakin banyak jumlah dokumen yang relevan, yang dikembalikan oleh hasil pencarian sistem tersebut, berada pada posisi atau rank atas. Nilai *MAP* menjadi penting mengingat perilaku sebagian besar *user* yang hanya melakukan

pengecekan terhadap sebagian dokumen yang berada di rank atau posisi awal saja

E. Pembentukan relevansi set secara otomatis menggunakan data fusion dan metode condorcet

Latar belakang dari dikembangkannya teknik pembentukan *relevansi set* secara otomatis utamanya adalah banyaknya waktu yang dibutuhkan untuk membuat relevansi set jika koleksi dokumen yang digunakan dalam uji coba jumlahnya sangat banyak. Apalagi penilaian relevansi dilakukan untuk setiap query yang ada di dalam *query set.*

Untuk menghemat waktu dalam pembentukan *relevansi set*, dikembangkanlah suatu metode untuk membentuk relevansi set secara otomatis. Karena tidak dinilai secara manual, maka relevansi set yang dibuat secara otomatis ini dinamakan *pseudo relevance documents (pseudorels)*.

Pembentukan pseudorels dilakukan dengan teknik *data fusion* dan *metode condorcet* [5]. Teknik *data fusion* dilakukan dengan cara menggabungkan hasil pencarian dari beberapa buah sistem menjadi satu. *Data fusion* memiliki parameter yang dinamakan *pool depth*, yakni berapa banyak dokumen yang ingin diambil dari tiap sistem pencarian dokumen. Harapannya adalah mendapatkan hasil pencarian yang memiliki performa lebih baik dari hasil pencarian oleh sistem-sistem pembentuknya. Penggabungan hasil dari beberapa buah sistem tidak dilakukan begitu saja. Penggabungan dilakukan dengan *metode Condorcet* yang berperan dalam proses pemberian ranking dokumen relevan dengan memperhatikan rangking suatu dokumen pada tiap-tiap sistem. Pemberian ranking ini diperlukan jika *pseudorels* tidak digunakan semuanya (hanya sekian persen saja dari dokumen teratas).

Metode *condorcet* bekerja seperti proses pemungutan suara pada pemilihan umum. Yang bertindak sebagai pemilih adalah sistem yang akan dievaluasi. Sedangkan, yang bertindak sebagai kandidat yang dipilih adalah dokumen. Setiap sistem akan melakukan pemilihan dokumen-dokemen kemudian dilakukan pengurutan kandidat kondumen. Dokumen dengan nilai perbandingan menang kalah terhadap dokumen lainya yang paling besar akan ditempatkan pada urutan pertama.

Misalkan, terdapat 3 buah sistem yakni :

- Sistem A : sistem pencarian dokumen tanpa fitur *stemming*
- Sistem B : sistem pencarian dokumen dengan *ECS Stemmer*
- Sistem C : sistem pencarian dokumen dengan *corpus based stemming*

Query yang digunakan (*query set*) : perombakan kabinet

Dokumen yang terdapat pada koleksi (*dokumen set*) :

a. Pemerintah Belum Akan Rombak Direksi PLN
b. Waluyo Resmi Jadi Direktur Pertamina Lagi
c. 19 Program Ekonomi Hatta Rajasa

TABLE 3.1
MATRIKS MENANG-KALAH-SERI ANTAR DOKUMEN

	a	b	c
a	-	2, 1, 0	3, 0, 0
b	1, 2, 0	-	3, 0, 0
c	0, 3, 0	0, 3, 0	-

TABLE 3.2
PROSES PERANGKINGAN DOKUMEN BERDASARKAN NILAI MENANG, KALAH DAN SERI

	Menang	Kalah	Seri
a	2	0	0
b	1	1	0
c	0	2	0

Hasil pencarian dokumen yang dilakukan oleh tiap sistem menghasilkan posisi tiap dokumen sebagai berikut :

- Sistem A : a > b > c
- Sistem B : a > b > c
- Sistem C : b > a > c

Keterangan : a > b berarti rank dokumen a lebih tinggi dari dokumen b

Langkah pertama, membentuk matriks N x N untuk perbandingan berpasangan, dengan N adalah banyaknya kandidat. Tiap elemen matriks (i,j) yang tidak berada pada garis diagonal matriks, menunjukkan nilai kandidat i terhadap kandidat j. Nilai yang ditunjukkan adalah nilai menang, kalah, dan seri.

