Ekstraksi Kata Kunci Otomatis untuk Dokumen Berbahasa Indonesia menggunakan metode Genitor-plus-Extractor (GenEx)

Gregorius Satia Budhi[1], Agustinus Noertjahyana[2], Risky Yuniarto Susilo[3]
[1, 2, 3)] Teknik Informatika Universitas Kristen Petra Surabaya
E-Mail: greg@petra.ac.id, agust@petra.ac.id

Abstrak

Ekstraksi Kata Kunci Otomatis adalah sebuah aplikasi yang digunakan untuk menghasilkan sebuah daftar *keyphrase* / kata kunci secara otomatis. Algoritma utama yang digunakan adalah algortima *GenEx* yang dibuat oleh Turney, dengan beberapa penyesuaian karena digunakan untuk ekstraksi *keyphrase* dari artikel berbahasa Indonesia. Penyesuaian dilakukan pada proses *stemming* pada bagian *Extractor* dengan menggantinya mengunakan algoritma *Porter Stemmer for Bahasa Indonesia* yang dibuat oleh Tala. Penyesuaian perlu dilakukan karena kata dalam bahasa Indonesia memiliki tiga macam imbuhan (prefiks, infiks dan suffiks) sementara kata dalam bahasa Inggris hanya memiliki imbuhan dibelakang (suffiks). *GenEx* adalah algoritma yang memanfaatkan Algortima Genetika (*Genitor*) untuk membentuk sekelompok parameter yang digunakan saat mengekstrak kata kunci dari sebuah artikel didalam proses *Extractor*. Hasil pengujian nilai *recall* dari *keyphrase* yang di-*generate* terhadap kata kunci dari author bernilai rata - rata 60%. Sementara hasil pengujian oleh responden menunjukkan bahwa 95% responden menyatakan bila *keyphrase* yang di-*generate* dapat mewakili artikelnya. Kedua hasil menunjukkan bahwa aplikasi ini telah berhasil menggenerate kata kunci (*keyphrase*) yang sesuai dan dapat mewakili artikel yang diproses.

Kata kunci: Ekstraksi Kata Kunci Otomatis, Kata kunci, Algoritma *GenEx*, *Porter Stemmer for Bahasa Indonesia*, Dokumen Berbahasa Indonesia

Abstract

Automatic Keyword Extraction is an application used to generate a list of keyphrases / keywords automatically. The main algorithm that is used is GenEx by Turney, with some adjustments because it is used for keyphrase extraction from articles in Indonesian language. Adjustments are made on the process of stemming inside the Extractor part. It is replaced with "Porter Stemmer for Bahasa Indonesia" algorithm by Tala. Adjustments need to be made because the words in the Indonesian language has three kinds of affixes (prefix, infix and suffix) while the English word has only suffixes. GenEx is an algorithm that uses Genetic algorithms (Genitor) to form a group of parameters that are used when extracting keywords from an article in the Extractor. The test results of recall value for keyphrase that are generated by the application divided by the number of keyword author worth the average of 60%. While the results of testing by the respondents indicated that 95% of respondents said if the keyphrase that are generated could represent the article. Both results indicate that the application has been successfully generating the keywords (keyphrases) that are suitable and can represent the processed article.

Key words: Automatic Keyphrase Extraction, Keyphrase, GenEx algorithm, Porter Stemmer for Bahasa Indonesia, Indonesian Language Document

PENDAHULUAN

Proses pemberian daftar kata kunci *(keywords / keyphrases)* pada dokumen cukup sulit, terutama jika hal itu harus dilakukan kemudian oleh orang lain yang bukan pengarang dari dokumen, misal petugas perpustakaan *online* atau administrator web penyedia file dokumen makalah / *paper*. Oleh karena itu dibutuhkan suatu proses otomatis yang mampu mengekstraksi kata kunci secara langsung dari sebuah input dokumen. Disini kami menggunakan istilah *keyphrase* karena biasanya kata kunci diabuat dalama bentuk lebih dari 1 kata (frase).

Aplikasi yang dikembangkan pada penelitian ini adalah sebuah software yang dapat secara otomatis menghasilkan daftar kata kunci. Dimana daftar kata kunci tersebut mewakili poin-poin penting dari sebuah dokumen.

Algoritma yang digunakan dipenelitian ini adalah algoritma *GenEx (Genitor Plus Extractor)* yang telah disesuaikan agar tepat untuk ekstraksi kata kunci dari dokumen berbahasa indonesia. Penyesuaian dilakukan pada tahap *stemming* dari proses *Extractor*. Metode "*Stemming by Truncation"* pada proses ini diganti dengan metode *"Porter Stemmer for Bahasa Indonesia"* yang dibuat oleh F.Z. Tala pada tahun 2003.

TINJAUAN PUSTAKA

Text Mining

Text Mining dapat didefinisikan sebagai metode atau teknik komputasi pada data berbentuk teks guna menemukan informasi yang relevan, intrinsik dan tidak diketahui sebelumnya. Metode text mining dikelompokkan dalam empat kategori yaitu: *clasification, clustering, association analysis* dan *information extraction* [7].

Automatic Keyphrase Extraction

Automatic keyphrase extraction adalah sebuah proses untuk menghasilkan daftar *keyphrase* yang dapat mewakili poin-poin penting dari sebuah teks. *Automatic keyphrase extraction* adalah sebuah bentuk implementasi dari *text mining.* Proses kerja dari *Automatic keyphrase extraction* secara umum sama dengan proses kerja didalam *text mining* [1].

