**IMPLEMENTASI ALGORTIMA *PHRASE RANK* PADA SISTEM *INFORMATION RETRIEVAL***

**Laporan Tugas Akhir**

**Disusun sebagai syarat kelulusan tingkat sarjana**

**Oleh**

**Mochamad Lutfi Fadlan**

**NIM : 13512087**



**PROGRAM STUDI TEKNIK INFORMATIKA**

**SEKOLAH TEKNIK ELEKTRO & INFORMATIKA**

**INSTITUT TEKNOLOGI BANDUNG**

**Agustus 2016**

**IMPLEMENTASI ALGORITMA PHRASE RANK PADA SISTEM *INFORMATION RETRIEVAL***

**Laporan Tugas Akhir**

**Oleh**

**Mochamad Lutfi Fadlan**

**NIM : 13512087**

**Program Studi Teknik Informatika**

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

Telah disetujui dan disahkan sebagai Laporan Tugas Akhir

di Bandung, pada tanggal 5 Agustus 2016

Pembimbing,

Dr. Ir. Rila Mandala, M.Eng

NIP. 19680803 199302 1 001

**LEMBAR PERNYATAAN**

Dengan ini saya menyatakan bahwa:

1. Pengerjaan dan penulisan Laporan Tugas Akhir ini dilakukan tanpa menggunakan bantuan yang tidak dibenarkan.
2. Segala bentuk kutipan dan acuan terhadap tulisan orang lain yang digunakan di dalam penyusunan laporan tugas akhir ini telah dituliskan dengan baik dan benar.
3. Laporan Tugas Akhir ini belum pernah diajukan pada program pendidikan di perguruan tinggi mana pun.

Jika terbukti melanggar hal-hal di atas, saya bersedia dikenakan sanksi sesuai dengan Peraturan Akademik dan Kemahasiswaan Institut Teknologi Bandung bagian Penegakan Norma Akademik dan Kemahasiswaan khususnya Pasal 2.1 dan Pasal 2.2.

Bandung, 12 Agustus 2016

Mochamad Lutfi Fadlan

NIM 13512087

ABSTRAK

**IMPLEMENTASI ALGORITMA *PHRASE RANK* PADA SISTEM *INFORMATION RETRIEVAL***

Oleh

Mochamad Lutfi Fadlan

NIM : 13512087

TBD.

KATA PENGANTAR

TBD.

# DAFTAR ISI

[ABSTRAK iv](#_Toc458754667)

[KATA PENGANTAR v](#_Toc458754668)

[DAFTAR ISI vi](#_Toc458754669)

[DAFTAR GAMBAR ix](#_Toc458754670)

[DAFTAR TABEL x](#_Toc458754671)

[BAB I PENDAHULUAN 1](#_Toc458754672)

[I.1 Latar Belakang 1](#_Toc458754673)

[I.2 Rumusan Masalah 2](#_Toc458754674)

[I.3 Tujuan 3](#_Toc458754675)

[I.4 Batasan Masalah 3](#_Toc458754676)

[I.5 Metodologi 3](#_Toc458754677)

[I.6 Sistematika Pembahasan 4](#_Toc458754678)

[BAB II STUDI LITERATUR 5](#_Toc458754679)

[II.1 Bobot Kata 5](#_Toc458754680)

[II.2 *Term* *Frequency* 5](#_Toc458754681)

[II.3 *Inverse* *Document Frequency* 5](#_Toc458754682)

[II.4 Normalisasi 5](#_Toc458754683)

[II.5 *Scoring* *Document* 6](#_Toc458754684)

[II.6 *Inverted* *File Index* 7](#_Toc458754685)

[II.7 Evaluasi Sistem *Information Retrieval* 8](#_Toc458754686)

[II.8 Proses Sistem *Information Retrieval* *Convensional* 8](#_Toc458754687)

[II.9 *Pseudo* *Relevance Feedback* 10](#_Toc458754688)

[II.10 *Verbose* *Query* 11](#_Toc458754689)

[II.11 Prinsip *Term Selection* pada *Phrase Rank* 11](#_Toc458754690)

[II.12 *Markov* *chain framework* untuk reformulasi *query* 12](#_Toc458754691)

[II.13 *Phrase* *Rank* 12](#_Toc458754692)

[II.13.1 Algoritma *Phrase Rank* 13](#_Toc458754693)

[II.13.2 Konstruksi Graf 14](#_Toc458754694)

[II.14 *Sequential Dependence* *Model* 17](#_Toc458754695)

[II.15 *Mean* *Average* *Precision* 18](#_Toc458754696)

[BAB III ANALISIS RENCANA SOLUSI PERMASALAHAN 19](#_Toc458754697)

[III.1 Analisis Permasalahan dan Solusi yang Dibutuhkan 19](#_Toc458754698)

[III.2 Rancangan Solusi 20](#_Toc458754699)

[III.2.1 Membangun Sistem *Information* *Retrieval Convensional* 20](#_Toc458754700)

[III.2.2 Implementasi Algoritma *Phrase Rank* pada Sistem *Information Retrieval* yang Telah Dibuat 20](#_Toc458754701)

[III.2.3 Alur Sistem *Information Retrieval* yang Mengimplentasi Algoritma *Phrase Rank* 21](#_Toc458754702)

[III.2.4 Ilustrasi Algoritma *Phrase Rank* 23](#_Toc458754703)

[III.3 Rencana Pengujian *Phrase Rank* pada Koleksi Dokumen 37](#_Toc458754704)

[BAB IV PENGUJIAN 39](#_Toc458754705)

[IV.1 Batasan Pengujian 39](#_Toc458754706)

[IV.2 Koleksi Uji yang Digunakan 40](#_Toc458754707)

[IV.3 Pengujian Sistem *Information Retrieval* yang Mengimplementasi *Phrase Rank* Standar 40](#_Toc458754708)

[IV.3.1 Koleksi ADI 41](#_Toc458754709)

[IV.3.2 Koleksi CISI 41](#_Toc458754710)

[IV.3.3 Koleksi CRAN 42](#_Toc458754711)

[IV.4 Pengujian Sistem *Information Retrieval* yang Mengimplementasi Modifikasi *Phrase Rank* 43](#_Toc458754712)

[IV.4.1 Koleksi ADI 43](#_Toc458754713)

[IV.4.2 Koleksi CISI 44](#_Toc458754714)

[IV.4.3 Koleksi CRAN 45](#_Toc458754715)

[IV.5 Analisis Hasil Pengujian 46](#_Toc458754716)

[BAB V KESIMPULAN DAN SARAN 48](#_Toc458754717)

[V.1 Kesimpulan 48](#_Toc458754718)

[V.2 Saran 48](#_Toc458754719)

[DAFTAR PUSTAKA 49](#_Toc458754720)

# DAFTAR GAMBAR

[Gambar II.1 Alur Sistem *Information Retrieval* 9](#_Toc458754721)

[Gambar II.3 *Markov* *random field* yang digunakan 17](#_Toc458754722)

[Gambar II.4 *Sequential Dependence Model* 18](#_Toc458754723)

[Gambar III.1 Alur sistem *information retrieval* yang Mengimplementasi *phrase rank* 22](#_Toc458754724)

# DAFTAR TABEL

[Tabel II.1 Contoh struktur *inverted file index* 8](#_Toc458754725)

[Tabel II.2 Algoritma *phrase rank* 13](#_Toc458754726)

[Tabel II.3 Contoh hasil reformulasi query dengan menggunakan *phrase rank* 14](#_Toc458754727)

[Tabel III.1 Contoh hasil *text operations* 23](#_Toc458754728)

[Tabel III.2 Contoh hasil *query reformulations* 23](#_Toc458754729)

[Tabel III.3 Contoh hasil *indexing* 24](#_Toc458754730)

[Tabel III.4 Contoh hasil *ranking* 28](#_Toc458754731)

[Tabel III.5 Contoh hasil evaluasi koleksi ADI 30](#_Toc458754732)

[Tabel III.6 Contoh hasil *pseudo relevance feedback* 30](#_Toc458754733)

[Tabel III.7 Contoh hasil algoritma *phrase rank* 31](#_Toc458754734)

[Tabel IV.1 Format koleksi dokumen 39](#_Toc458754735)

[Tabel IV.2 Format koleksi *query* 39](#_Toc458754736)

[Tabel IV.3 Format *relevance judgement* 40](#_Toc458754737)

[Tabel IV.4 Koleksi uji yang digunakan 40](#_Toc458754738)

[Tabel IV.5 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada ADI untuk *k* = 2 41](#_Toc458754739)

[Tabel IV.6 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada ADI untuk *k* = 5 41](#_Toc458754740)

[Tabel IV.7 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada ADI untuk *k* = 10 41](#_Toc458754741)

[Tabel IV.8 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada ADI untuk *k* = 20 41](#_Toc458754742)

[Tabel IV.9 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CISI untuk *k* = 2 41](#_Toc458754743)

[Tabel IV.10 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CISI untuk *k* = 5 42](#_Toc458754744)

[Tabel IV.11 Perbandingan NIAP antara sistem *information retrieval convensional* dan *phrase rank* standar pada CISI untuk k = 10. 42](#_Toc458754745)

[Tabel IV.12 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CISI untuk *k* = 20. 42](#_Toc458754746)

[Tabel IV.13 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CRAN untuk *k* = 2. 42](#_Toc458754747)

[Tabel IV.14 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CRAN untuk *k* = 5. 42](#_Toc458754748)

[Tabel IV.15 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CRAN untuk *k* = 10. 43](#_Toc458754749)

[Tabel IV.16 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CRAN untuk *k* = 20. 43](#_Toc458754750)

[Tabel IV.17 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada ADI untuk *k* = 2 43](#_Toc458754751)

[Tabel IV.18 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada ADI untuk *k* = 5 43](#_Toc458754752)

[Tabel IV.19 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada ADI untuk *k* = 10 44](#_Toc458754753)

[Tabel IV.20 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada ADI untuk *k* = 20 44](#_Toc458754754)

[Tabel IV.21 perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CISI untuk *k* = 2 44](#_Toc458754755)

[Tabel IV.22 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CISI untuk *k* = 5 44](#_Toc458754756)

[Tabel IV.23 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CISI untuk *k* = 10 45](#_Toc458754757)

[Tabel IV.24 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CISI untuk *k* = 20 45](#_Toc458754758)

[Tabel IV.25 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CRAN untuk *k* = 2 45](#_Toc458754759)

[Tabel IV.26 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CRAN untuk *k* = 5 45](#_Toc458754760)

[Tabel IV.27 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CRAN untuk *k* = 10 46](#_Toc458754761)

[Tabel IV.28 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CRAN untuk *k* = 20 46](#_Toc458754762)

# PENDAHULUAN

## Latar Belakang

Sistem temu balik informasi atau *information retrieval* adalah penemuan dokumen dari suatu teks (*query*) yang dimasukkan oleh pengguna yang memenuhi kebutuhan informasi dalam koleksi dokumen yang besar (Manning, Raghavan, & Schutze, 2009). Seringkali, *query* yang dimasukkan pada *information retrieval* ditemukan kurang lengkap, atau terlalu panjang sehingga dokumen yang ditemukan kurang relevan dengan apa yang dimaksud oleh pengguna. Oleh karena itu terdapat beberapa metode yang dapat digunakan agar *query* dapat diperbaiki sehingga dapat menghasilkan dokumen yang lebih relevan.

Reformulasi *query* merupakan salah satu metode untuk meningkatkan performansi *information retrieval* yang mencakup teknik-teknik seperti *query expansion* dan *term selection*. Pada penanganan *query* yang pendek, efektifitas *information retrieval* dapat ditingkatkan dengan melakukan *query expansion* yang menggunakan metode *pseudo relevance feedback*. Di sisi lain, *query* yang panjang atau *verbose* memiliki banyak topik/konteks sehingga ekspansi yang terjadi cenderung dikendalikan oleh *query* tersebut atau disebut dengan *query drift*.