Selanjutnya, ditentukan pemenang hasil perbandingan. Setiap dokumen dibandingkan nilai menang dan kalahnya terhadap dokumen lainnya. Untuk dokumen yang memiliki nilai menang lebih banyak, pada kolom menang dokumen tersebut akan mendapat tambahan nilai 1 dan yang kalah mendapat tambahan nilai 1 pada kolom kalah. Jika hasilnya seri, maka pada kolom seri akan ditambahkan nilai 1.

Penentuan ranking dokumen ditentukan berdasarkan nilai menang dan kalahnya. Dokumen yang memiliki nilai menang paling banyak akan terletak di urutan paling atas. Jika terdapat 2 dokumen yang memiliki nilai menang sama, maka yang urutannya di atas adalah yang memiliki nilai kalah paling sedikit. Jika 2 buah dokumen yang dibandingkan memiliki nilai menang, kalah dan seri yang sama, maka urutannya dapat ditentukan secara *random*. Dari contoh di atas, maka urutan akhir dokumen adalah a > b > c.

Setelah penggabungan dan pengurutan hasil pencarian dari beberapa sistem menggunakan *data fusion* dan metode *condorcet*, maka langkah selanjutnya adalah penentuan jumlah dokumen dari *data fusion* yang akan digunakan sebagai *pseudorels*. Parameter ini dinamakan dengan *percentage merged documents*. Jika *percentage* yang digunakan adalah 100%, maka seluruh dokumen hasil penggabungan, dalam contoh di atas yakni dokumen a, b, dan c, seluruhnya digunakan sebagai relevan set dokumen terhadap query yang diiputkan di awal. Hasil pemrosesan inilah yang disebut sebagai *pseudo relevance dokumen (pseudorels)* karena tidak dibentuk melalui prosedur penilaian relevansi dokumen secara manual.

IV. UJI COBA DAN EVALUASI

A. Data Uji Coba

Data yang digunakan dalam uji coba aplikasi ini merupakan *corpus* atau kumpulan dokumen teks berita berbahasa Indonesia, yang dikumpulkan dari dua situs berita online berbahasa Indonesia www.detikfinance.com. ***Corpus*** yang

dimaksud di sini adalah kumpulan file *XML* yang isinya merupakan penyaduran judul, tanggal, dan isi dari dokumen berita yang pada awalnya berbentuk file *HTML*.

B. Uji coba menggunakan perbaikan dari algoritma ECS Stemmer

Berikut ini adalah term-term yang telah diperbaiki hasil *stemming*-nya. Perbaikan dilakukan dengan memperbaiki tabel aturan pemenggalan 18 dan 31, penambahan proses reduksi sisipan, penambahan akhiran serapan ke dalam proses reduksi akhiran dan tentu saja penggunakan metode *corpus based stemming* untuk menetukan hasil *stemming* dari term-*term* yang memiliki hasil *stemming* lebih dari satu.

TABEL 4.1
HASIL PERBAIKAN NILAI EM SUATU TERM PADA TIAP KELAS STEM

Aturan Nomor 18		
term	***ECS Stemmer***	**perbaikan *ecs***
menyala	sala	nyala
menyanyikan	menyanyikan	nyanyi
menyatakannya	menyatakannya	nyata

TABEL 4.2
HASIL PERBAIKAN ATURAN PEMENGGALAN NOMOR 31

Aturan Nomor 31		
term	***ECS Stemmer***	**perbaikan *ecs***
penyanyi	sanyi	nyanyi
penyawaan	sawa	nyawa

TABEL 4.2
HASIL PERBAIKAN DENGAN PENAMBAHAN REDUKSI SISIPAN

Sisipan		
term	***ECS Stemmer***	**perbaikan *ecs***
melamah	melamah	mamah
jelambar	jelambar	jambar
lemigas	lemigas	ligas
rerata	rerata	rata

TABEL 4.3
PERBAIKAN REDUKSI AKHIRAN SERAPAN BAHASA ASING

Akhiran serapan bahasa asing ('-is', '-isasi', '-isme', '-wan', '-wati')		
term	***ECS Stemmer***	**perbaikan *ecs***
relawan	relawan	rela
riawan	riawan	ria
salawati	salawat	sala
belesis	belesis	beles
eksis	eksis	eks
finalis	finalis	final
menepis	tepis	tep
minimalis	minimalis	minimal
brokerisasi	brokerisasi	broker
difinalisasi	difinalisasi	final
finalisasi	finalisasi	final
maksimalisasi	maksimalisasi	maksimal
memfinalisasi	memfinalisasi	final
standarisasi	standarisasi	standar
terealisasi	realisasi	real