Algoritma genetika *Genitor*

Genitor yang merupakan akronim dari *Genetic ImplemenTOR*, adalah algoritma genetika *steady-state*, yang berbeda dengan banyak algoritma genetika lainnya. Algoritma genetika *steady-state* memperbaharui populasi hanya satu individu di satu waktu, mengakibatkan perubahan populasi yang terus menerus, dengan tidak ada beda generasi. Biasanya individu baru dengan fitnes terbaik menggantikan individu dengan fitnes terrendah [2]. *Pseudo-code* dari *Genitor* dapat dilihat pada Gambar 1.

```
GENITOR (P)
create population of size P
while not (termination condition)
        select two parents
        breeds a single offspring by
        (optional) crossover followed by
        mutation O
        evaluate least-fit chromosome by O
        replace the least-fit member of the
population
output fittest kromosom
```

Gambar 1. *Pseudo-code Genitor* [2]

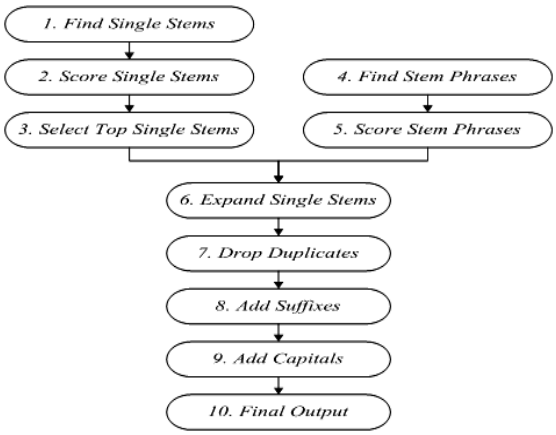
Algoritma ekstraksi *keyphrase Extractor*

Extractor akan mengambil sebuah dokumen sebagai input dan menghasilkan daftar *keyphrase* sebagai output. *Extractor* mempunyai dua belas parameter yang akan mempengaruhi pemrosesan dokumen teks input [1]. Parameter - parameter tersebut dapat dilihat pada Tabel 1.

Tabel 1. Daftar Parameter pada *Extractor* [1]

No	Nama Parameter	Nilai Awal	Deskripsi
1	NUM_PHRASES	10	Maks frase di daftar frase akhir
2	NUM_WORKING	50	5 * NUM_PHRASES
3	FACTOR_TWO_ONE	2,33	Faktor dua kata dalam satu frase
4	FACTOR_THREE_ONE	5	Faktor tiga kata dalam satu frase
5	MIN_LENGTH_LOW_RANK	0,9	Peringkat yang rendah harus memiliki Kata yang lebih penjang dari nilai ini
6	MIN_RANK_LOW_LENGTH	5	Kata yang pendek harus memiliki peringkat lebih dari nilai ini
7	FIRST_LOW_THRESH	40	Definisi posisi kata yang ditemukan lebih awal
8	FIRST_HIGH_THRESH	400	Definisi posisi kata yang ditemukan lebih akhir
9	FIRST_LOW_FACTOR	2	Penghargaan untuk kata yang ditemukan lebih awal
10	FIRST_HIGH_FACTOR	0,65	Penalti untuk kata yang ditemukan lebih akhir
11	STEM_LENGTH	5	Maksimum karakter panjang *stem*
12	SUPRESS_PROPER	0	Flag untuk *suppres proper nouns*

Di dalam *Extractor* terdapat sepuluh langkah seperti terlihat pada Gambar 2.



Gambar 2. Langkah - langkah proses algoritma *Extractor* [1]

Algoritma *GenEx*

Algoritma *GenEx* adalah algoritma *hybrid genetic* yang digunakan untuk mengekstraksi kata kunci. *GenEx* memiliki dua komponen, yaitu algoritma genetika *Genitor* dan algoritma ektraksi *keyphrase Extractor*. *Extractor* akan mengambil sebuah dokumen sebagai *input* dan menghasilkan daftar *keyphrase* sebagai *output*. *Extractor* mempunyai dua belas parameter yang akan mempengaruhi pemrosesan teks input. Parameter dari *Extractor* ini diatur menggunakan algoritma genetika *Genitor*. Genitor tidak dibutuhkan lagi ketika proses training telah selesai yaitu ketika nilai parameter terbaik telah diketahui. Sementara itu proses untuk mendapatkan *keyphrase* saat testing atau pemakaian digunakan *Extractor* (*GenEx without Genitor*).

Algoritma *GenEx* menggunakan *dataset* (*training* dan *testing subset*) berupa pasangan dokumen dengan target *keyphrase*-nya. Proses *learning* yang ada di algoritma *GenEx* bertujuan untuk menyesuaikan parameter menggunakan data *training* yang berguna untuk memaksimalkan kesamaan hasil antara output dari *Extractor* dengan daftar target *keyphrase*. Keberhasilan proses *learning* diukur dengan memeriksa kecocokan menggunakan data *testing*.

Jika diasumsikan bahwa *user* menentukan nilai NUM_PHRASES, jumlah frase yang dikehendaki ke nilai antara lima dan lima belas. Kemudian diatur NUM_WORKING adalah 5 * NUM_PHRASES. Menyisakan sepuluh parameter yang akan ditetapkan oleh *Genitor*. *Genitor* menggunakan *binary string* 72 bit untuk mewakili sepuluh parameter, seperti yang ditunjukkan pada Tabel 2. *Genitor* dijalankan dengan ukuran populasi 50 sampai 1050 percobaan / individu (nilai default). Setiap individu adalah parameter pengaturan *Extractor* yang ditentukan dalam *binary string*. *Fitness cost* tiap *binary string* dihtung menggunakan rumus 1 sampai dengan 7 untuk seluruh *training set*. Output akhir dari *Genitor* adalah nilai tertinggi *binary string*. Output ini nantinya digunakan pada proses *Extractor* di saat pemakaian aplikasi [3].