*Term selection* dan *term weighting* merupakan salah satu cara untuk melihat kepentingan atau esensi dari suatu *query*. Terdapat berbagai macam metode yang dapat digunakan untuk mengoptimasi *term selection* dan *term weighting*, beberapa di antaranya yaitu *global statistic* dan *local syntactic*. *Global statistic* merupakan indikator kuat dalam menentukan kepentingan suatu *term*, namuntidak dapat merefleksikan konteks dari suatu *local query*. Di sisi lain, *local syntactic* dapat mengidentfikasi dengan tepat *term* manayang memiliki relasi, namun tidak dapat mengidentifikasi relasi *term* yang memiliki informasi penting. Kedua metode tersebut tidak dapat mendeteksi *informative terms,* padahal *informative terms* merupakan aspek yang penting dari sebuah *query*.

*PhRank* (*Phrase Rank*) merupakan algoritma yang menggunakan *pseudo relevance feedback* yang pemakaiannya cocok untuk *term selection* pada suatu *query*, terutama pada *query* yang panjang. Selain itu, *PhRank* memiliki tiga karakteristik yang penting dalam mengidentifikasi *informative terms*, seperti *query context*, *compactness*, dan intergasi pengetahuan *syntactic* dan *semantic*. Hasil reformulasi *query* dari *PhRank* berdasarkan pengujian pada TREC (*Text Retrieval Conference*) menunjukkan bahwa jumlah *term* yang dipilih dari *query* sedikit, jumlah kata pada *term* tersebut juga sedikit, dan konteks dari *term* tersebut sesuai dengan *query* (*informative*).

## Rumusan Masalah

Seperti yang telah dijelaskan pada bagian latar belakang, ekspansi pada query yang panjang cenderung menjadi *query drift* sehingga dibutuhkan penanganan khusus untuk dapat meningkatkan performansi sistem *information retrieval*. Reformulasi *query* dengan menggunakan *PhRank* merupakan salah satu solusi yang tepat pada pencarian dokumen dengan *query* yang panjang, sekaligus dapat meningkatkan performansi sistem *information retrieval*. Pada penelitian ditunjukkan bahwa *PhRank* menggunakan kandidat *term* dengan kombinasi satu sampai tiga kata yang telah dipilih berdasarkan skor untuk kemudian dilakukan *ranking* pada kandidat *term* tersebut. Berikut adalah rumusan masalah yang terdapat pada tugas akhir ini.

1. Bagaimana implementasi algoritma *PhRrank* yang berdasarkan peneliatian pada sistem *information retrieval*?
2. Bagiamana implementasi algoritma *PhRank* yang menggunakan kandidat *term* dengan kombinasi satu sampai enam kata pada sistem *information retrieval*?
3. Bagaimana hasil performansi sistem *information retrieval* dari implementasi algoritma *PhRank* yang berdasarkan penelitian dan yang sudah dimodifikasi?

## Tujuan

Berdasarkan rumusan masalah di atas perlu dilakukan implementasi *PhRank* pada sistem *information retrieval* sehingga dapat menunjukkan bahwa algoritma tersebut merupakan solusi yang tepat untuk reformulasi *query* yang panjang. Pada tugas akhir ini akan dibahas implementasi algoritma *PhRank* untuk *term selection* pada *query*. Tugas akhir ini memiliki beberapa tujuan, di antaranya:

1. Melakukan implementasi algoritma *PhRank* yang berdasarkan penelitian pada sistem *information retrieval*.
2. Melakukan implementasi algoritma *PhRank* yang dimodifikasi, yaitu menggunakan kandidat *term* dengan kombinasi satu sampai enam kata pada sistem *information retrieval*.
3. Melakukan pengujian dari sistem *information retrieval* dengan algoritma *PhRank* yang telah diterapkan.
4. Melakukan analisis berdasarkan performansi sistem *information retrieval* dari hasil pengujian yang telah dilakukan.

## Batasan Masalah

1. Tidak menangani *abbreviation* pada reformulasi *query*.
2. Tidak menggunakan koleksi TREC sesuai pada penelitian, akan tetapi menggunakan koleksi ADI, CISI, dan CRAN.

## Metodologi

Berikut adalah metodologi yang digunakan pada tugas akhir ini.

1. Merancang sistem *information retrieval*

Pada tahapan ini ditentukan modul-modul yang akan dibangun dan fitur yang akan diuji dalam pengujian sistem information *retrieval*.

1. Pembangunan sistem *information retrieval convensional*

Agar dapat melakukan implementasi algoritma *PhRank* dan melakukan pengujian perlu dibangun terlebih dahulu suatu sistem *information retrieval convensional*.

1. Mengimplementasi algoritma *PhRank* berdasarkan penelitian pada *sistem information retrieval*
2. Mengimplementasi algoritma *PhRank* yang sudah dimodifikasi pada sistem *information retrieval*.
3. Pengujian pada koleksi dokumen

Akan ditunjukkan hasil implementasi dari algoritma *PhRank* berdasarkan penelitian maupun yang sudah dimodifikasi dengan melakukan pengujian pada beberapa koleksi dokumen yang ditentukan.

1. Analisis terhadap pengujian yang telah dilakukan berdasarkan performansi dari sistem *information retrieval*.

## Sistematika Pembahasan

Bab I berisi tentang latar belakang mengapa algoritma *phrase rank* dibutuhkan. Selain itu terdapat juga rumusan masalah yang menjadi landasan pengerjaan tugas akhir, tujuan pengerjaan tugas akhir, beserta batasan masalah yang terdapat pada tugas akhir. Kemudian dipaparkan juga metodologi yang mendukung pengerjaan tugas akhir.

Bab II berisi studi literatur tugas akhir yang mencakup teori-teori yang mendukung pengerjaan tugas akhir. Teori yang didapatkan berasal dari penelitian yang berkaitan dengan tugas akhir. Teori-teori yang terdapat pada bab ini digunakan untuk seluruh penulisan tugas akhir yang berkaitan dengan teori tersebut.

Bab III berisi analisis permasalahan yang lebih detil dan pemaparan solusi yang dibutuhkan. Pada bab ini dibahas juga rancangan dari solusi yang akan diimplementasikan. Selain itu rencana pengujian juga dipaparkan pada bab ini.

Bab IV berisi hasil pengujian yang telah dilakukan berdasarkan rencana pengujian. Pada bab ini dicakup juga batasan yang terdapat pada pengujian. Selain itu dipaparkan juga analisis hasil pengujian.

Bab V berisi kesimpulan dan saran dari pengerjaan tugas akhir.

# STUDI LITERATUR

## Bobot Kata

Setiap *term* yang terdapat dalam dokumen memiliki bobotnya masing-masing. Bobot *term* digunakan untuk menentukan seberapa pentingnya suatu *term* dalam dokumen tersebut. Dalam menentukan bobot *term* terdapat beberapa cara yang dapat digunakan, yaitu *term frequency*, *inverse document frequency*, dan *normalization*.

## *Term* *Frequency*

Semakin besar kemunculan *term* pada suatu dokumen, *term* tersebut akan semakin penting. Kemunculan *term* dalam satu dokumen disebut dengan *term frequency* (*tf*) (Mandala, R & Hendra, S, 2002). Salah satu cara yang dapat digunakan adalah dengan menggunakan *raw tf* di mana nilai *tf* sesuai dengan jumlah kemunculan *term* pada dokumen.

## *Inverse* *Document Frequency*

Pembobotan dengan menggunakan *inverse document frequency* (*idf*) memperhitungkan kebalikan dari frekuensi dokumen yang mengandung suatu *term*. Berikut adalah persamaan yang digunakan dalam menghitung *inverse document frequency*.

……………………………..II.1

*N* adalah jumlah total dokumen dan *dfi* adalah jumlah dokumen yang mengandung *term* *ti*(Mandala, R & Hendra, S, 2002).

## Normalisasi

Setiap dokumen yang terdapat dalam koleksi dokumen memiliki panjang yang berbeda-beda. Berikut adalah akibat dari variasi panjang dokumen dalam koleksi dokumen.

* 1. Besarnya frekuensi *term*

Pada dokumen yang panjang, *term* yang sama cenderung muncul berulang kali sehingga menyebabkan *term frequency* cenderung besar (Mandala, R & Hendra, S, 2002). Semakin besar *term frequency* dapat membuat rata-rata bobot *term* menjadi tinggi dan meningkatkan relevansi dokumen terhadap *query*.

* 1. Banyaknya *term*

Dalam dokumen yang panjang sering ditemukan sejumlah *term* yang berbeda di mana dapat meningkatkan relevansi dokumen terhadap query pula.

Dengan melakukan normalisasi panjang dokumen, dokumen yang pendek akan ikut diperhitungkan dalam *document similarity*, yaitu hasil *ranking* dokumen berdasarkan kesamaan pada *query* yang menggunakan model ruang vektor. Berikut adalah rumus faktor normalisasi untuk suatu dokumen (Manning, Raghavan, & Schutze, 2009).

…………………..II.2

*M* adalah jumlah kata pada dokumen, *i* adalah indeks kata pada dokumen, adalah bobot dari kata pada dokumen *d*.

## *Scoring* *Document*

Untuk mendapatkan bobot setiap kata dalam koleksi dokumen dilakukan pembobotan kata dengan *tf*.*idf* (Manning, Raghavan, & Schutze, 2009)*.* Dengan *t* adalah kata yang akan diberikan bobot dan *d* adalah dokumen yang mengandung kata *t* tersebut, berikut adalah rumus *tf*.*idf* yang digunakan.

………………………….II.3

Untuk mendapatkan dokumen yang penting berdasarkan suatu *query q,* dan *t* adalah *term* dari *query* diperlukan *scoring* pada koleksi dokumen terhadap *query* tersebut. Berikut adalah rumus dari *scoring document* yang digunakan, yaitu perkalian bobot *tf.idf normalization* antara setiap *term* pada *query* dan *term* pada dokumen yang sesuai dengan *term* pada *query*.

…………………….II.4

Untuk melakukan normalisasi setiap bobot kata pada dokumen maupun *query* digunakan faktor normalisasi. Berikut adalah rumus untuk *tf*.*idf* *normalization*.

…………………II.5

Berikut adalah rumus dari *scoring document* dengan bobot yang sudah dinormalisasi.

…………….II.6

## *Inverted* *File Index*

*Indexing* adalah salah satu cara untuk menghindari pencarian *query* pada seluruh teks dokumen. *Indexing* dilakukan pada setiap *term* dokumen maupun *query* sehingga setiap *term* pada dokumen memiliki nomor dokumen, begitu pun pada *query*. *Inverted file index* adalah *file* yang berisi urutan *term* sesuai abjad yang terkandung dalam koleksi dokumen, beserta bobot *term* tersebut, dan juga nomor dokumen pada *term* tersebut (Manning, Raghavan, & Schutze, 2009). Dengan *inverted file index*, proses penemuan dokumen terhadap *query* akan lebih cepat karena struktur *file*-nya mendukung pencarian dengan cepat. Kebanyakan *search engine* yang ada menggunakan *inverted file index*. Setelah *inverted file index* dibuat dapat dilakukan *retrieval,* yaitu pemilihan dokumen yang mengandung kata-kata pada *query* yang dicari. Dalam proses *retrieval* pada s*istem information retrieval* di sini dilakukan juga pemilihan dokumen relevan dan yang tidak relevan berdasarkan *relevance judgement*, yaitu informasi mengenai dokumen relevan mana saja untuk setiap *query* yang telah ditentukan. Pemilihan dokumen relevan dan tidak relevan ini akan berguna untuk evaluasi. Tabel II.1 adalah contoh struktur dari *inverted file index* yang digunakan.