TABEL 4.4
PERBAIKAN TERM DENGAN BENTUK DASAR KATA GABUNGAN

Kata Gabungan (*compound words)*		
term	***ECS Stemmer***	**perbaikan *ecs***
bekerjasama	bekerjasama	kerjasama
beritahukan	beritahukan	beritahu
berkerjasama	berkerjasama	kerjasama
berkewarganegaraan	berkewarganegaraan	warganegara
berterimakasih	berterimakasih	terimakasih
dibagihasilkan	dibagihasilkan	bagihasil
dibebastugaskan	dibebastugaskan	bebastugas
diberitahu	diberitahu	beritahu
diberitahukan	diberitahukan	beritahu
dibertanggungjawabkan	dibertanggungjawabkan	tanggungjawab
dipertanggungjawabkan	dipertanggungjawabkan	tanggungjawab
ditandatangani	ditandatangani	tandatangan
ditandatanganinya	ditandatanganinya	tandatangan
ditindaklanjut	ditindaklanjut	tindaklanjut
ditindaklanjuti	ditindaklanjuti	tindaklanjut
diujicoba	diujicoba	ujicoba
diujicobakan	diujicobakan	ujicoba
keanekaragaman	keanekaragaman	anekaragam

Pada Tabel 4.5 akan ditampilkan hasil perbaikan dari proses *stemming* menggunakan 'algoritma pemilihan hasil *stemming* dengan nilai *em* yang terbesar' :

TABEL 4.5
PERBAIKAN HASIL *STEMMING* DENGAN METODE *CORPUS BASED STEMMING*

***Overstemming, understemming*, nama orang, nama tempat**			
term	***ECS Stemmer***	**List Bentuk Dasar**	**Perbaikan *ECS Stemmer***
beratus	atus	[atus, beratus, ratus]	ratus
desakan	desa	[desa, desak, desakan]	desak
ditahan	tah	[ditah, ditahan, tah, tahan]	tahan
indukan	indu	[indu, induk, indukan]	induk
keliaran	keliar	[keliar, liar]	liar
kerusakan	rusa	[kerusa, kerusak, rusa, rusak, rusakan]	rusak
memadamkan	madam	[madam, madamkan, memadam, padam,	padam

		padamkan]	
memakai	maka	[maka, pakai]	pakai
memangkas	mangkas	[mangkas, memangkas, pangkas]	pangkas
memperhatikan	perhati	[hati, hatikan, memperhati, perhati, perhatikan]	hati
menandai	nanda	[nanda, nandai, tanda, tandai]	tanda
mengembangkan	embang	[embang, embangkan, kembang, kembangkan]	kembang
mengunjungi	unjung	[kunjung, unjung]	kunjung
mengurangi	urang	[kurang, urang]	kurang
pemadaman	madam	[madam, madaman, padam, padaman]	padam
pemajakan	maja	[maja, majakan, pajak]	pajak
pemungutan	mungut	[mungut, mungutan, pungut, pungutan]	pungut
penarikan	tari	[penari, penarik, tari, tarik, tarikan]	tarik
pengecekan	kece	[cek, ecek, kece, kecek, kecekan, pengecek]	cek
pengurangan	urang	[kurang, kurangan, urang]	kurang
penjajakan	jaja	[jaja, jajak, jajakan, penjaja, penjajak]	jajak
penyelidikan	lidi	[lidi, lidikan, selidik]	selidik
penyidikan	sidi	[nyidi, nyidik, nyikan, sidi, sidik, sidikan, sikan]	sidik
perancang	ancang	[ancang, pancang, perancang, rancang]	rancang
perancangan	ancang	[ancang, perancang, rancang]	rancang
perbankan	perban	[bank, perban, perbank, perbankan]	bank
pergerakan	gera	[gera, gerak, gerakan, pergera, pergerak]	gerak
perombakan	ombak	[ombak, perombak, rombak]	rombak
beratus	atus	[atus, beratus, ratus]	ratus
desakan	desa	[desa, desak, desakan]	desak
ditahan	tah	[ditah, ditahan, tah, tahan]	tahan
indukan	indu	[indu, induk, indukan]	induk
keliaran	keliar	[keliar, liar]	liar
kerusakan	rusa	[kerusa, kerusak, rusa, rusak, rusakan]	rusak