Perhitungan *fitness* cost didapatkan dari rumus perhitungan di bawah ini [3]:

total_matches = jumlah frasa yang sama antara *GenEx* dan manusia (*author*) (1)

total_machine_phrases = jumlah *output* frasa yang dihasilkan dari proses *GenEx* (2)

precision = *total_matches* / *total_machine_phrases* (3)

num_docs = jumlah dokumen yang digunakan sebagai *training set* (4)

total_desired = *num_docs* * NUM_PHRASES (5)

penalty = (*total_machine_phrases* / *total_desired*)^2 (6)

fitness = *precision* * *penalty* (7)

Tabel 2. Sepuluh Parameter *Extractor* yang diatur oleh *Genitor* [3]

Nama Parameter	Tipe	Range	Jml Bit
FACTOR_TWO_ONE	real	[1, 3]	8
FACTOR_THREE_ONE	real	[1, 5]	8
MIN_LENGTH_LOW_RANK	real	[0.3, 3.0]	8
MIN_RANK_LOW_LENGTH	integer	[1, 20]	5
FIRST_LOW_THRESH	integer	[1, 1000]	10
FIRST_HIGH_THRESH	integer	[1, 4000]	12
FIRST_LOW_FACTOR	real	[1, 15]	8
FIRST_HIGH_FACTOR	real	[0.01, 1.0]	8
STEM_LENGTH	integer	[1, 10]	4
SUPRESS_PROPER	boolean	[0, 1]	1
Total Panjang Bit dalam Binary String:			72

Faktor *penalty* bervariasi antara 0 dan 1. Faktor *penalty* tidak mempunyai efek (jika 1) ketika jumlah frase yang keluar sebagai output dari *GenEx* sama dengan jumlah frase yang diinginkan. *Penalty* didapat (mendekati 0) dari pangkat hasil pembagian antara jumlah frase yang keluar sebagai output dari *GenEx* dengan jumlah frase yang diinginkan. Eksperimen *preliminary* pada data *training* menetapkan ukuran *fitness* yang membantu *GenEx* untuk menemukan nilai parameter dengan rata-rata ketepatan yang tinggi dan kepastian frase

sebanyak NUM_PHRASES akan keluar sebagai *output*.

Genitor dijalankan dengan *Selection Bias of* 2.0 dan *Mutation Rate of* 0.2. Ini adalah pengaturan *default* untuk *Genitor*. PAda Genitor digunakan *Adaptive Mutation Operator* dan *Reduced Surrogate Crossover Operator*. *Adaptive Mutation* menentukan tingkatan mutasi yang sesuai untuk seorang *children* menurut *hamming distance* antara kedua *parent*; semakin sedikit perbedaan, lebih tinggi tingkat *mutation rate*. *Reduced Surrogate Crossover* pertama mengidentifikasi semua posisi di mana terjadi perbedaan pada *parent string*. Poin-poin *crossover* hanya diijinkan untuk terjadi pada posisi ini [3].

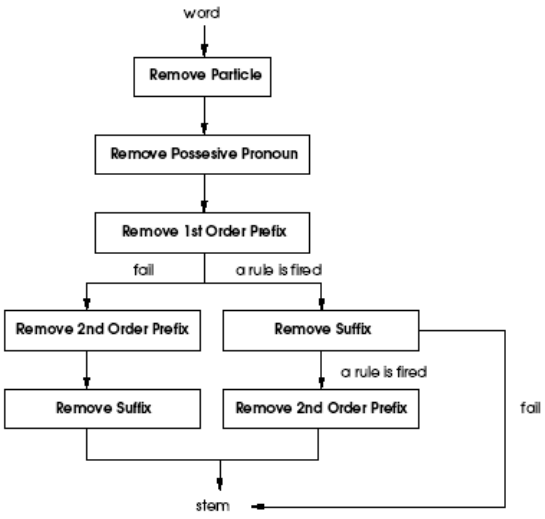
Stemming

Stemming adalah proses pemetaan dan penguraian berbagai bentuk (*variants*) dari suatu kata menjadi bentuk kata dasarnya [4]. Proses ini juga disebut sebagai *conflation.* Proses *Stemming* membuat bentuk sebuah kata menjadi kata dasarnya. Metode umum untuk *stemming* adalah menggunakan kombinasi dari analisa morphological [8].

Porter Stemmer for Bahasa Indonesia

Porter Stemmer for Bahasa Indonesia dikembangkan oleh Fadillah Z. Tala pada tahun 2003. Implementasi *Porter Stemmer for Bahasa Indonesia* berdasarkan English *Porter Stemmer* yang dikembangkan oleh M.F. Porter pada tahun 1980. Karena bahasa Inggris datang dari kelas yang berbeda, beberapa modifikasi telah dilakukan untuk membuat Algoritma *Porter* dapat digunakan sesuai dengan bahasa Indonesia [5]. Desain dari *Porter Stemmer for Bahasa Indonesia* dapat dilihat pada Gambar 3.

Implementasi dari algoritma ini telah dioptimasi untuk digunakan pada *text mining* oleh penulis dan hasilnya telah dipublikasi sebelumnya [6].



Gambar 3. Blok diagram dari *Porter Stemmer for Bahasa Indonesia* [5]

Pada Gambar 3 terlihat beberapa langkah *'removal'* menurut aturan yang ada pada tabel 3 sampai dengan tabel 7.