Tabel II.1 Contoh struktur *inverted file index*

|  |  |  |
| --- | --- | --- |
| *Term* | Nomor Dokumen | Bobot *Term* |
| abandon | 612 | 0,199761459449107 |
| abort | 932 | 0,136220315010701 |
| cost | 12 | 0,0504533810905941 |

## Evaluasi Sistem *Information Retrieval*

Tujuan dari *information retrieval* yang ideal adalah menemukan seluruh dokumen yang relevan dan menemukan dokumen relevan saja, artinya tidak terdapat dokumen yang tidak relevan pada dokumen hasil pencarian (Mandala, R & Hendra, S, 2002). Evaluasi suatu sistem *information retrieval* dapat dilakukan dengan menilai performansi dari sistem *information retrieval* tersebut. Performansi *information retrieval* bergantung pada dua hal ini, yaitu dengan hasil penghitungan *recall* dan *precision*. *Recall* adalah bagian dari dokumen relevan dalam dokumen yang ditemukan. *Precision* adalah bagian dari dokumen relevan yang ditemukan.

*Precision* dan *recall* adalah ukuran yang digunakan berdasarkan kumpulan dokumennya sehingga yang dievaluasi adalah kualitas dari kumpulan dokumen yang di-*retrieve* yang tidak terurut berdasarkan skor. Untuk mengevaluasi berdasarkan skor, nilai precision diinterpolasi terhadap sekumpulan recall. IAP (*Interpolated Average Precision*) adalah nilai rata-rata interpolasi dari precision terhadap recall. NIAP (*Non*-*Interpolated Average Precision*) adalah rata-rata precision untuk setiap kemunculan dokumen relevan. NIAP adalah nilai yang merefleksikan performansi terhadap semua dokumen yang relevan. Oleh karena itu, di antara ukuran yang lain, NIAP paling mencerminkan performansi dari suatu sistem *information retrieval*.

## Proses Sistem *Information Retrieval* *Convensional*

Kata-kata yang sangat umum biasanya dihilangkan dari koleksi dokumen maupun *query* karena kurang membantu dalam pemilihan dokumen yang sesuai dengan *query*. Kata-kata yang sangat umum tersebut disebut dengan *stop words*. Dalam sistem *information retrieval* di sini digunakan *default english stop word list*, yaitu *stop words* berbahasa Inggris dengan jumlah 317 kata.

Dalam koleksi dokumen maupun *query* sering ditemukan kata-kata yang berbeda, namun dengan bentuk dasar yang sama, seperti *definition*, *defitions*, *definiting*. Sebaiknya, ketika terdapat kata *definitions* pada *query,* dokumen-dokumen yang mengandung kata dasar *definition* dan *definiting* ikut menjadi bagian dokumen yang dipilih oleh *query* tersebut. Oleh karena itu dibutuhkan *stemming* pada koleksi dokumen dan *query* untuk membuat suatu kata menjadi bentuk kata dasar. Pada sistem *information retrieval* digunakan algoritma *stemming* bahasa Inggris yang umum, yaitu *Porter Stemmer*.

Gambar II.1 berikut ini adalah alur sistem *information retrieval convensional* yang digunakan pada tugas akhir ini.



Gambar II.1 Alur Sistem *Information Retrieval*

Berikut ini adalah penjelasan dari alur sistem *information retrieval*.

1. *Text* *Operations* adalah proses yang mencakup penghilangan *stop words* dan *stemming* pada koleksi *query* maupun dokumen.
2. *Query formulation* adalah proses yang mencakup *indexing* pada *query* menjadi *terms index,* sekaligus pembobotan dengan *tf.idf* normalization pada setiap kata dari hasil *text operations* koleksi *query*.
3. *Indexing* adalah proses yang mencakup *indexing* pada dokumen yang disimpan dalam *collection index*, sekaligus pembobotan dengan *tf.idf normalization* pada setiap kata dari hasil *text operations* koleksi dokumen.
4. *Ranking* adalah proses yang mencakup pencarian dokumen yang relevan terhadap *query* (*retrieval*)dan mengurutkan dokumen berdasarkan skor yang telah didapatkan dari *scoring document* yang menggunakan persamaan II.6. Daftar dokumen yang relevan terhadap suatu *query* didapatkan dari *relevance judgement*.

## *Pseudo* *Relevance Feedback*

Terdapat beberapa metode untuk melakukan penyesuaian suatu query terhadap dokumen. Metode tersebut adalah *relevance feedback*, *pseudo relevance feedback*, dan *indirect relevance feedback*. *Relevance feedback* merupakan metode yang melibatkan *feedback* pengguna dalam proses *retrieval* (Manning, Raghavan, & Schutze, 2009). *User* dapat memberikan *feedback* berupa memilih dokumen yang relevan terhadap *query*. Dengan *feedback* ini diharapkan *information retrieval* dapat menampilkan urutan dokumen sesuai dengan yang diinginkan.

*Relevance feedback* memiliki kekurangan, yaitu prosesnya tidak stabil karena jumlah dokumen relevan yang sering berubah. Oleh karena itu, untuk menghindari proses yang tidak stabil, terdapat metode *pseudo relevance feedback* yang menyediakan otomatisasi *relevance feedback* di mana *user* dapat meningkatkan performansi *retrieval* tanpa breinteraksi dengan *information retrieval*. Pada *pseudo relevance feedback* diasumsikan beberapa dokumen merupakan *top ranked* dokumen yang relevan. Berikut merupakan langkah-langkah yang terjadi dalam *pseudo*-*relevance feedback*.

1. *User* memasukkan *query*.
2. Sistem melakukan *pseudo*-*relevance feedback* pada *retrieval* awal dengan melakukan *expansion* pada hasil *retrieval* awal.
3. Sistem melakukan komputasi untuk dapat memberikan hasil *retrieval* yang lebih baik berdasarkan *top ranked* dokumen yang relevan dan *expansion*-nya.
4. Sistem menampilkan hasil *retrieval* akhir kepada *user*.

## *Verbose* *Query*

*Verbose query* adalah *query* yang mengekspresikan suatu kebutuhan informasi yang spesifiik atau kompleks. Algoritma *ranking* dokumen dari kebanyakan *search engine* modern dibuat untuk *keyword query*, yaitu *query* pendek dengan satu sampai tiga kata. Dibandingkan dengan *keyword* *query*, *verbose query* mengandung kata-kata yang memang sesuai dengan konteks *query*, namun dapat menemukan dokumen yang tidak relevan pada *retrieval*. Oleh karena itu, performansi algoritma *ranking* cenderung menurun ketika verbose *query* digunakan. Selain itu, *query* dengan informasi yang lebih sedikit sering mendapatkan performansi yang lebih baik dibandingkan dengan *verbose query* untuk kebutuhan informasi yang sama (Maxwell, 2014). Jumlah dokumen relevan yang didapatkan dari *verbose query* dapat ditingkatkan dengan representasi *query* yang lebih baik dan pemodelan *information retrieval* yang lebih baik (Maxwell, 2014). *Phrase rank* menggunakan pendekatan *dependency* dalam menentukan suatu term diskriminatif atau tidak dan menggunakan *word adjacency* untuk memilih *term*. Setelah itu, *phrase rank* melakukan *term selection* untuk mereformulasi *query*.

## Prinsip *Term Selection* pada *Phrase Rank*

Terdapat beberapa prinsip dalam *PhRank*, yaitu penggunaan istilah *word* dan *term*. *Word* adalah informasi yang relatif terhadap *query* dan memiliki hubungan dengan *word* yang lain. *Term* adalah pembeda dalam koleksi dokumen dan mengandung *word* (Maxwell, 2014).

Terdapat sebuah hipotesis, yaitu jika suatu *term* dapat membedakan antara dokumen yang relevan dan tidak relevan, *term* lain yang dekat dan terhubung dengan *term* tersebut pun dapat seperti itu (Maxwell & Croft, 2013). Algoritma *PhRank* mengimplementasikan hipotesis di atas dengan menggunakan *Markov* *chain framework*. Bentuk dari graf dapat membantu memperkirakan semantik dari *word* yang berhubungan. *Random walk* graf menentukan keterhubungan dari suatu *word* dengan *word* lain yang terhubung.

## *Markov* *chain framework* untuk reformulasi *query*

*Markov* *chain* adalah suatu proses yang mana hasil pengujian yang telah dilakukan dapat memengaruhi hasil pada pengujian berikutnya (Grinstead & Snell, 1997). *Markov chain framework* di sini menggunakan distribusi stasioner dari *random walk* pada graf yang terhubung untuk mendapatkan *node*/*vertex* dari graf. Distribusi stasioner pada *Markov* *chain* dengan matriks transisi *P* adalah suatu vektor *ψ* di mana *ψP* = *ψ* (Goldman, 2008). *Node* pada graf merepresentasikan kata dan *edge*/busur antar *node* merepresentasikan hubungan antar kata. *Random walk* mendeskripsikan langkah bertahap secara *random* / semi-*random* antara *node* dalam graf. Misalkan *lij* adalah *edge weight* antara *vertex* *vi* dan *vj*. *Path* ditentukan oleh probabilitas matrix *H* = *hij* dengan ukuran n, di mana n adalah jumlah *node* unik dalam graf. Probabilitas *hij* = *lij* jika *vi* dan *vj* terhubung dan *hij* = 0 jika sebaliknya.

Misalkan, *πjt* merupakan *affinity score* dari *node vj* pada waktu *t* dan *πjt+1* adalah jumlah *affinity score* untuk tiap *node* *vi* yang terhubung dengan *vj*. Berikut merupakan rumus dari jumlah *affinity score*.

….……………………………II.8

## *Phrase* *Rank*

*PhRank* bisa mendapatkan konteks *query* dengan suatu graf terhubung yang dibangun dari dokumen *pseudo*-*relevant* yang mana *stopword*-nya telah dihilangkan dan telah dilakukan *stemming* pada dokumennya. *Node* dalam graf merepresentasikan *word* *stemmed* yang unik. *Edge* / busur menghubungkan *word* terhubung yang terdapat dalam kumpulan *pseudo-relevant* dokumen yang telah diproses. *Edge weight* dihitung dengan menggunakan kombinasi *weight* linier dari *word co*-*occurence*, relevansi dokumen yang memiliki *word co*-*occurence*, dan faktor bigram dalam kumpulan *pseudo*-*relevant* dokumen. *Edge weight* yang merepresentasikan kecenderungan dua *stemmed word* untuk muncul dalam *proximity* yang dekat dalam dokumen merefleksikan suatu topik *query*.

*Score* dari *word* dalam graf dihitung dengan menggunakan suatu algoritma *random walk*. *Score* dari *term* dihitung dengan menggunakan skor rata-rata untuk *stemmed word* dalam suatu *term*. *Term* dengan *score* yang tertinggi digunakan untuk reformulasi *query*.

### Algoritma *Phrase Rank*

Berikut adalah *pseudocode* yang digunakan untuk algoritma *PhRank* (Maxwell, 2014).