memadamkan	madam	[madam, madamkan, memadam, padam, padamkan]	padam
memakai	maka	[maka, pakai]	pakai
memangkas	mangkas	[mangkas, memangkas, pangkas]	pangkas
memperhatikan	perhati	[hati, hatikan, memperhati, perhati, perhatikan]	hati
menandai	nanda	[nanda, nandai, tanda, tandai]	tanda
mengembangkan	embang	[embang, embangkan, kembang, kembangkan]	kembang
mengunjungi	unjung	[kunjung, unjung]	kunjung
mengurangi	urang	[kurang, urang]	kurang
pemadaman	madam	[madam, madaman, padam, padaman]	padam
pemajakan	maja	[maja, majakan, pajak]	pajak
pemungutan	mungut	[mungut, mungutan, pungut, pungutan]	pungut
penarikan	tari	[penari, penarik, tari, tarik, tarikan]	tarik
pengecekan	kece	[cek, ecek, kece, kecek, kecekan, pengecek]	cek
pengurangan	urang	[kurang, kurangan, urang]	kurang
penjajakan	jaja	[jaja, jajak, jajakan, penjaja, penjajak]	jajak
penyelidikan	lidi	[lidi, lidikan, selidik]	selidik
penyidikan	sidi	[nyidi, nyidik, nyikan, sidi, sidik, sidikan, sikan]	sidik
perancang	ancang	[ancang, pancang, perancang, rancang]	rancang
perancangan	ancang	[ancang, perancang, rancang]	rancang
perbankan	perban	[bank, perban, perbank, perbankan]	bank
pergerakan	gera	[gera, gerak, gerakan, pergera, pergerak]	gerak
perombakan	ombak	[ombak, perombak, rombak]	rombak

C. Uji coba performa sistem ECS Stemmer sebelum dan sesudah diperbaiki menggunakan Data Fusion dan metode Condorcet

Query set yang digunakan dalam uji coba :
1. perombakan kabinet
2. penyidikan tersangka mafia pajak
3. tersangka gayus ditahan
4. kriminalisasi pimpinan kpk
5. memangkas praktek pemungutan liar
6. pengecekan kasus korupsi
7. menyuntikan modal tambahan
8. kerusakan lingkungan hidup
9. pemadaman listrik bergilir
10. penandatangan perjanjian ekstradisi
11. kerangka pembangunan ekonomi
12. rasionalisasi karyawan
13. pengurangan jumlah pegawai
14. hasil pemungutan suara
15. beratus wisatawan luar negeri
16. tersangka korupsi surati presiden
17. membangun kebijakan persaingan usaha
18. menandai majunya kondisi ekonomi
19. kerusakan akibat pemadaman listrik bergilir
20. sejarah pemikiran ekonomi

Proses evaluasi efektifitas sistem pencarian dokumen dilakukan menggunakan teknik *data fusion* dan metode *condorcet* dalam melakukan proses pembentukan *relevansi set* dokumen untuk tiap query pada *query set*. *Relevansi set* tersebut akan terbentuk secara otomatis untuk setiap query yang diinputkan ke dalam sistem pencarian dokumen.

Teknik *data fusion* dan *metode condorcet* yang digunakan dalam uji coba dilakukan dengan parameter *pool depth* 10, 20, dan 30 serta parameter *percetage merged documents* 30, 40 dan 50 persen. Pemilihan parameter *pool depth* dengan nilai 10, 20, 30 dilatarbelakangi dari kebiasaan *user* yang hanya melihat beberapa dokumen saja yang terletak di rangking teratas hasil pencarian. Sedangkan pemilihan parameter *percentage merged documents* dengan nilai 30, 40, dan 50 dilatarbelakangi oleh keinginan mendapat kurang dari 50% dokumen terbaik hasil penggabungan yang dilakukan sebelumnya.

Pengamatan terhadap hasil uji coba menunjukkan bahwa terjadi variasi dalam pencarian sistem terbaik untuk tiap nilai *pool depth* dan *percentage merged documents* pada tiap variabel *IR*. Untuk tiap nilai *pool depth* dan *percentage merged documents*, sistem terbaik untuk masing-masing variabel *IR* tidaklah sama.