Tabel 3. Kelompok *rule* pertama : *inflectional particles* [5]

Suffix	Replacement	Measure Condition	Additional Condition	Examples
kah	NULL	2	NULL	bukukah → buku
lah	NULL	2	NULL	adalah → ada
pun	NULL	2	NULL	bukupun → buku

Tabel 4. Kelompok *rule* kedua :*inflectional possesive pronouns* [5]

Suffix	Replacement	Measure Condition	Additional Condition	Examples
ku	NULL	2	NULL	bukuku → buku
mu	NULL	2	NULL	bukumu → buku
nya	NULL	2	NULL	bukunya → buku

Tabel 5. Kelompok *rule* ketiga: *first order of derivational prefixes* [5]

Prefix	Replacement	Measure Condition	Additional Condition	Examples
meng	NULL	2	NULL	mengukur → ukur
meny	s	2	V…*	menyapu → sapu
men	NULL	2	NULL	menduga → duga menuduh → uduh
mem	p		V…	memilah → pilah
mem	NULL	2	NULL	membaca → baca
me	NULL	2	NULL	merusak → rusak
peng	NULL	2	NULL	pengukur → ukur
peny	s	2	V…	penyapu → sapu
pen	NULL	2	NULL	penduga → duga penuduh → uduh
pem	p	2	V…	pemilah → pilah
pem	NULL	2	NULL	pembaca → baca
di	NULL	2	NULL	diukur → ukur
ter	NULL	2	NULL	tersapu → sapu
ke	NULL	2	NULL	kekasih → kasih

Tabel 6. Kelompok *rule* keempat: *second order of derivational prefixes* [5]

Prefix	Replacement	Measure Condition	Additional Condition	Examples
ber	NULL	2	NULL	berlari → lari
bel	NULL	2	Ajar	belajar → ajar
be	NULL	2	K* er…	bekerja → kerja
per	NULL	2	NULL	perjelas → jelas
pel	NULL	2	Ajar	pelajar → ajar
pe	NULL	2	NULL	pekerja → kerja

Tabel 7: Kelompok *rule* kelima: *derivational suffixes* [5]

Suffix	Replacement	Measure Condition	Additional Condition	Examples
kan	NULL	2	Prefix ∉ {ke, peng}	tarikkan → tarik (meng)ambilkan → ambil
an	NULL	2	Prefix ∉ {di, meng, ter}	makanan → makan (per)janjian → janji
i	NULL	2	V|K…c1c1, c1 ≠ s, s2 ≠ i and prefix ∉ {ber, ke, peng}	tandai → tanda (men)dapati → dapat pantai → panta

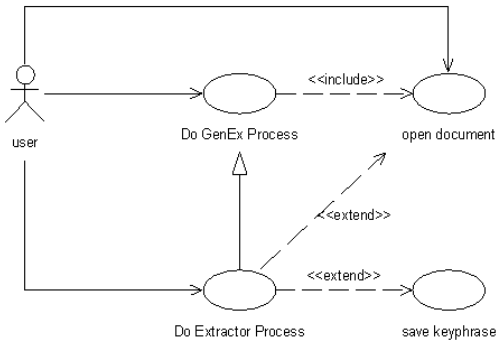
Frase dalam bahasa Indonesia

Frase adalah bagian kalimat yang terdiri atas dua kata atau lebih yang tidak melebihi batas fungsi. Misalnya: akan datang, kemarin pagi, yang sedang menulis. Artinya satu frase maksimal hanya menduduki gatra subjek (S), predikat (P) atau objek (O) atau keterangan (K).

Perbedaan antara frase dalam Bahasa Indonesia dengan frase dalam Bahasa Inggris bukanlah perbedaan yang mencolok dan tanpa pola [9].

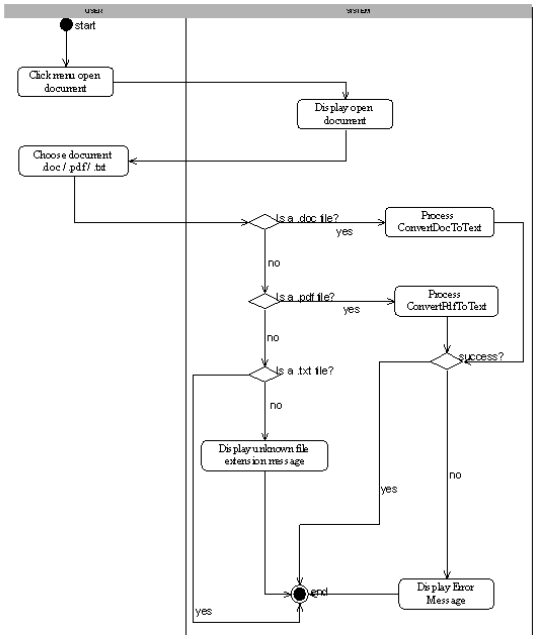
DISAIN APLIKASI

Berikut pada Gambar 4 dapat dilihat *use case diagram* dari aplikasi yang dibuat.



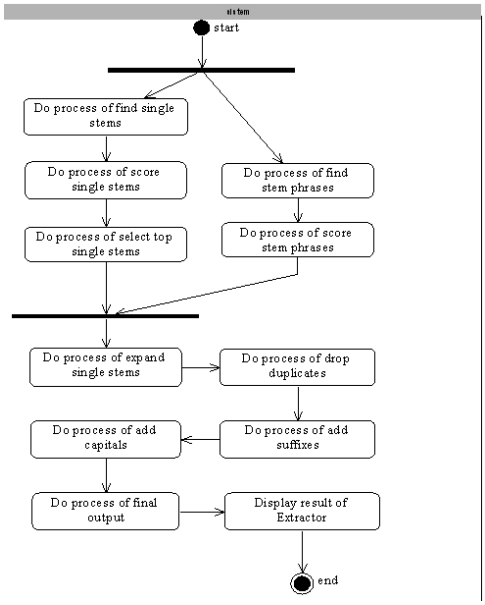
Gambar 4. Diagram *Use Case* dari aplikasi Ekstraksi Kata Kunci Otomatis

Berikut pada Gambar 5 sampai dengan Gambar 8 dapat dilihat diagram *activity* dari masing - masing *use case*.



Gambar 5. Diagram *activity* dari *use case Open Document*

Open document memungkinkan *user* untuk melihat isi dokumen atau memilih dokumen yang ingin dicari *keyphrase*-nya. Proses open document akan mentransfer dokumen berbentuk *,doc / *.pdf untuk menjadi teks. Teks yang terbentuk akan dikenai proses selanjutnya.