Tabel II.2 Algoritma *phrase rank*

|  |  |
| --- | --- |
| k = 5 # jumlah top document  resourceList = [C, wikipedia]  # C adalah koleksi dokumen yang diambil dari wikipedia  for q in queryList:  N = set()  for rsc in resourceList: N.add(retrieve\_top\_k(q,rsc))  N = retrieve\_top\_k(q,N)  N.add(q)  G = arrayStruct()  for(doc, docRel) in N:  doc.stopStem() G.grow(buildGraph(doc,docRel))  G.IdfWeightEdge()#factor r  G.normalize()  G.Iterate()  G.weightVertex()#factor s  T = q.terms  for term in q:  term.wt=G.score(term)  #factorz  term.wt+=term.globalWt(C)  T.sortByWeight() | def buildGraph(doc, docRel):  docG = Index(doc)  docG.linearWt(uw2, uw10)  docG.weight(docRel)  return docG  def score(term):  S = 0  for w in term.wordSplit():  S = S +  self.affinityScore(term)  return S /= term.length()  def globalWt(C):  I = self.length()  wt = C.tfidf(self) \* I^I  return wt |

Pada *resource list* didapatkan kumpulan kata dari wikipedia yang disimpan dalam koleksi dokumen *C*. *N* adalah kumpulan *top*-*k* dokumen dari koleksi dokumen *rsc* beserta *query* *q* yang berasal dari *query list*. *G* adalah graf dengan *node* berupa stem *word* yang unik dan *edge* berupa *weight* yang didapatkan dari *word* yang bertetanggaan. Penjelasan lebih detil dari algoritma ini akan dipaparkan pada bab III. Berikut adalah contoh dari hasil reformulasi *query* dengan menggunakan algoritma *phrase rank*.

Tabel II.3 Contoh hasil reformulasi query dengan menggunakan *phrase rank*

|  |
| --- |
| *Query* : *Locations of volcanic activity which occurred within the present day boundaries of the U.S. and its territories.* |
| *Query* yang telah direformulasi : *volcanic* |

### Konstruksi Graf

Misalkan terdapat suatu *query* *Q* yang tersusun dari kumpulan *word*, *Q* = {*w1*,...*wn*} dengan *i* adalah indeks *word* pada *query*,dan *wi* adalah *word* ke-*i* pada *query* dan *C* adalah koleksi dokumen. *Top* *k* dokumen yang di-*retrieve* dari *C* dengan menggunakan *Q* diasumsikan untuk mendeskripsikan suatu topik yang mirip terhadap *Q* (Maxwell, 2014). Peningkatan performansi sistem *information retrieval* terbaik didapatkan ketika *k* yang digunakan berjumlah 2 sampai 5 (Maxwell & Croft, 2013). Jika *k* semakin besar, performansi sistem *information retrieval* akan menurun karena semakin banyak pula dokumen tidak relevan yang di-*retrieve*. Akan tetapi, variasi *k* tidak membuat perubahan performansi yang signifikan karena adanya pembobotan dari dua kata yang muncul bersamaan pada dokumen relevan (Maxwell & Croft, 2013). Top *k* dokumen dalam *C*, bersama dengan *Q* yang diartikan sebagai dokumen d0 membentuk dokumen *neighboring*dalam kumpulan *neighborhood*, yaitu *N* = {*d0*,...*dk*} dengan *i* adalah indeks dokumen, dan *di* (*i* bernilai 1 sampai k) adalah dokumen ke-*i* pada koleksi dokumen C.

Terdapat sekurang-kurangnya 18 *stop word* yang digunakan pada dokumen-dokumen di dalam *N* dan di-stem dengan menggunakan Krovertz *stemmer*. *Edge* dalam graf menghubungkan *stemmed words i* dan *j* pada *node vi* dan *vj* jika *i* dan *j* bertetangga dalam *N*. Dokumen dalam *N* yang hanya memiliki 1 *word* dihapus untuk memastikan semua *node* sekurang-kurangnya terhubung pada suatu *edge*.

#### *Edge* *Weight*

*Edge weight lij* berdasarkan pada kombinasi bobot linier dari jumlah kemunculan *i* dan *j* secara bersamaan dalam *window W* yang berukuran 2 dan 10 (Maxwell, 2014). Hal ini disebabkan perbedaan derajat *proximity* dapat menunjukkan keterhubungan kata dalam *information retrieval*. *Edge weight* didefinisikan sebagai berikut.

……………….II.9

*p(dk│Q)* adalah probabilitas dokumen di mana *stemmed word* *i* dan *j* muncul bersamaan dengan query *Q* yang diberikan. *λ* adalah fitur bobot yang digunakan pada *sequential dependence model*. *λ* yang digunakan di ini bernilai 0,6. serta adalah *count* dari *word* *co*-*occurence* dalam *windows* yang berukuran 2 dan 10 dalam *N*.

Graf mengandung banyak *stemmed word* sehingga *unweighted affinity score* dapat dipengaruhi oleh *co*-*occurrence* yang sering muncul, namun tidak memiliki hubungan yang informatif. Faktor *r* adalah suatu bobot *tf.idf* yang menunjukkan seberapa pentingnya hubungan antara *stemmed word i* dan *j* dalam *N*. Faktor *r* meminimasi pengaruh dari *co*-*occurrence* yang tidak informatif. Berikut adalah formula dari *r*.

…………………………....II.10

#### *Vertex* *Weight*

Berdasarkan *random walk*, *stemmed word* dalam graf diberikan *weight* untuk menangkap kelengkapan (*exhaustiveness*) *query* dan global *saliency* (*word* yang menonjol)dalam koleksi (Maxwell, 2014). Kelengkapan menunjukkan apakah suatu word w1 merepresentasikan *query* atau tidak. Jika *w1*sering muncul dalam *N*, kemudian suatu term *x* yang berisi *w1*akan mendapatkan benefit dari *additional word* *w2*...*wn*. *Word* yang umum cenderung memiliki *affinity score* yang tinggi karena *word* tersebut dapat muncul bersamaan dengan banyak *word* yang lain.

Faktor *s* menyeimbangkan kelengkapan dengan global *saliency* untuk mengidentifikasi *word* yang merupakan diskriminator buruk terhadap dokumen yang relevan dan tidak relevan.

……………………………II.11

adalah frekuensi dari suatu *word wn* dalam *N*, yang dirata-ratakan terhadap *k* + 1 dokumen (frekuensi rata-rata) dan dinormalisasi oleh frekuensi rata-rata maksimum dari semua *term* dalam *N*. adalah frekuensi *inverse document* dari *wn* dalam koleksi.

…………………………….II.12

*|D|* adalah jumlah dokumen dalam koleksi *C* dan adalah jumlah dokumen dalam *C* yang mengandung . Faktor *s* memiliki beberapa keuntungan, yaitu membuat *PhRank* untuk menjadi independen terhadap model *information retrieval* dan memastikan bahwa *term* yang dipilih diskriminatif.

#### *Term* *Ranking*

Untuk menghindari suatu bias terhadap *term* yang panjang, suatu *term x* dihitung dengan merata-ratakan *affinity score* untuk komponen *word* {*w1*,[...*wn*]} (Maxwell, 2014). *Term rank* ditentukan dengan merata-ratakan *score* yang dikalikan dengan suatu faktor *zx*. Berikut adalah faktor *zx* yang merepresentasikan seberapa diskriminatif *term* dalam suatu koleksi.

…………………………….II.13

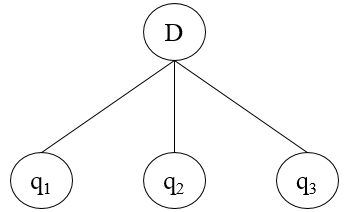
*xe* adalah ekspresi *proximity*di mana komponen *word x* muncul dalam suatu *unordered window* dengan ukuran *W* = 4 tiap *word*. adalah suatu faktor *weighting* eksponensial yang berguna untuk normalisasi frekuensi ngram selama segmentasi *query*. Misalkan, *|x|* adalah jumlah *word* dalam *x*, . Algoritma *PhRank* menjelaskan informasi dari suatu *term x* untuk suatu *query Q* yang dibandingkan dengan *term* lain. Berikut merupakan persamaan *term ranking*.

…………………......II.14

Berdasarkan hasil penelitian, performansi *phrase rank* stabil ketika dilakukan variasi pada jumlah *term* yang dipilih.

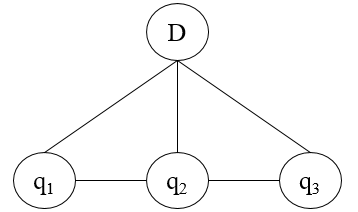
## *Sequential Dependence* *Model*

Terdapat beberapa cara untuk memodelkan distribusi *query*, salah satunya adalah dengan menggunakan *Markov* *random field* (MrF) (Metzler, 2007). *Markov* *random field* adalah *undirected graphical model* yang biasanya digunakan dalam domain *statistical machine learning* untuk memodelkan distribusi *joint* yang kompleks (Metzler, 2007). *Markov* *random field* dibangun dengan suatu graf di mana *node­-*nya menggambarkan variabel *random* dan *edge*-nya menggambarkan *independence semantic* antar variabel *random*-nya. *Independence semantic* sendiri ditentukan oleh *Markov* *property* yang menyatakan bahwa setiap variabel *random* dalam grap adalah independen dari *­node* yang bukan tetangganya. Berikut adalah cara yang digunakan pada *sequential dependence* (SD) *model* untuk memodelkan *joint distribution* dengan menggunakan MrF.



Gambar II.2 *Markov* *random field* yang digunakan

Pada model ini, *query* dipecah menjadi *term*. Suatu *query* dengan panjang *n* menghasilkan graf dengan *node term query qi,* dengan *i* adalah indeks *query,* sebanyak *n* dan satu *node* *D*, yaitu *node* yangbertetanggaan dengan semua *term query*. Dengan cara seperti ini dapat lebih dikendalikan *term dependency* mana yang dimodelkan. Terdapat beberapa variasi struktur pada model ini, salah satunya adalah struktur *sequential dependence* yang digunakan pada model SD. Berikut adalah gambar dari struktur *sequential dependence*.



Gambar II.3 *Sequential Dependence Model*

Struktur SD mengasumsikan terdapat *dependence* antar *term query* yang bertetanggaan.

## *Mean* *Average* *Precision*

Pada penelitian, performansi sistem *information retrieval* dinilai dengan *mean average precision* (MAP). *Average precision* adalah rata-rata dari nilai *precision* yang didapatkan untuk sekumpulan dokumen relevan dengan *ranking* tertentu dan kemudian dirata-ratakan terhadap panjang *query*-nya. Berikut adalah rumusan yang digunakan pada MAP.

…………….II.15

*Q* adalah *query*, *|Q|* adalah panjang *query, j* adalah indeks dari *query* yang bernilai dari satu sampai |*Q*|, *mj* adalah jumlah dokumen, dan *Rjk* adalah kumpulan dari hasil *retrieval* yang didapatkan dari dokumen yang memiliki *ranking* paling tinggi sampai dengan dokumen dengan *ranking k*.

# ANALISIS RENCANA SOLUSI PERMASALAHAN

## Analisis Permasalahan dan Solusi yang Dibutuhkan

Untuk memenuhi kebutuhan yang sesuai dengan pengguna, suatu sistem *information retrieval* membutuhkan performansi yang baik. Performansi ini dapat dilihat dari nilai NIAP berdasarkan evaluasi yang telah dilakukan pada koleksi dokumen terhadap koleksi *query*. Semakin besar nilai NIAP, performansi akan semakin baik.

Pada penelitian, ukuran performansi yang digunakan adalah MAP. Dengan menggunakan algoritma Phrase Rank, nilai MAP berdasarkan evaluasi yang telah dilakukan pada koleksi dokumen terhadap koleksi *query* lebih besar dibandingkan dengan sistem *information retrieval* yang menggunakan *model sequential dependence*. Pada tugas akhir ini, ukuran performansi yang digunakan adalah NIAP sehingga evaluasi algoritma *phrase rank* akan dinilai berdasarkan NIAP.

Setelah dilakukan pengujian, berdasarkan evaluasi standar *phrase rank* didapatkan bahwa nilai NIAP lebih besar dibandingkan dengan nilai NIAP pada evaluasi sistem *information retrieval convensional*. Evaluasi *phrase rank* tersebut menggunakan kombinasi satu sampai tiga kata dari kandidat *term* yang didapatkan. Pada tugas akhir ini akan dilakukan percobaan apakah jika kandidat *term* yang digunakan menggunakan kombinasi satu sampai enam kata dapat membuat nilai NIAP lebih besar dibandingkan dengan nilai NIAP dari evaluasi standar *phrase rank*.

