PERINGKASAN ARTIKEL BERBAHASA INDONESIA MENGGUNAKAN *TEXTRANK* DENGAN PEMBOBOTAN BM25

PROPOSAL SKRIPSI

Disusun oleh:

Yurdha Fadhila Hernawan

NIM: 165150200111094



PROGRAM STUDI TEKNIK INFORMATIKA

JURUSAN TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS BRAWIJAYA

MALANG

2019

DAFTAR ISI

[DAFTAR ISI i](#_Toc26960021)

[DAFTAR TABEL iii](#_Toc26960022)

[DAFTAR GAMBAR iv](#_Toc26960023)

[BAB 1 PENDAHULUAN 1](#_Toc26960024)

[1.1 Latar Belakang 1](#_Toc26960025)

[1.2 Rumusan Masalah 2](#_Toc26960026)

[1.3 Tujuan 2](#_Toc26960027)

[1.4 Manfaat 2](#_Toc26960028)

[1.5 Batasan Masalah 2](#_Toc26960029)

[1.6 Sistematika Pembahasan 2](#_Toc26960030)

[BAB 2 LANDASAN KEPUSTAKAAN 4](#_Toc26960031)

[2.1 Kajian Pustaka 4](#_Toc26960032)

[2.2 Dasar Teori Peringkasan Teks 6](#_Toc26960033)

[2.3 *Preprocessing* 7](#_Toc26960034)

[2.3.1 Segmentasi 7](#_Toc26960035)

[2.3.2 *Cleaning* 8](#_Toc26960036)

[2.3.3 Tokenisasi 8](#_Toc26960037)

[2.3.4 Stemming 8](#_Toc26960038)

[2.4 *Term Weighting* 8](#_Toc26960039)

[2.4.1 *Term Frequency (tf)* 8](#_Toc26960040)

[2.4.2 *Document Frequency (df)* 9](#_Toc26960041)

[2.4.3 *Inverse Document Frequency (Idf)* 9](#_Toc26960042)

[2.5 *TextRank* 10](#_Toc26960043)

[2.6 Fungsi *Similarity* BM25 10](#_Toc26960044)

[2.7 *PageRank* 11](#_Toc26960045)

[2.8 Evaluasi 12](#_Toc26960046)

[BAB 3 METODOLOGI 14](#_Toc26960047)

[3.1 Jenis Penelitian 14](#_Toc26960048)

[3.2 Pengambilan Data 14](#_Toc26960049)

[3.3 Metode Penelitian 14](#_Toc26960050)

[3.4 Peralatan Pendukung 14](#_Toc26960051)

[3.5 Teknik Analisis Data 15](#_Toc26960052)

[BAB 4 perancangan 16](#_Toc26960053)

[4.1 Perancangan Sistem 16](#_Toc26960054)

[4.1.1 Perancangan Umum Sistem Peringkasan Otomatis 16](#_Toc26960055)

[4.1.2 Perancangan *Preprocessing* 17](#_Toc26960056)

[4.1.3 Perancangan *Term Weighting* 18](#_Toc26960057)

[4.1.4 Perancangan Perhitungan BM25 19](#_Toc26960058)

[4.1.5 Perancangan Perhitungan *PageRank* 20](#_Toc26960059)

[4.1.6 Perancangan Mendapatkan Ringkasan 21](#_Toc26960060)

[4.2 Manualisasi 21](#_Toc26960061)

[4.2.1 Data Manualisasi 22](#_Toc26960062)

[4.2.2 *Preprocessing* 22](#_Toc26960063)

[4.2.3 *Term Weighting* 26](#_Toc26960064)

[4.2.4 Perhitungan BM25 29](#_Toc26960065)

[4.2.5 Perhitungan *PageRank* 32](#_Toc26960066)

[4.2.6 Mendapatkan Ringkasan 35](#_Toc26960067)

[4.3 Perancangan evaluasi 36](#_Toc26960068)