Sistem pencarian dokumen yang menggunakan perbaikan dari *ECS Stemmer* mencatatkan nilai tertinggi untuk *recall*. Untuk beberapa nilai parameter, *recall* yang diperoleh antara sistem pencarian menggunakan *ECS Stemmer* sebelum dan sesudah diperbaiki bernilai sama besar. Hal ini terjadi akibat adanya fungsi *random* yang digunakan saat pengurutan ranking dengan *condorcet method* terhadap lebih dari satu dokumen memiliki nilai menang dan kalah yang sama.

TABEL 4.6
NILAI EFEKTIFITAS UNTUK TIAP SISTEM DENGAN PERCENTAGE MERGED DOCUMENTS = 30

Percentage merged documents = 30				
Pool Depth	***Measure***	**Tanpa *Stemming***	***ECS Stemmer***	**Perbaikan *ECS Stemmer***
10	***Recall***	0,955	**1,000**	**1,000**
	Recall(10)	0,862	**0,890**	0,787
	Recall(20)	0,879	**0,941**	0,863
	Precision	**0,073**	0,055	0,038
	Precision(10)	**0,069**	0,052	0,030
	Precision(20)	**0,072**	0,054	0,036
	MAP	0,758	**0,802**	0,663
20	***Recall***	0,941	**1,000**	**1,000**
	Recall(10)	0,733	**0,780**	0,640
	Recall(20)	0,829	**0,907**	0,835
	Precision	**0,143**	0,106	0,070
	Precision(10)	**0,121**	0,088	0,044
	Precision(20)	**0,136**	0,103	0,064
	MAP	0,750	**0,811**	0,681
30	***Recall***	0,890	0,992	**1,000**
	Recall(10)	0,548	**0,558**	0,483
	Recall(20)	0,761	**0,877**	0,861
	Precision	**0,198**	0,148	0,103
	Precision(10)	**0,139**	0,094	0,049
	Precision(20)	**0,184**	0,141	0,094
	MAP	0,685	**0,790**	0,705

TABEL 4.7
NILAI EFEKTIFITAS UNTUK TIAP SISTEM DENGAN PERCENTAGE MERGED DOCUMENTS = 40

Percentage merged documents = 40				
Pool Depth	***Measure***	**Tanpa *Stemming***	***ECS Stemmer***	**Perbaikan *ECS Stemmer***
10	***Recall***	0,956	**1,000**	**1,000**
	Recall(10)	0,825	**0,853**	0,765
	Recall(20)	0,850	**0,916**	0,865
	Precision	**0,106**	0,078	0,053
	Precision(10)	**0,096**	0,072	0,040
	Precision(20)	**0,102**	0,076	0,049
	MAP	0,799	**0,846**	0,702
20	***Recall***	0,928	0,992	**1,000**
	Recall(10)	0,681	**0,695**	0,583
	Recall(20)	0,818	**0,883**	0,832
	Precision	**0,182**	0,136	0,095
	Precision(10)	**0,146**	0,102	0,052
	Precision(20)	**0,172**	0,132	0,086
	MAP	0,790	**0,864**	0,726
30	***Recall***	0,889	0,981	**1,000**
	Recall(10)	**0,501**	0,471	0,412
	Recall(20)	0,752	**0,808**	0,791
	Precision	**0,248**	0,189	0,139
	Precision(10)	**0,161**	0,103	0,055
	Precision(20)	**0,230**	0,171	0,118
	MAP	0,775	**0,854**	0,768

TABEL 4.8
NILAI EFEKTIFITAS UNTUK TIAP SISTEM DENGAN PERCENTAGE MERGED DOCUMENTS = 50

Percentage merged documents = 50				
Pool Depth	***Measure***	**Tanpa *Stemming***	**ECS *Stemmer***	**Perbaikan *ECS Stemmer***
10	***Recall***	0,959	**1,000**	**1,000**
	Recall(10)	0,800	**0,828**	0,698
	Recall(20)	0,831	**0,908**	0,843
	Precision	**0,134**	0,098	0,068
	Precision(10)	**0,117**	0,087	0,045
	Precision(20)	**0,125**	0,095	0,062
	MAP	0,805	**0,864**	0,720
20	***Recall***	0,912	0,983	**1,000**
	Recall(10)	**0,587**	0,563	0,480
	Recall(20)	0,786	**0,845**	0,816
	Precision	**0,220**	0,167	0,121
	Precision(10)	**0,160**	0,103	0,054
	Precision(20)	**0,205**	0,156	0,105
	MAP	0,788	**0,858**	0,750
30	***Recall***	0,876	0,978	**0,997**
	Recall(10)	**0,403**	0,380	0,334
	Recall(20)	0,721	**0,746**	0,683
	Precision	**0,297**	0,236	0,176
	Precision(10)	**0,164**	0,106	0,056
	Precision(20)	**0,266**	0,202	0,126
	MAP	0,770	**0,853**	0,776