Untuk mengetahui performansi dari sistem *information retrieval* yang mengimplementasi algoritma *phrase rank* menggunakan kombinasi kata satu sampai enam kata dari kandidat *term* diperlukan implementasi *phrase rank* dengan modifikasi pada kandidat *term* yang digunakan. Setelah sistem *information retrieval* *convensional* dibangun akan dilakukan implementasi standar *phrase rank* untuk mendapatkan perbandingan performansi antara sistem *information retrieval convensional* dengan sistem information retrieval yang mengimplementasi *phrase rank* standar. Setelah implementasi *PhRank standar* diselesaikan akan dibuat implementasi phrase rank yang dimodifikasi untuk mendapatkan perbandingan performansi antara sistem *information retrieval* *convensional*, sistem *information retrieval* yang mengimplementasi *phrase rank* standar, dan sistem *information retrieval* yang mengimplementasi *phrase rank* yang dimodifikasi.

## Rancangan Solusi

### Membangun Sistem *Information* *Retrieval Convensional*

Untuk melakukan pengujian tentu diperlukan sistem *information retrieval* agar bisa ditunjukkan, seperti apa implementasi yang telah dilakukan dan juga hasil yang telah didapatkan. Sebelum membangun sistem *information retrieval* dilakukan perancangan kasar, seperti membuat rancangan modul sistem, menentukan koleksi dokumen dan *query* yang akan digunakan, serta merancang *user interface* yang akan dibuat. Sistem *information retrieval convensional* dibangun berdasarkan modul-modul, seperti modul penyimpanan koleksi, penghilangan *stop words*, *stemming*, *indexing*, *weighting*, *inverted file index*, *retrieval*, *scoring*, dan evaluasi.

### Implementasi Algoritma *Phrase Rank* pada Sistem *Information Retrieval* yang Telah Dibuat

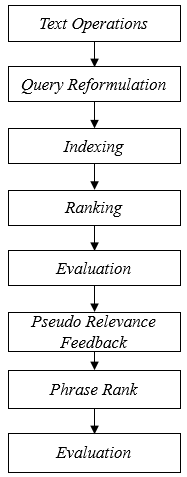
Algoritma *PhRank* yang digunakan sesuai dengan *pseudo*-*code* yang telah dipaparkan pada bagian studi literatur dengan keterangan yang akan dijelaskan kemudian. Sebelum algoritma dijalankan telah terdapat data mengenai kata mana saja yang bertetanggaan pada setiap *query* untuk semua dokumen. Selain itu telah dijalankan juga proses *pseudo relevance feedback* sehingga setiap *query* sudah memiliki *pseudo relevant document*. Berikut adalah tahapan yang dilakukan pada algoritma *PhRank*.

Untuk setiap *query* pada koleksi *query* lakukan

1. Menambahkan satu *query* pada *pseudo relevant document* yang didapatkan dari *pseudo relevance feedback*.
2. Menjadikan semua kata pada *query* maupun *pseudo relevant document* unik.
3. Membuat graf dengan langkah-langkah berikut.
4. Membuat *node* di mana setiap *node*-nya adalah kata unik.
5. Memberikan bobot dengan menggunakan faktor *s* yang terdapat pada persamaan II.11 untuk setiap *node*.
6. Membuat *edge* pada dua kata yang bertetanggaan dengan memanfaatkan matriks probabilitas yang didapatkan dari persamaan II.9 yang terdapat faktor *r*.
7. Melakukan iterasi *random walk* dari matriks probabilitas untuk mendapatkan bobot untuk *node* (jika menghilangkan faktor *s*).
8. Membuat bobot dari *node* sebagai *affinity score*.
9. Membuat kandidat *term* yang merupakan kombinasi dari satu sampai tiga kata yang didapatkan dari tiga *node* dengan *affinity score* tertinggi.
10. Melakukan *ranking* pada kandidat *term* menggunakan persamaan II.14 yang terdapat faktor *z*.
11. Memilih satu *term* dengan *affinity score* tertinggi untuk dijadikan sebagai *query* yang telah direformulasi.

### Alur Sistem *Information Retrieval* yang Mengimplentasi Algoritma *Phrase Rank*

Berikut adalah alur sistem *information retrieval* yang mengimplentasi *phrase rank* yang digunakan pada tugas akhir ini.



Gambar III.1 Alur sistem *information retrieval* yang Mengimplementasi *phrase rank*

Berikut adalah penjelasan tiap tahapan dari gambar III.1.

1. *Text operations*, *query reformulation*, *indexing*, dan *ranking* adalah tahapan yang dilakukan seperti pada alur sistem *information retrieval* yang dijelaskan pada subbab II.8.
2. Melakukan evaluasi untuk sistem *information retrieval convensional* dari semua *query* terhadap koleksi dokumen.
3. Melakukan *pseudo relevance feedback* dengan jumlah *ranked document* yang telah ditentukan.
4. Menjalankan algoritma *phrase rank* dengan dokumen yang relevan berdasarkan *pseudo relevant document* yang didapatkan dari *pseudo relevance feedback* berdasarkan dengan *feature* yang diinginkan.
5. Melakukan *retrieval* dari setiap *query* yang telah direformulasi pada koleksi *query* terhadap koleksi dokumen.
6. Melakukan evaluasi untuk sistem *information retrieval* yang mengimplementasi algoritma *phrase rank* dari semua *query* yang telah direformulasi terhadap koleksi dokumen.

### Ilustrasi Algoritma *Phrase Rank*

Berikut ini adalah ilustrasi dari implementasi algoritma *phrase rank* pada sistem *information retrieval*. Misalkan, koleksi uji yang digunakan adalah ADI, *k* bernilai 2, dan *phrase rank* yang diuji adalah *zF*.

1. Lakukan *text operations* pada koleksi *query* dan dokumen pada ADI. Tabel III.1 berikut adalah contoh hasil *text operations*.

Tabel III.1 Contoh hasil *text operations*

|  |
| --- |
| Input adalah query no. 1 pada koleksi query ADI dengan konten :  What problems and concerns are there in making up descriptive titles?  What difficulties are involved in automatically retrieving articles from approximate titles? What is the usual relevance of the content of articles to their titles?  Proses :  Penghilangan stop words dan melakukan stemming  Output :  Problem concern make descript titl difficulti involv automat retriev  articl approxim titl usual relev content articl titl |

1. Lakukan *query reformulation* dengan menggunakan *query* dari hasil *text operations*. Tabel III.2 Berikut adalah contoh hasil *query reformulations*.

Tabel III.2 Contoh hasil *query reformulations*

|  |
| --- |
| Input:  Problem concern make descript titl difficulti involv automat retriev  articl approxim titl usual relev content articl titl  Proses:  Indexing dan pembobotan dengan tf.idf normalization  Hasil:  Query 1  [TFIDFNormalized term problem = 0,16704419627752]  [TFIDFNormalized term concern = 0,241743644550063]  [TFIDFNormalized term make = 0,241743644550063]  [TFIDFNormalized term descript = 0,241743644550063]  [TFIDFNormalized term titl = 0,58384062017863]  [TFIDFNormalized term difficulti = 0,241743644550063]  [TFIDFNormalized term involv = 0,241743644550063]  [TFIDFNormalized term automat = 0,119914091787]  [TFIDFNormalized term retriev = 0,0623026092409946]  [TFIDFNormalized term articl = 0,294966871138048]  [TFIDFNormalized term approxim = 0,241743644550063]  [TFIDFNormalized term usual = 0,241743644550063]  [TFIDFNormalized term relev = 0,241743644550063]  [TFIDFNormalized term content = 0,241743644550063] |

1. Lakukan proses *indexing* sekaligus pembobotan *tf.idf normalization* pada koleksi dokumen yang menggunakan dokumen hasil *text operations*. Tabel II.3 berikut adalah contoh hasil *indexing*.

Tabel III.3 Contoh hasil *indexing*

|  |
| --- |
| Input adalah hasil text operations dari dokumen nomor 1 pada koleksi dokumen ADI :  ibm data system divis technic inform center tic provid oper development integr compat mechan process technic inform receiv organ offer advantag 1 sophist mechan dissemin retriev 2 compat librari mechan record produc standard process format ibm librari provid tradit tool 3 5 catalog card circul record overdu notic 3 revers discontinu machin process caus gap librari manual record 4 control produc statist evalu program effici 5 user orient provid 24 hour copi access immedi microfilm access document 6 relat simpl reli ibm 1401 data process process output oper year conclus drawn base actual experi  Proses :  Indexing dan pembobotan dengan tf.idf normalization  Hasil :  Dokumen 1  [TFIDFNormalized term ibm = 0,232085411042221]  [TFIDFNormalized term data = 0,11111234612462]  [TFIDFNormalized term system = 0,0509361138852585]  [TFIDFNormalized term divis = 0,0914892320265473]  [TFIDFNormalized term technic = 0,144639012503011]  [TFIDFNormalized term inform = 0,0455329679478287]  [TFIDFNormalized term center = 0,0723195062515054]  [TFIDFNormalized term tic = 0,121872528529096]  [TFIDFNormalized term provid = 0,204168913560114]  [TFIDFNormalized term oper = 0,154723607361481]  [TFIDFNormalized term development = 0,121872528529096]  [TFIDFNormalized term integr = 0,0835330769790127]  [TFIDFNormalized term compat = 0,182978464053095]  [TFIDFNormalized term mechan = 0,232085411042221]  [TFIDFNormalized term process = 0,265748902382318]  [TFIDFNormalized term receiv = 0,121872528529096]  [TFIDFNormalized term organ = 0,0611059355239981]  [TFIDFNormalized term offer = 0,0835330769790127]  [TFIDFNormalized term advantag = 0,102702802754055]  [TFIDFNormalized term 1 = 0,0835330769790127]  [TFIDFNormalized term sophist = 0,0914892320265473]  [TFIDFNormalized term dissemin = 0,0914892320265473]  [TFIDFNormalized term retriev = 0,0339800547014216]  [TFIDFNormalized term 2 = 0,0773618036807403]  [TFIDFNormalized term librari = 0,166668519186931]  [TFIDFNormalized term record = 0,250599230937038]  [TFIDFNormalized term produc = 0,116384155811397]  [TFIDFNormalized term standard = 0,0835330769790127]  [TFIDFNormalized term format = 0,0723195062515054]  [TFIDFNormalized term tradit = 0,0914892320265473]  [TFIDFNormalized term tool = 0,121872528529096]  [TFIDFNormalized term 3 = 0,167066153958025]  [TFIDFNormalized term 5 = 0,205405605508109]  [TFIDFNormalized term catalog = 0,102702802754055]  [TFIDFNormalized term card = 0,0723195062515054]  [TFIDFNormalized term circul = 0,0914892320265473]  [TFIDFNormalized term overdu = 0,121872528529096]  [TFIDFNormalized term notic = 0,102702802754055]  [TFIDFNormalized term revers = 0,102702802754055]  [TFIDFNormalized term discontinu = 0,121872528529096]  [TFIDFNormalized term machin = 0,0643633512039708]  [TFIDFNormalized term caus = 0,121872528529096]  [TFIDFNormalized term gap = 0,121872528529096]  [TFIDFNormalized term manual = 0,0723195062515054]  [TFIDFNormalized term 4 = 0,0835330769790127]  [TFIDFNormalized term control = 0,0680563045200381]  [TFIDFNormalized term statist = 0,0680563045200381]  [TFIDFNormalized term evalu = 0,0680563045200381]  [TFIDFNormalized term program = 0,0488865787449962]  [TFIDFNormalized term effici = 0,102702802754055]  [TFIDFNormalized term user = 0,0723195062515054]  [TFIDFNormalized term orient = 0,0773618036807403]  [TFIDFNormalized term 24 = 0,121872528529096]  [TFIDFNormalized term hour = 0,102702802754055]  [TFIDFNormalized term copi = 0,0723195062515054]  [TFIDFNormalized term access = 0,182978464053095]  [TFIDFNormalized term immedi = 0,102702802754055]  [TFIDFNormalized term microfilm = 0,0835330769790127]  [TFIDFNormalized term document = 0,0451936254289289]  [TFIDFNormalized term 6 = 0,0914892320265473]  [TFIDFNormalized term relat = 0,0643633512039708]  [TFIDFNormalized term simpl = 0,102702802754055]  [TFIDFNormalized term reli = 0,121872528529096]  [TFIDFNormalized term 1401 = 0,0914892320265473]  [TFIDFNormalized term output = 0,0835330769790127]  [TFIDFNormalized term year = 0,0914892320265473]  [TFIDFNormalized term conclus = 0,121872528529096]  [TFIDFNormalized term drawn = 0,121872528529096]  [TFIDFNormalized term base = 0,0680563045200381]  [TFIDFNormalized term actual = 0,121872528529096]  [TFIDFNormalized term experi = 0,0643633512039708] |

1. Lakukan proses *ranking* pada dokumen terhadap *query* pada semua koleksi *query*. Tabel III.4 berikut adalah contoh hasil *ranking* yang telah dilakukan.