Gambar 6. Diagram *activity* dari *use case Do Extractor Process*

Do Extractor Process adalah implementasi dari algoritma *Extractor*. Proses yang dilakukan kurang lebih sama dengan sepuluh langkah pada algoritma *Extractor*, namun dengan sedikit penyesuaian karena digunakan untuk mengekstrak dokumen berbahasa Indonesia. Langkah - langkah tersebut adalah sebagai berikut:

Langkah 1: *Find Single Stems*

Hal pertama yang dilakukan pada langkah ini adalah membuat daftar kata-kata dari teks input. Jika kata kurang dari 3 huruf maka hapus kata tersebut kecuali kata memiliki kapitalisasi pattern stem yang menunjukkan sebuah singkatan. Penghapusan ini dilakukan dengan asumsi bahwa kata kurang dari 3 huruf tidak memiliki arti penting. Setelah itu hapus stopword menggunakan daftar *stopword* yang disediakan pada *paper* dari Tala [5]. Proses selanjutnya adalah *Stemming*. Di dalam algoritma *GenEx, stemming* dilakukan dengan cara memotong kata sesuai dengan nilai STEM_LENGTH (metode *stemming by truncation*) dengan tujuan mempercepat proses. Namun metode ini tidak cocok bila diterapkan untuk bahasa Indonesia. Alasannya adalah karena kata dalam bahasa Indonesia memiliki imbuhan di depan (prefiks), belakang (suffiks), sisipan (infiks) atau gabungan ketiganya, sementara kata dalam bahasa inggris hanya memiliki imbuhan dibelakang (suffiks). Oleh sebab itu proses stemming pada langkah ini dirubah menggunakan algoritma *"Porter Stemmer for Bahasa Indonesia"* [5] yang kemudian telah dioptimasi oleh penulis untuk diimplementasikan pada text mining [6].

Langkah 2: Score Single Stems

Pada langkah ini dihitung seberapa sering sebuah stem tunggal ada di dalam teks, dan catat dimana kata tersebut pertama kali ditemukan. Setelah itu, berikan skor untuk setiap stem tunggal. Skor adalah jumlah berapa kali stem tunggal tersebut muncul di dalam teks dikalikan dengan sebuah faktor. Jika stem tunggal pertama kali ditemukan sebelum FIRST_LOW_THRESH, maka kalikan frekuensi tersebut dengan FIRST_ LOW_FACTOR. Jika stem tunggal pertama kali ditemukan sesudah FIRST_HIGH_ THRESH, maka kalikan frekuensi tersebut dengan FIRST_HIGH_FACTOR.

Langkah 3: Select Top Single Stems

Langkah ketiga adalah merangking nilai tiap stem tunggal dari skor tinggi ke skor rendah dan buat daftar top stem tunggal sebanyak NUM_WORKING.

Pomotongan daftar stem tunggal sebanyak NUM_WORKING untuk mengatasi agar stem tunggal di dalam daftar tidak terlalu banyak sehingga dapat meningkatkan efisiensi Extractor. Pemotongan juga sebagai penyaring untuk menghapuskan stem tunggal dengan kualitas rendah.

Langkah 4: Find Stem Phrases

Pada langkah ini dibuat daftar semua frase yang ada di input teks. Sebuah frase didefinisikan sebagai kumpulan dari satu, dua, atau tiga kata yang teratur di dalam teks, dan tidak terdapat stopword atau batasan frase (tanda baca). Setelah itu dilakukan *stemming* setiap frase dengan memotong tiap kata di dalam frase menjadi *root word*.

Extractor hanya mempertimbangkan frase dengan satu, dua, atau tiga kata karena Frase dengan empat kata atau lebih sangat jarang. Extractor tidak mempertimbangkan frase dengan stopword didalamnya, karena author cenderung menghindari keyphrase dengan stopword didalamnya.

Langkah 5: Score Stem Phrases

Pada langkah kelima dilakukan perhitungan seberapa sering setiap stem frase muncul di teks dan catat dimana stem frase tersebut pertama kali muncul. Tetapkan skor untuk setiap stem frase, dengan cara yang sama dengan langkah kedua, dengan menggunakan parameter FIRST_LOW_FACTOR, FIRST_ LOW_THRESH, FIRST_HIGH_FACTOR, dan FIRST_HIGH_THRESH. Selanjutnya buat penyesuaian untuk setiap skor, berdasarkan jumlah stem kata dari tiap frase. Jika hanya ada satu stem tunggal di dalam frase, maka tidak dilakukan apa-apa. Jika terdapat dua stem tunggal dalam frase, kalikan skor dengan FACTOR_TWO_ONE. Jika terdapat tiga stem tunggal dalam frase, kalikan skor dengan FACTOR_THREE_ONE.

Langkah 6: Expand Single Stems

Pada langkah keenam, untuk setiap stem dalam daftar top stem tunggal sebanyak NUM_WORKING dicari nilai tertinggi stem frase dari satu, dua, atau tiga stem yang berisi stem tunggal tersebut. Hasilnya adalah daftar

stem frase sebanyak NUM_WORKING. Simpan daftar ini dan susun sesuai dengan ranking skor yang dihitung di langkah 2.

Langkah 7: Drop Duplicates

Pada langkah ketujuh dilakukan penghapusan stem frase yang memiliki duplikat pada daftar top NUM_WORKING. Misalnya, dua stem tunggal mungkin diperluas menjadi stem frase yang terdiri dari dua kata. Hapus duplikat dari daftar peringkat NUM_WORKING stem frase, dan pertahankan peringkat tertinggi frase.

Langkah 8: Add Suffixes

Untuk masing-masing dari sisa stem frase, temukan frase yang paling sering muncul di seluruh input teks. Misalnya, jika "penggalian data" sepuluh kali muncul dalam teks dan "menggali data" muncul tiga kali, maka "penggalian data" lebih sering sesuai untuk stem frase "gali data". Di dalam bahasa Indonesia kata dapat memiliki prefiks, suffiks, atau kombinasi dari keduanya (*confixes*). Maka dari itu pada langkah ini kata dikembalikan ke bentuk awal, yaitu dengan penambahan *prefiks, suffiks*, atau kombinasi dari keduanya (confixes). Jadi sebenarnya langkah ini lebih tepat bila dinamakan *"Add Affixes"*.