Semakin besar nilai *pool depth* dan *percentage merged documents*, semakin besar pula variasi sistem terbaik untuk tiap variabel *IR*.

Nilai *precision* yang paling tinggi dihasilkan oleh sistem pencarian dokumen tanpa *stemming*. Hal ini tidak lepas dari fungsi *stemming* itu sendiri yang mengubah tiap kata ke bentuk dasarnya masing-masing.

Nilai *MAP* yang paling tinggi dicatat oleh sistem pencarian dokumen menggunakan *ECS Stemmer*. Perbaikan terhadap *ECS Stemmer* ternyata membuat nilai *MAP* yang dihasilkan lebih rendah dari sistem pencarian dokumen tanpa *stemming*.

V. SIMPULAN DAN SARAN

A. Simpulan

Dari uji coba yang telah dilakukan dan setelah menganalisis hasil pengujian terhadap implementasi metode *corpus based stemming* untuk memperbaiki kesalahan *stemming* dari algoritma *ECS Stemmer* ini dapat diambil beberapa kesimpulan antara lain :

a) Perbaikan yang dilakukan dapat memperbaiki seluruh kesalahan *stemming* yang dilakukan oleh algoritma *ECS Stemmer*.
b) Untuk kesalahan *overstemming* dan *understemming*, di mana hasil stemming dari suatu term dapat berjumlah lebih dari satu, penggunaan metode *corpus based stemming* dapat digunakan untuk memilih hasil *stemming* yang tepat berdasarkan koleksi dokumen yang digunakan.
c) Penggunaan *data fusion* dan *metode condorcet* dapat mempersingkat waktu yang dibutuhkan untuk melakukan pembentukan *relevan set* dalam proses penilaian efektifitas sistem temu kembali informasi.

B. Saran

Saran untuk pengembangan lebih lanjut dari tugas akhir ini antara lain:

a) Dilakukan pengujian menggunakan koleksi dokumen yang berbeda untuk mengetahui efek dari metode *corpus based stemming* terhadap hasil dari proses *stemming* yang dilakukan.
b) Dilakukan percobaan terhadap parameter *data fusion* dan *metode Condorcet* dengan nilai yang lebih bervariasi untuk mengetahui konsistensi hasil efektifitas sistem temu kembali informasi.

REFERENSI

[1] Arifin, A.Z. dan A.N. Setiono. 2002. **Klasifikasi Dokumen Berita Kejadian Berbahasa Indonesia dengan Algoritma Single Pass Clustering**. *Proceeding of Seminar on Intelligent Technology and Its Applications (SITIA)*, Teknik Elektro, Institut Teknologi Sepuluh Nopember

[2] Arifin, A.Z., I.P.A.K. Mahendra dan H.T. Ciptaningtyas. 2009. ***Enhanced Confix Stripping Stemmer and Ants Algorithm for Classifying News Document in Indonesian Language***, *Proceeding of International Conference on Information & Communication Technology and Systems (ICTS)*

[3] Asian, J. 2007. ***Effective Techniques for Indonesian Text Retrieval***. *PhD Thesis.* School of Computer Science and Information Technology RMIT University Australia

[4] Larkey, L. S., Ballesteros, L., and Connell, M.E. 2002. ***Improving Stemming for Arabic Information Retrieval : Light Stemming and Co-occurrence Analysis***. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, August 11-15, Tampere, Finland

[5] Nuray, R. and Can, F. 2006. ***Automatic ranking of information retrieval systems using data fusion***. Information Processing and Management, 42, pp. 595-614

[6] Xu, J. and Croft, W. B. 1998. ***Corpus-based stemming using cooccurrence of word variants***. ACM Transactions on Information Systems, 16 (1), pp. 61-81.