Tabel III.4 Contoh hasil *ranking*

|  |
| --- |
| Input :  Hasil pembobotan tf.idf normalization query no.1 dan hasil pembobotan tf.idf normalization semua dokumen, serta relevance judgement  Proses :  Retrieval dan scoring document  Hasil :  Nomor Query = 1  Dokumen relevan yang ditemukan:  17, 46, 62  Score dokumen terurut yang ditemukan:  [Nomor Dokumen = 46 | Score = 0,368802911463392]  [Nomor Dokumen = 69 | Score = 0,344484876920876]  [Nomor Dokumen = 17 | Score = 0,258642397395006]  [Nomor Dokumen = 75 | Score = 0,114250561059334]  [Nomor Dokumen = 71 | Score = 0,102880069710133]  [Nomor Dokumen = 25 | Score = 0,0985345471181925]  [Nomor Dokumen = 47 | Score = 0,0835359519526074]  [Nomor Dokumen = 56 | Score = 0,0796073186434694]  [Nomor Dokumen = 19 | Score = 0,062620878469506]  [Nomor Dokumen = 30 | Score = 0,0573537025916276]  [Nomor Dokumen = 27 | Score = 0,0536977061295634]  [Nomor Dokumen = 4 | Score = 0,0523740298597113]  [Nomor Dokumen = 57 | Score = 0,0461846758983535]  [Nomor Dokumen = 79 | Score = 0,0440182871751083]  [Nomor Dokumen = 23 | Score = 0,0423964482189871]  [Nomor Dokumen = 14 | Score = 0,0402188352559104]  [Nomor Dokumen = 70 | Score = 0,0386465831241677]  [Nomor Dokumen = 58 | Score = 0,0340400585497007]  [Nomor Dokumen = 9 | Score = 0,0322236248567033]  [Nomor Dokumen = 51 | Score = 0,0271189127950853]  [Nomor Dokumen = 66 | Score = 0,0218110963232248]  [Nomor Dokumen = 61 | Score = 0,0208666136036942]  [Nomor Dokumen = 41 | Score = 0,0189987792857384]  [Nomor Dokumen = 18 | Score = 0,0184159324783297]  [Nomor Dokumen = 2 | Score = 0,0175001958067784]  [Nomor Dokumen = 50 | Score = 0,0171877141944971]  [Nomor Dokumen = 7 | Score = 0,0170174444540624]  [Nomor Dokumen = 64 | Score = 0,0113900187164678]  [Nomor Dokumen = 16 | Score = 0,0109591261072501]  [Nomor Dokumen = 15 | Score = 0,0104524859429365]  [Nomor Dokumen = 12 | Score = 0,00939730169988732]  [Nomor Dokumen = 39 | Score = 0,00845555845430128]  [Nomor Dokumen = 55 | Score = 0,00828536669423907]  [Nomor Dokumen = 22 | Score = 0,00551196660452794]  [Nomor Dokumen = 43 | Score = 0,00478220759192664]  [Nomor Dokumen = 21 | Score = 0,00438416785935013]  [Nomor Dokumen = 26 | Score = 0,00404827495199404]  [Nomor Dokumen = 62 | Score = 0,00400171640610096]  [Nomor Dokumen = 38 | Score = 0,00381311306780065]  [Nomor Dokumen = 11 | Score = 0,00319210770943147]  [Nomor Dokumen = 36 | Score = 0,00317686439046739]  [Nomor Dokumen = 1 | Score = 0,00211704607005029] |

1. Lakukan evaluasi pada sistem *information retrieval* berdasarkan hasil *retrieval* dan *ranking* dokumen terhadap *query* yang didapatkan. Tabel III.5 Berikut adalah contoh hasil evaluasi koleksi ADI.

Tabel III.5 Contoh hasil evaluasi koleksi ADI

|  |
| --- |
| Input :  Hasil retrieval dan ranking dokumen terhadap query  Proses :  Perhitungan recall, precision, dan NIAP  Hasil :  NIAP ADI = 0,424252103396301 |

1. Lakukan *pseudo relevance feedback* pada setiap *query* terhadap koleksi dokumen. Tabel III.6 berikut adalah contoh hasil *pseudo relevance feedback*.

Tabel III.6 Contoh hasil *pseudo relevance feedback*

|  |
| --- |
| Input :  Hasil ranking dokumen terhadap query, query no.1  Proses :  Melakukan pseudo relevance feedback dengan k = 2  Hasil :  Nomor Query = 1  Pseudo Relevant Document :  46, 69 |

1. Lakukan algoritma *phrase rank* pada setiap *query* terhadap koleksi dokumen. Tabel III.7 berikut adalah contoh hasil algoritma *phrase rank* *zF* dengan *k =* 2.