Di dalam metode automatic keyphrase extractor dokumen berbahasa Inggris jika ada frase yang memiliki kata dengan suffiks, yang menunjukkan kemungkinan menjadi kata sifat (*adjective*), maka frekuensi untuk seluruh frase diatur ke nol. Juga, jika ada frase yang memiliki kata yang menunjukkan kemungkinan kata kerja (*verb*), frekuensi untuk frase diatur ke nol.

Pada bahasa Indonesia teknik untuk mendeteksi kemungkinan kata sifat dan kata kerja berbeda dengan dalam bahasa Inggris, Oleh sebab itu didalam aplikasi ini digunakan tabel kata sifat dan kata kerja yang terdapat pada database kamus elektronik Indonesia. Bila ditemukan maka frekuensi dari frase-nya diatur ke nol.

Langkah 9: Add Capitals

Dilangkah kesembilan dicari kemungkinan terbaik untuk proses kapitalisasi pada seluruh frase yang ada. Istilah terbaik didefinisikan sebagai berikut: Untuk setiap frase, temukan proses kapitalisasi dengan jumlah kapital paling sedikit. Dari aturan ini kandidat terbaik adalah frase yang hanya berisi satu kata. Untuk frase yang terdiri dari dua atau tiga kata, disebut terbaik (best) di dalam kapitalisasi jika konsisten.

Langkah 10: Final Output

Pada langkah sepuluh diambil sejumlah frase yang telah dirangking sebayak NUM_WORKING dari rangking tertinggi. Selanjutnya adalah menampilkan daftar output, frase sebanyak NUM_PHRASES yang paling baik. Untuk mendapatkan hal itu setiap frase akan melewati tes berikut:

(1) frase tidak memiliki kapitalisasi dari kata benda, kecuali flag SUPPRESS_PROPER diatur ke 1 (jika 1 kemungkinan kata benda).

(2) frase sebaiknya tidak memiliki kemungkinan yang menunjukkan kata sifat.

(3) frase harus lebih panjang daripada MIN_LENGTH_LOW_RANK, dimana panjang diukur oleh rasio jumlah karakter dalam kandidat frase dengan jumlah karakter di rata-rata frase, dimana rata-rata dihitung untuk semua frase di input teks yang terdiri dari satu sampai tiga kata non-*stopword* yang berturut-turut.

(4) jika lebih pendek dari frase MIN_LENGTH_LOW_RANK, maka frase tersebut masih dapat diterima, jika peringkatnya di daftar kandidat frase lebih baik daripada MIN_RANK_LOW_LENGTH.

(5) jika kata gagal di kedua tes (3) dan (4), ia masih dapat diterima, jika kapitalisasi pattern stem menunjukkan bahwa mungkin sebuah singkatan.

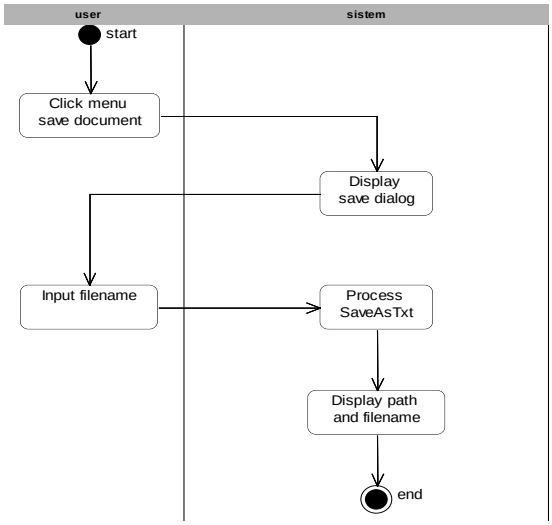
(6) frase tidak boleh mengandung apapun kata yang sering digunakan sebagai kata kerja.

(7) frase seharusnya tidak sama dengan isi daftar *stop-phrase*.

Frase yang ditampilkan harus lulus tes (1), (2), (6), (7), dan sekurang-kurangnya salah satu tes (3), (4), dan (5). Bila jumlah frase yang lolos tes lebih banyak dari jumlah NUM_PHRASE maka hanya frase rangking 1 sampai dengnan NUM_PHRASE yang ditampilkan.

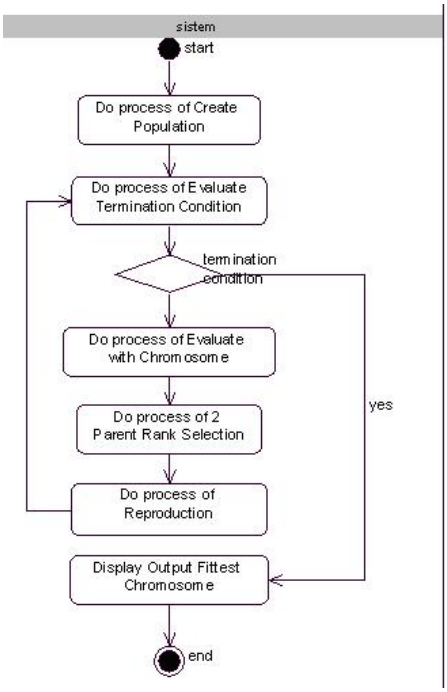
Langkah 11: Display Result

Pada langkah ini hasil dari langkah 10 ditampilkan ke pada user. Selanjutnya user dapat memilih *keyphrase* mana yang digunakan dan mana yang dihapus.



Gambar 7. Diagram *activity* dari *use case Save Keyphrase*

Pada aktifitas ini hasil *keyphrase* yang telah dipilih oleh user beserta dokumennya disimpan dengan bentuk file text (*.txt).