Tabel III.7 Contoh hasil algoritma *phrase rank*

|  |
| --- |
| Input :   1. Query no.1 :   What problems and concerns are there in making up descriptive titles?  What difficulties are involved in automatically retrieving articles from approximate titles? What is the usual relevance of the content of articles to their titles?   1. Hasil text operations 2. Hasil query formulation 3. Hasil indexing 4. Hasil ranking 5. Hasil pseudo relevane feedback dari query no.1   Proses :   1. Membuat sequential dependence model, yaitu menyimpan semua ketetanggaan kata pada semua dokumen terhadap semua query 2. Membentuk neighborhood *N* yang berisi query no.1 dan pseudo relevant dokumen, yaitu *N* = {query no.1, dokumen no.46, dokumen no. 69} 3. Membuat semua kata pada *N* unik 4. Membuat graf dengan 5. Node adalah semua kata unik pada *N*.   1401, accuraci, addit, administr, approxim, area, articl, author, automat, base, biolog, chemic, citat, code, communic, comput, concern, content, cost, deal, describ, descript, determin, develop, difficulti, flow, futur, ibm, includ, inform, initi, involv, journal, keyword, life, literatur, make, morphem, name, non, occur, page, paper, polici, problem, program, project, refer, relev, research, retriev, russian, scienc, scientif, select, sourc, statist, stem, studi, subject, support, titl, typic, usual   1. Memberikan bobot dengan menggunakan faktor s yang terdapat pada persamaan II.11 untuk setiap node.   Node 1401 | Weight Vertex = 1,30521787396701  Node accuraci | Weight Vertex = 1,47188454063368  Node addit | Weight Vertex = 1,25156319148578  Node administr | Weight Vertex = 1,37439079051348  Node approxim | Weight Vertex = 1,47188454063368  Node area | Weight Vertex = 1,20772412384682  Node articl | Weight Vertex = 2,61043574793402  Node author | Weight Vertex = 1,30521787396701  Node automat | Weight Vertex = 1,13855120730034  Node base | Weight Vertex = 1,13855120730034  Node biolog | Weight Vertex = 1,47188454063368  Node chemic | Weight Vertex = 1,25156319148578  Node citat | Weight Vertex = 1,25156319148578  Node code | Weight Vertex = 1,13855120730034  Node communic | Weight Vertex = 1,17065872029074  Node comput | Weight Vertex = 0,85514458761016  Node concern | Weight Vertex = 1,63855120730034  Node content | Weight Vertex = 2,61043574793402  Node cost | Weight Vertex = 1,20772412384682  Node deal | Weight Vertex = 1,47188454063368  Node describ | Weight Vertex = 0,943563707059957  Node descript | Weight Vertex = 1,47188454063368  Node determin | Weight Vertex = 1,25156319148578  Node develop | Weight Vertex = 1,11023037372662  Node difficulti | Weight Vertex = 1,63855120730034  Node flow | Weight Vertex = 1,47188454063368  Node futur | Weight Vertex = 1,20772412384682  Node ibm | Weight Vertex = 1,20772412384682  Node includ | Weight Vertex = 1,04105745718015  Node inform | Weight Vertex = 1,54061795939103  Node initi | Weight Vertex = 2,50312638297156  Node involv | Weight Vertex = 1,30521787396701  Node journal | Weight Vertex = 2,12395854172158  Node keyword | Weight Vertex = 1,20772412384682  Node life | Weight Vertex = 1,47188454063368  Node literatur | Weight Vertex = 1,25156319148578  Node make | Weight Vertex = 1,47188454063368  Node morphem | Weight Vertex = 1,47188454063368  Node name | Weight Vertex = 1,37439079051348  Node non | Weight Vertex = 1,37439079051348  Node occur | Weight Vertex = 1,47188454063368  Node page | Weight Vertex = 1,17065872029074  Node paper | Weight Vertex = 1,13855120730034  Node polici | Weight Vertex = 1,47188454063368  Node problem | Weight Vertex = 1,11023037372662  Node program | Weight Vertex = 1,97480554939784  Node project | Weight Vertex = 1,17065872029074  Node refer | Weight Vertex = 2,27710241460068  Node relev | Weight Vertex = 2,50312638297156  Node research | Weight Vertex = 2,16979304963823  Node retriev | Weight Vertex = 1,72915035134244  Node russian | Weight Vertex = 1,47188454063368  Node scienc | Weight Vertex = 2,04362250855365  Node scientif | Weight Vertex = 1,20772412384682  Node select | Weight Vertex = 1,30521787396701  Node sourc | Weight Vertex = 1,30521787396701  Node statist | Weight Vertex = 1,13855120730034  Node stem | Weight Vertex = 2,94376908126735  Node studi | Weight Vertex = 1,11023037372662  Node subject | Weight Vertex = 1,20772412384682  Node support | Weight Vertex = 1,47188454063368  Node titl | Weight Vertex = 6,52608936983504  Node typic | Weight Vertex = 1,47188454063368  Node usual | Weight Vertex = 1,63855120730034   1. Edge adalah dua kata unik pada *N* yang bertetanggaan dengan memanfaatkan matriks probabilitas yang didapatkan dari persamaan II.9 yang terdapat faktor *r*. Bobot pada edge digunakan untuk random walk.   Edge (typic,studi) | Weight Edge = -0,333333333333333  Edge (studi,flow) | Weight Edge = 0  Edge (flow,life) | Weight Edge = -0,333333333333333  Edge (life,scienc) | Weight Edge = 0  Edge (scienc,inform) | Weight Edge = -0,333333333333333  Edge (scienc,communic) | Weight Edge = 0  Edge (inform,biolog) | Weight Edge = 0  Edge (inform,administr) | Weight Edge = -0,333333333333333  Edge (biolog,scienc) | Weight Edge = -0,333333333333333  Edge (communic,project) | Weight Edge = -0,333333333333333  Edge (project,deal) | Weight Edge = 0  Edge (deal,citat) | Weight Edge = -0,333333333333333  Edge (citat,russian) | Weight Edge = 0  Edge (russian,literatur) | Weight Edge = -0,333333333333333  Edge (literatur,sourc) | Weight Edge = 0  Edge (sourc,support) | Weight Edge = -0,333333333333333  Edge (support,scientif) | Weight Edge = 0  Edge (scientif,research) | Weight Edge = -0,333333333333333  Edge (research,page) | Weight Edge = 0  Edge (page,cost) | Weight Edge = -0,333333333333333  Edge (cost,polici) | Weight Edge = 0  Edge (polici,journal) | Weight Edge = -0,333333333333333  Edge (journal,accuraci) | Weight Edge = 0  Edge (accuraci,titl) | Weight Edge = -0,333333333333333  Edge (titl,describ) | Weight Edge = 0  Edge (describ,content) | Weight Edge = -0,333333333333333  Edge (content,paper) | Weight Edge = 0  Edge (paper,program) | Weight Edge = -0,333333333333333  Edge (program,inform) | Weight Edge = 0  Edge (research,includ) | Weight Edge = 0  Edge (journal,refer) | Weight Edge = -0,333333333333333  Edge (titl,develop) | Weight Edge = 0  Edge (program,retriev) | Weight Edge = 0  Edge (ibm,1401) | Weight Edge = -0,333333333333333  Edge (1401,comput) | Weight Edge = 0  Edge (comput,program) | Weight Edge = -0,333333333333333  Edge (retriev,refer) | Weight Edge = -0,333333333333333  Edge (refer,chemic) | Weight Edge = 0  Edge (refer,code) | Weight Edge = 0  Edge (chemic,titl) | Weight Edge = -0,333333333333333  Edge (develop,futur) | Weight Edge = -0,333333333333333  Edge (futur,area) | Weight Edge = 0  Edge (area,research) | Weight Edge = -0,333333333333333  Edge (includ,select) | Weight Edge = -0,333333333333333  Edge (select,base) | Weight Edge = 0  Edge (base,addit) | Weight Edge = -0,333333333333333  Edge (addit,keyword) | Weight Edge = 0  Edge (keyword,initi) | Weight Edge = -0,333333333333333  Edge (initi,occur) | Weight Edge = 0  Edge (initi,morphem) | Weight Edge = -0,333333333333333  Edge (occur,stem) | Weight Edge = -0,333333333333333  Edge (stem,author) | Weight Edge = 0  Edge (stem,subject) | Weight Edge = -0,333333333333333  Edge (author,name) | Weight Edge = -0,333333333333333  Edge (name,journal) | Weight Edge = 0  Edge (code,non) | Weight Edge = -0,333333333333333  Edge (non,initi) | Weight Edge = 0  Edge (morphem,stem) | Weight Edge = 0  Edge (subject,statist) | Weight Edge = 0  Edge (statist,determin) | Weight Edge = -0,333333333333333  Edge (determin,relev) | Weight Edge = 0  Edge (titl,difficulti) | Weight Edge = -0,333333333333333  Edge (titl,usual) | Weight Edge = 0  Edge (content,articl) | Weight Edge = -0,2  Edge (retriev,articl) | Weight Edge = -0,333333333333333  Edge (relev,content) | Weight Edge = 0  Edge (problem,concern) | Weight Edge = -0,333333333333333  Edge (concern,make) | Weight Edge = 0  Edge (make,descript) | Weight Edge = -0,333333333333333  Edge (descript,titl) | Weight Edge = 0  Edge (difficulti,involv) | Weight Edge = 0  Edge (involv,automat) | Weight Edge = -0,333333333333333  Edge (automat,retriev) | Weight Edge = 0  Edge (articl,titl) | Weight Edge = 0  Edge (articl,approxim) | Weight Edge = 0  Edge (approxim,titl) | Weight Edge = -0,2  Edge (usual,relev) | Weight Edge = -0,2   1. Melakukan iterasi random walk dari matriks probabilitas untuk mendapatkan bobot untuk node (jika menghilangkan faktor s). 2. Membuat bobot dari node dari query no.1 sebagai *affinity score*.   Node titl | Affinity Score = 6,52608936983504  Node content | Affinity Score = 2,61043574793402  Node articl | Affinity Score = 2,61043574793402  Node relev | Affinity Score = 2,50312638297156  Node retriev | Affinity Score = 1,72915035134244  Node concern | Affinity Score = 1,63855120730034  Node difficulti | Affinity Score = 1,63855120730034  Node usual | Affinity Score = 1,63855120730034  Node make | Affinity Score = 1,47188454063368  Node descript | Affinity Score = 1,47188454063368  Node approxim | Affinity Score = 1,47188454063368  Node involv | Affinity Score = 1,30521787396701  Node automat | Affinity Score = 1,13855120730034  Node problem | Affinity Score = 1,11023037372662   1. Membuat kandidat *term* yang merupakan kombinasi dari satu sampai tiga kata yang didapatkan dari tiga *node* dengan *affinity score* tertinggi.   Kandidat term = titl, content, articl  Kombinasi term :   1. titl 2. content 3. articl 4. titl content 5. titl articl 6. content articl 7. titl content articl 8. Melakukan ranking pada kandidat term menggunakan persamaan II.14, namun tanpa menggunakan faktor *z*   titl | Candidate Term Score = 6,52608936983504  titl content | Candidate Term Score = 4,56826255888453  titl articl | Candidate Term Score = 4,56826255888453  titl content articl | Candidate Term Score = 3,91565362190103  content | Candidate Term Score = 2,61043574793402  articl | Candidate Term Score = 2,61043574793402  content articl | Candidate Term Score = 2,61043574793402   1. Memilih kandidat term dengan candidate term score tertinggi untuk dijadikan query baru   Hasil :  Reformulasi query no.1 : titl |

## Rencana Pengujian *Phrase Rank* pada Koleksi Dokumen

Pada bagian ini akan dijelaskan tentang bagaiman pengujian dilakukan dengan algoritma *PhRank* yang telah diimplementasi. Jumlah *top ranked pseudo relevant document* (*k*) yang ditentukan di sini adalah *k* = 2, 5, 10, 20. Terdapat tiga fitur yang akan dijadikan bahan perbandingan di sini :

* 1. Faktor *r* : faktor yang mendukung dalam kebergantungan *word* pada *term*.
  2. Faktor *s* : faktor yang mendukung dalam ketidakbergantungan *word*.
  3. Faktor *z* : faktor yang mendukung *term* sebagai *elemental unit*.

Pada algoritma *phrase rank* akan dilakukan empat uji :

1. Pengujian dengan menghilangkan faktor *s*, dan menggunakan faktor *r* dan *z* yang akan disebut dengan *sF*.
2. Pengujian dengan menghilangkan faktor *r*, dan menggunakan faktor *s* dan *z* yang akan disebut dengan *rF*.
3. Pengujian dengan menghilangkan faktor *z*, dan menggunakan faktor *s* dan *r* yang akan disebut dengan *zF*.
4. Pengujian dengan menggunakan faktor *s*, *r*, dan *z* yang akan disebut dengan *sTrTzT*.

Untuk setiap *k* yang sudah ditentukan dilakukan semua uji terhadap algoritma *phrase rank*, baik yang menggunakan kandidat term dengan kombinasi satu sampai tiga kata, maupun kandidat term dengan kombinasi satu sampai enam kata pada semua koleksi dokumen. Kemudian akan dilakukan perbandingan performansi antara sistem *information retrieval* standar, sistem *information retrieval* yang mengimplementasi standar *phrase rank*, dan sistem *information retrieval* yang mengimplementasi *phrase rank* yang dimodifikasi. Pada penelitian koleksi TREC ditunjukkan bahwa performansi terbaik untuk deskripsi topik didapatkan ketika menghilangkan faktor *z* atau dengan *zF*.

# PENGUJIAN

## Batasan Pengujian

Berikut adalah batasan yang digunakan pada pengujian sistem *information retrieval*.

1. Koleksi uji yang digunakan terdiri dari tiga *file*, yaitu file koleksi dokumen, koleksi *query*, dan *relevance judgement*.
2. Koleksi uji harus sesuai dengan format yang sudah ditentukan. Berikut adalah format untuk koleksi uji yang digunakan.
3. Koleksi dokumen

Tabel IV.1 Format koleksi dokumen

|  |
| --- |
| .I  <nomor dokumen>  .W  <isi dokumen> |

Konten dari koleksi dokumen minimal harus mengandung kedua tag *.I* dan *.W* untuk setiap dokumennya. Jika tidak mengandung kedua tag tersebut, konten tidak diidentifikasi sebagai dokumen.

1. Koleksi *query*

Tabel IV.2 Format koleksi *query*

|  |
| --- |
| .I  <nomor *query*>  .W  <isi *query*> |

Konten dari koleksi *query* minimal harus mengandung kedua tag .*I* dan .*W* untuk setiap *query*-nya. Jika tidak mengandung kedua tag tersebut, konten tidak diidentifikasi sebagai *query*.

1. *Relevance* *judgement*

Tabel IV.3 Format *relevance judgement*

|  |
| --- |
| <nomor *query*> <nomor dokumen yang relevan terhadap *query*> |

## Koleksi Uji yang Digunakan

Pengujian dilakukan pada tiga koleksi uji, yaitu ADI, CISI, dan CRAN. Jumlah dokumen dan *query* yang terdapat pada masing-masing koleksi uji adalah sebagai berikut.

Tabel IV.4 Koleksi uji yang digunakan

|  |  |  |
| --- | --- | --- |
| Koleksi uji | Jumlah Dokumen | Jumlah *Query* |
| ADI | 82 | 35 |
| CISI | 1460 | 112 |
| CRAN | 1400 | 225 |

## Pengujian Sistem *Information Retrieval* yang Mengimplementasi *Phrase Rank* Standar

Berikut adalah hasil pengujian pada semua koleksi uji yang diperoleh dari evaluasi sistem *information retrieval convensional* dan sistem *information retrieval* yang mengimplementasi *phrase rank* standar.

### Koleksi ADI

Tabel IV.5 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada ADI untuk *k* = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Convensional* | ***PhRank* (*k* = 2)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| 0,424252103 | 0,352668 | 0,354121 | 0,485534 | 0,354121 |

Tabel IV.6 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada ADI untuk *k* = 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Convensional* | ***PhRank* (*k* = 5)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| 0,424252103 | 0,287896 | 0,386671 | 0,360925 | 0,386671 |

Tabel IV.7 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada ADI untuk *k* = 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Convensional* | ***PhRank* (*k* = 10)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| 0,424252103 | 0,303886 | 0,366996 | 0,318259 | 0,366996 |

Tabel IV.8 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada ADI untuk *k* = 20

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Convensional* | ***PhRank* (*k* = 20)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| 0,424252103 | 0,290451 | 0,31103 | 0,239873 | 0,31103 |

### Koleksi CISI

Tabel IV.9 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CISI untuk *k* = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Convensional* | ***PhRank* (*k* = 2)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| 0,149269914 | 0,120611 | 0,078471 | 0,257244 | 0,078471 |

Tabel IV.10 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CISI untuk *k* = 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Convensional* | ***PhRank* (*k* = 5)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| 0,149269914 | 0,121537 | 0,088515 | 0,253143 | 0,088515 |

Tabel IV.11 Perbandingan NIAP antara sistem *information retrieval convensional* dan *phrase rank* standar pada CISI untuk k = 10.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Convensional* | ***PhRank* (*k* = 10)** | | | |
| sF | rF | zF | sTrTzT |
| 0,149269914 | 0,114203 | 0,103633 | 0,1987 | 0,103633 |

Tabel IV.12 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CISI untuk *k* = 20.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Convensional* | ***PhRank* (*k* = 20)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| 0,149269914 | 0,127974 | 0,114171 | 0,163374 | 0,114171 |

### Koleksi CRAN

Tabel IV.13 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CRAN untuk *k* = 2.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Convensional* | ***PhRank* (*k* = 2)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| 0,388610023 | 0,214259 | 0,187267 | 0,31578 | 0,187267 |

Tabel IV.14 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CRAN untuk *k* = 5.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Convensional* | ***PhRank* (*k* = 5)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| 0,388610023 | 0,220331 | 0,17344 | 0,283851 | 0,17344 |

Tabel IV.15 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CRAN untuk *k* = 10.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Convensional* | ***PhRank* (*k* = 10)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| 0,388610023 | 0,213037 | 0,188107 | 0,240514 | 0,188107 |

Tabel IV.16 Perbandingan NIAP antara sistem *information retrieval* *convensional* dan *phrase rank* standar pada CRAN untuk *k* = 20.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Convensional* | ***PhRank* (*k* = 20)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| 0,388610023 | 0,204479 | 0,150722 | 0,19675 | 0,150722 |

## Pengujian Sistem *Information Retrieval* yang Mengimplementasi Modifikasi *Phrase Rank*

Berikut adalah hasil pengujian pada semua koleksi uji yang diperoleh dari evaluasi sistem *information retrieval* yang mengimplementasi *phrase rank* standar dan sistem *information retrieval* yang mengimplementasi *phrase rank* yang dimodifikasi.

### Koleksi ADI

Tabel IV.17 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada ADI untuk *k* = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Phrase Rank* yang Diuji | ***PhRank* (*k* = 2)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| *PhRank* standar | 0,352668 | 0,354121 | 0,485534 | 0,354121 |
| *PhRank* modifikasi | 0,354696 | 0,383634 | 0,485534 | 0,383634 |

Tabel IV.18 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada ADI untuk *k* = 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Phrase Rank* yang Diuji | ***PhRank* (*k* = 5)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| *PhRank* standar | 0,287896 | 0,386671 | 0,360925 | 0,386671 |
| *PhRank* modifikasi | 0,302457 | 0,318957 | 0,360925 | 0,318957 |

Tabel IV.19 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada ADI untuk *k* = 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Phrase Rank* yang Diuji | ***PhRank* (*k* = 10)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| *PhRank* standar | 0,303886 | 0,366996 | 0,318259 | 0,366996 |
| *PhRank* modifikasi | 0,31774 | 0,239144 | 0,318259 | 0,239144 |

Tabel IV.20 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada ADI untuk *k* = 20

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Phrase Rank* yang Diuji | ***PhRank* (*k* = 20)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| *PhRank* standar | 0,290451 | 0,31103 | 0,239873 | 0,31103 |
| *PhRank* modifikasi | 0,343444 | 0,336224 | 0,239873 | 0,336224 |

### Koleksi CISI

Tabel IV.21 perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CISI untuk *k* = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Phrase Rank* yang Diuji | ***PhRank* (*k* = 2)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| *PhRank* standar | 0,120611 | 0,078471 | 0,257244 | 0,078471 |
| *PhRank* modifikasi | 0,118673 | 0,09125 | 0,257244 | 0,09125 |

Tabel IV.22 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CISI untuk *k* = 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Phrase Rank* yang Diuji | ***PhRank* (*k* = 5)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| *PhRank* standar | 0,121537 | 0,088515 | 0,253143 | 0,088515 |
| *PhRank* modifikasi | 0,126212 | 0,094474 | 0,253143 | 0,094474 |

Tabel IV.23 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CISI untuk *k* = 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Phrase Rank* yang Diuji | ***PhRank* (*k* = 10)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| *PhRank* standar | 0,114203 | 0,103633 | 0,1987 | 0,103633 |
| *PhRank* modifikasi | 0,109358 | 0,087924 | 0,1987 | 0,087924 |

Tabel IV.24 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CISI untuk *k* = 20

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Phrase Rank* yang Diuji | ***PhRank* (*k* = 20)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| *PhRank* standar | 0,127974 | 0,114171 | 0,163374 | 0,114171 |
| *PhRank* modifikasi | 0,130941 | 0,08691 | 0,163374 | 0,08691 |

### Koleksi CRAN

Tabel IV.25 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CRAN untuk *k* = 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Phrase Rank* yang Diuji | ***PhRank* (*k* = 2)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| *PhRank* standar | 0,214259 | 0,187267 | 0,31578 | 0,187267 |
| *PhRank* modifikasi | 0,222808 | 0,093737 | 0,31578 | 0,093737 |

Tabel IV.26 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CRAN untuk *k* = 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Phrase Rank* yang Diuji | ***PhRank* (*k* = 5)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| *PhRank* standar | 0,220331 | 0,17344 | 0,283851 | 0,17344 |
| *PhRank* modifikasi | 0,219129 | 0,093483 | 0,283851 | 0,093483 |

Tabel IV.27 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CRAN untuk *k* = 10

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Phrase Rank* yang Diuji | ***PhRank* (*k* = 10)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| *PhRank* standar | 0,213037 | 0,188107 | 0,240514 | 0,188107 |
| *PhRank* modifikasi | 0,212113 | 0,098258 | 0,240514 | 0,098258 |

Tabel IV.28 Perbandingan NIAP antara *phrase rank* standar dan *phrase rank* modifikasi pada CRAN untuk *k* = 20

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| *Phrase Rank* yang Diuji | ***PhRank* (*k* = 20)** | | | |
| *sF* | *rF* | *zF* | *sTrTzT* |
| *PhRank* standar | 0,204479 | 0,150722 | 0,19675 | 0,150722 |
| *PhRank* modifikasi | 0,220095 | 0,102217 | 0,19675 | 0,102217 |

## Analisis Hasil Pengujian

* 1. Implementasi algoritma *phrase rank* terbukti dapat meningkatkan performansi sistem *information retrieval*. Hal ini dapat diketahui dengan melihat perbandingan NIAP antara sistem information *retrieval convensional* dan sistem *information retrieval* yang mengimplementasi algoritma *phrase rank*. Performansi *phrase rank* terbaik dari setiap koleksi didapatkan pada *zF* dengan *k* = 2. Hasil ini sesuai dengan hasil yang didapatkan dari penelitian. Pada *zF* dengan *k* = 2, NIAP koleksi ADI meningkat 14,33% dan NIAP koleksi CISI meningkat 72,3%. Namun, NIAP koleksi CRAN menurun 18,736% pada *zF* dengan *k* = 2. Hal ini dapat disebabkan oleh dua faktor, yaitu *phrase rank* memilih *term* yang kurang penting disebabkan banyaknya *word co-occurrence* pada *query* atau banyaknya dokumen yang tidak relevan dalam *N,* dan terdapatnya lebih dari satu fokus, terutama pada *query* panjang. *Query* pada koleksi CRAN cenderung pendek, sedangkan jumlah dokumen pada CRAN cukup banyak, yaitu 1400 dengan konten dokumen yang cukup panjang*.* Tentu saja, keadaan ini dapat mendukung faktor yang telah disebutkan di atas. Oleh karena itu, koleksi CRAN kurang cocok dengan algoritma *phrase rank*.
  2. Implementasi algoritma *phrase rank* yang dimodifikasi terbukti dapat lebih baik dibandingkan dengan algoirtma *phrase rank* standar pada *k* tertentu dan fitur tertentu. Hal ini dapat diketahui dengan melihat hasil perbandingan NIAP antara sistem *information retrieval* yang mengimplementasi *phrase rank* standar dan sistem *information retrieval* yang mengimplementasi *phrase rank* yang dimodifikasi. Pada koleksi ADI, untuk *k* = 2, NIAP meningkat 0,567% pada *sF*, 8,33% pada *rF*, dan 0,567% pada *sTrTzT*, untuk *k* = 5, NIAP meningkat 5,07% pada *sF*, untuk *k* = 10, NIAP meningkat 4,54% pada *sF* , untuk *k* = 20, NIAP meningkat pada 18,2 % pada *sF*, 8,1% pada *rF*, dan 8,1% pada *sTrTzT*. Pada koleksi CISI, untuk k = 2, NIAP meningkat 16,178% pada *rF* dan 16,178% pada *sTrTzT*, untuk *k* = 5, NIAP meningkat 3,87% pada *sF*, 6,78% pada *rF*, dan 3,87% pada *sTrTzT*, untuk *k* = 20, NIAP meningkat 2,266% pada *sF*. Pada koleksi CRAN, untuk *k* = 2, NIAP meningkat 3,966% pada *sF*, untuk *k* = 20, NIAP meningkat 7,63% pada *sF*. Hasil ini sesuai dengan hasil penelitian, yaitu performansi *phrase rank* stabil pada variasi jumlah *term* yang dipilih.
  3. Pada kasus *word co-occurrence* yang tinggi, *word co-occurrence* ini memiliki derajat graf yang lebih tinggi sehingga *weight* cenderung terakumulasi saat *random walk*. Solusinya adalah dengan mengurangi jumlah iterasi pada *random walk*. Selain itu, saat dokumen yang tidak relevan terdapat pada *N*, keterhubungan graf akan berkurang dan memengaruhi keseimbangan *affinity score* pada *word*.

# KESIMPULAN DAN SARAN

## Kesimpulan

Berikut adalah kesimpulan yang didapatkan dari pengerjaan tugas akhir.

1. Implementasi algoritma *phrase rank* terbukti dapat meningkatkan performansi pada sistem *information retrieval*. Namun, banyaknya *word co-occurrence* pada *query* atau banyaknya dokumen yang tidak relevan dalam *N,* dan terdapatnya lebih dari satu fokus, terutama pada *query* panjang, dapat menurunkan performansi dari *phrase rank*.
2. Implementasi algoritma *phrase rank* yang dimodifikasi terbukti dapat lebih baik dibandingkan dengan algoirtma *phrase rank* standar pada *k* tertentu dan fitur tertentu. Namun, perbedaan performansi tidak signifikan.

## Saran

1. Agar lebih memastikan performansi dapat meningkat gunakan koleksi uji yang sesuai dengan kriteria *phrase rank*, yaitu tidak memiliki banyak *word co-occurrence* pada *query*, tidak memiliki dokumen tidak relevan yang banyak dalam *N*, dan memiliki satu fokus pada *query*.
2. Dapat dilakukan pendekatan lain untuk memodifikasi *phrase rank* agar menjadi lebih baik, seperti efisiensi waktu.

# DAFTAR PUSTAKA

Goldman, M. (2008). *Statistics for Bioinformatics.* Berkeley: University of California, Berkeley.

Grinstead, C., & Snell, J. (1997). *Introduction to Probability.* Swarthmore, Dartmouth: American Mathematical Society.

Mandala, R & Hendra, S. (2002). *Peningkatan Performansi Sistem Temu Balik Informasi dengan Perluasan Query secara Otomatis.* Bandung: Proceeding the 3rd Annual Meeting of the Indonesian Digital Library Networks.

Manning, C., Raghavan, P., & Schutze, H. (2009). *An Introduction to Information Retrieval.* Cambridge: Cambridge University Press.

Maxwell, K. (2014). *Term Selection in Information Retrieval.* Edinburgh: University of Edinburgh.

Maxwell, K., & Croft, W. (2013). *Compact Query Term Selection using Topically Related Text.*

Metzler, D. (2007). *Beyond Bags of Words: Effectively Modeling Dependence and Features in Information Retrieval.* Amherst: University of Massachusetts.