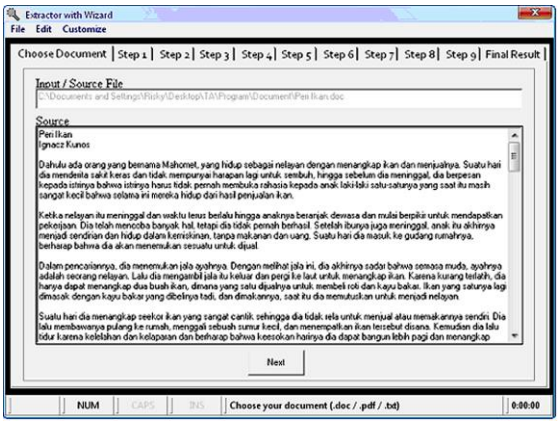


Gambar 8. Diagram *activity* dari *use case Do GenEx Process*

Pada aktifitas *Do GenEx Process* dilakukan proses yang sama persis dengan proses *GenEx* yang telah dijelaskan sebelumnya pada bab Tinjauan Pustaka.

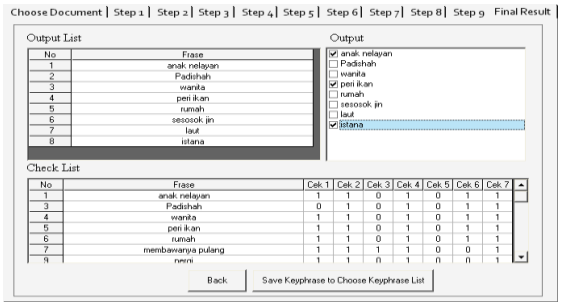
ANTAR MUKA APLIKASI

Pada Gambar 9 dilihat tampilan antar muka awal dari aplikasi yang dibuat di penelitian ini.



Gambar 9. Antar muka awal setelah proses open dokumen dilakukan oleh user

Setelah melewati 9 step yang ada dengan cara menekan tombol '*next*' maka kepada user akan disajikan hasil dari ekstraksi otomatis kata kunci dari dokumen yang diproses. Tampilan antar muka *'Final Result'* dapat dilihat pada Gambar 10.



Gambar 10. Antar muka *'Final Result'*

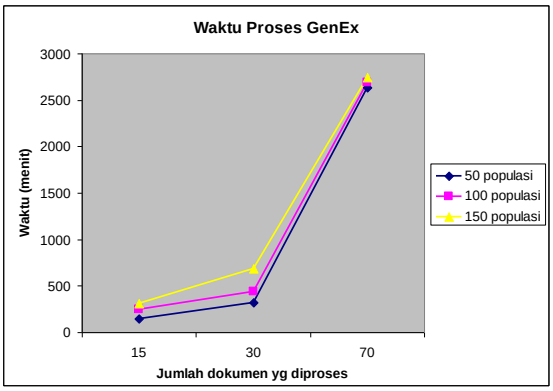
Pada antar muka *'Final Result'* akan ditampilkan hasil output yang dapat dipilih oleh user (*box Output*). Pada antar muka ini ditampilkan pula frase - frase yang lolos dari test pada langkah 10 *Extractor* (*box Output List*) dan juga informasi detail tentang hasil test dari urutan frase sebanyak NUM_WORKING (*box Check List*).

PENGUJIAN

Ada tiga macam pengujian yang dilakukan pada penelitian ini, yaitu:

1. Pengujian waktu proses learning *GenEx*. Pengujian ini dilakukan saat *Genitor* bekerja untuk mengoptimasi parameter *Extractor*. Sementara untuk proses

Extractor karena sangat cepat maka tidak perlu diuji. Hasil pengujian dapat dilihat pada Gambar 11.



Gambar 11. Hasil pengujian waktu proses GenEx

Dari hasil pengujian terlihat bahwa semakin banyak jumlah dokumen sample yang ikut dalam proses learning *GenEx* dan semakin besar populasi, maka semakin lama pula waktu prosesnya. Namun waktu proses learning ini masih dibawah 1 jam untuk 70 dokumen dengan maks 150 populasi (10500 kali proses).

2. Pengujian perbandingan hasil yang didapat dengan parameter default yang didapat dari paper PD Turney [1] dengan nilai parameter yang didapat dari proses *learning GenEx*. Pada pengujian ini digunakan 70 dokumen untuk *learning GenEx* dan 5 dokumen untuk *testing*. Hasil pengujian dapat dilihat pada Tabel 8 dan Tabel 9.

Tabel 8. Perbandingan precision antara hasil Extractor dengan parameter *default* dan parameter hasil *learning GenEx*

Judul Dokumen	*Precision* utk *Default* Parameter	*Precision* utk Parameter hasil GenEx
Analisis Persepsi Pelanggan Terhadap Kualitas Layanan Coffee Shop Asing Dan Coffee Shop Lokal	0.125	0.125
Representasi Perempuan Pada Lukisan Di Bak Truk	0.375	0.5
Desain Interior Dan Perilaku Pengunjung Di Ruang Publik	0.25	0.25
Penerapan Fuzzy If-Then Rules Untuk Peningkatan Kontras Pada Citra Hasil Mammografi	0.25	0.25
Perilaku Dan Keputusan Pembelian Konsumen Restoran Melalui Stimulus 50% Discount Di Surabaya	0.25	0.375
Rata - rata Precision	**0.25**	**0.3**

Nilai *precision* pada tabel 8 dihitung dengan rumus berikut:

Precision = jumlah hasil benar / total hasil proses (8)

Nilai *precision* yang tidak besar disebabkan karena besarnya nilai pembagi. Pada aplikasi ini nilai pembagi tersebut identik dengan isi variable NUM_PHRASES yang disetting oleh user. Oleh sebab itu dihitung pula nilai Recall menggunakan rumus sebagai berikut:

Recall = jumlah hasil benar / total keyphrase author (9)

Tabel 9. Perbandingan recall antara hasil Extractor dengan parameter *default* dan parameter hasil *learning GenEx*

Judul Dokumen	*Recall* utk *Default* Parameter	*Recall* utk Parameter hasil *GenEx*
Analisis Persepsi Pelanggan Terhadap Kualitas Layanan Coffee Shop Asing Dan Coffee Shop Lokal	0.33	0.33
Representasi Perempuan Pada Lukisan Di Bak Truk	0.5	0.75
Desain Interior Dan Perilaku Pengunjung Di Ruang Publik	0.66	0.66
Penerapan Fuzzy If-Then Rules Untuk Peningkatan Kontras Pada Citra Hasil Mammografi	0.5	0.5
Perilaku Dan Keputusan Pembelian Konsumen Restoran Melalui Stimulus 50% Discount Di Surabaya	0.5	0.75
Rata - rata Recall	**0.5**	**0.6**

Pada Tabel 9 dapat dilihat bahwa nilai recall dari kedua macam parameter yang diteliti cukup baik karena rata - rata setengah atau lebih dari hasil yang benar telah sesuai dengan kata kunci dari pengarang / *author*. Kelemahan dari aplikasi ini adalah *keyphrase* yang dihasilkan berasal dari frase - frase yang memang ada pada dokumen, sementara *author* dapat membuat kata kunci (*keyphrase*) dari frase diluar yang ada pada karangannya.

Pada dua pengujian terlihat bahwa parameter yang didapat dari *learning* menggunakan *GenEx* memiliki nilai rata - rata *precision* dan *recall* yang lebih baik.

3. Pengujian terhadap responden. Pengujian ini dilakukan dengan cara meminta 20 responden untuk membaca artikel berbahasa Indonesia lalu memberi jawaban apakah kata

kunci yang ada telah mewakili artikel dan memiliki arti yang jelas. Kata kunci tersebut dibuat secara otomatis oleh aplikasi. Selanjutnya respoden juga menjawab pertanyaan tentang daftar kata kunci mana yang lebih tepat. Hasil dapat dilihat pada Tabel 10.

Tabel 10. Pengujian tehadap responden

No	Mewakili Artikel	Arti Jelas	Daftar Keyphrase	
			default	hasil GenEx
1	Ya	Tidak	-	V
2	Ya	Ya	-	V
3	Ya	Ya	-	V
4	Ya	Ya	-	V
5	Ya	Ya	-	V
6	Ya	Tidak	-	V
7	Ya	Ya	V	-
8	Ya	Ya	-	V
9	Ya	Ya	V	-
10	Ya	Ya	-	V
11	Ya	Ya	-	V
12	Ya	Ya	-	V
13	Ya	Ya	-	V
14	Ya	Ya	-	V
15	Ya	Ya	-	V
16	Ya	Ya	-	V
17	Tidak	Tidak	-	V
18	Ya	Ya	-	V
19	Ya	Ya	-	V
20	Ya	Ya	-	V
	95%	85%	10%	90%

Dari pengujian dapat dilihat bahwa hampir semua responden beranggapan bahwa keyphrase yang dihasilkan aplikasi memiliki arti yang jelas dan dapat mewakili artikel. Responden juga berpendapat bahwa kata kunci yang digenerate dengan parameter hasil learning GenEx lebih baik dari parameter default.

KESIMPULAN

Dari pengujian dapat dilihat bahwa nilai recall dari aplikasi cukup baik (60%), sehingga dapat disimpulkan bahwa keyphrase yang dihasilkan aplikasi dapat membantu user untuk membuat kata kunci dari sebuah artikel. Hal ini didukung pula dengan hasil dari pengujian responden dimana 95% responden beranggapan bahwa keyphrase yang digenerate oleh aplikasi dapat mewakili artikelnya. Selain itu dari uji kecepatan proses GenEx, dapat disimpulkan bahwa proses GenEx masih dapat ditolerir waktunya mengingat proses ini hanya dilakukan sekali saja untuk kurun waktu lama. Proses GenEx baru dilakukan lagi bila ada penambahan jumlah dokumen sample yang cukup signifikan dan dianggap dapat merubah parameter yang dihasilkan.

DAFTAR PUSTAKA

Jurnal / Prosiding / Makalah Ilmiah:

[1] Turney PD. Learning to Extract Keyphrases from Text. National Research Council, Institute for Information Technology, Technical Report ERB-1057. 1999.

[2] Whitley D. The GENITOR algorithm and selective pressure. Proceedings of the Third International Conference on Genetic Algorithms (ICGA-89): 116-121. California: Morgan Kaufmann. 1989.

[3] Turney PD. Learning algorithms for keyphrase extraction. Information Retrieval, 2 (4): 303-336. (NRC #44105). 2000.

[4] Porter MF. An algorithm for suffix stripping. Program, 14(3): 130-137. 1980.

[5] Tala FZ. A Study of Stemming Effects on Information Retrieval in Bahasa Indonesia. Institute for Logic, Language and Computation Universeit Van Amsterdam. 2003.

[6] Budhi GS., Gunawan I., Yuwono F. Algortima Porter Stemmer For Bahasa Indonesia Untuk Pre-Processing Text Mining Berbasis Metode Market Basket Analysis. PAKAR Jurnal Teknologi Informasi Dan Bisnis vol. 7 no. 3 November 2006.

Buku :

[7] Prado HA, Ferneda E. Emerging Technologies of Text Mining: Techniques and Applications. New York: Information Science Reference. 2008.

[8] Chakrabarti S. Mining the Web: Discovering Knowledge from Hypertext Data. Morgan Kaufmann Publishers. 2003.

[9] Depdiknas. Panduan materi bahasa dan sastra Indonesia SMK. Depdiknas. 2003.