[BAB 5 IMPLEMENTASI 38](#_Toc26960069)

[5.1 Implementasi *Preprocessing* 38](#_Toc26960070)

[5.2 Implementasi *Term Weighting* 40](#_Toc26960071)

[5.3 Implementasi Perhitungan BM25 42](#_Toc26960072)

[5.4 Implementasi Perhitungan *PageRank* 42](#_Toc26960073)

[5.5 Implementasi Mendapatkan Ringkasan 43](#_Toc26960074)

[DAFTAR RUJUKAN 45](#_Toc26960075)

DAFTAR TABEL

[Tabel 2.1 Hasil Segmentasi Dokumen 7](#_Toc26960076)

[Tabel 2.2 Hasil Perhitungan *Tf* 9](#_Toc26960077)

[Tabel 2.3 Hasil Perhitungan *Df* dan *Idf* 9](#_Toc26960078)

[Tabel 2.4 Contoh *Confusion Matrix* 12](#_Toc26960079)

[Tabel 4.1 Hasil Segmentasi 22](#_Toc26960080)

[Tabel 4.2 Hasil *Cleaning* 23](#_Toc26960081)

[Tabel 4.3 Hasil Stemming 25](#_Toc26960082)

[Tabel 4.4 Hasil Tokenisasi 26](#_Toc26960083)

[Tabel 4.5 Hasil Perhitungan *tf* 26](#_Toc26960084)

[Tabel 4.6 Hasil Perhitungan *df* 27](#_Toc26960085)

[Tabel 4.7 Hasil Perhitungan *idf* 28](#_Toc26960086)

[Tabel 4.8 Hasil Perhitungan BM25 31](#_Toc26960087)

[Tabel 4.9 Inisialisasi Awal Bobot *PageRank* 32](#_Toc26960088)

[Tabel 4.10 Hasil Perhitungan Bobot *PageRank* 35](#_Toc26960089)

[Tabel 4.11 Hasil Pengurutan Bobot *PageRank* 35](#_Toc26960090)

[Tabel 4.12 Hasil Ringkasan Dokumen 36](#_Toc26960091)

[Tabel 4.13 Perancangan Evaluasi Sistem Compression Rate 5% 36](#_Toc26960092)

[Tabel 4.14 Perancangan Evaluasi Sistem Compression Rate 10% 36](#_Toc26960093)

[Tabel 4.15 Perancangan Evaluasi Sistem Compression Rate 20% 37](#_Toc26960094)

[Tabel 4.16 Perancangan Evaluasi Sistem Compression Rate 30% 37](#_Toc26960095)

DAFTAR GAMBAR

[Gambar 2.1 Hasil Tokenisasi Kalimat Index 1 8](file:///D:\college\script\p1\skripsi%20edited%20from%20flowchart%20-%20Copy.docx#_Toc26960096)

[Gambar 2.2 Hasil *Stemming* Kalimat Index 1 8](file:///D:\college\script\p1\skripsi%20edited%20from%20flowchart%20-%20Copy.docx#_Toc26960097)

[Gambar 2.3 Struktur *Textrank* Sebagai Graf 10](#_Toc26960098)

[Gambar 2.4 Struktur *Textrank* Setelah Perhitungan *Pagerank* 12](#_Toc26960099)

[Gambar 3.1 Gambaran Umum Proses Peringkasan *Single Document* 15](#_Toc26960100)

[Gambar 4.1 Diagram Alir Sistem 17](#_Toc26960101)

[Gambar 4.2 Diagram Alir *Preprocessing* 17](#_Toc26960102)

[Gambar 4.3 Diagram Alir *Term Weighting* 19](#_Toc26960103)

[Gambar 4.4 Diagram Alir Perhitungan BM25 20](#_Toc26960104)

[Gambar 4.5 Diagram Alir Perhiungan *PageRank* 21](#_Toc26960105)

[Gambar 4.6 Diagram Alir Pembentukan Ringkasan 21](#_Toc26960106)

[Gambar 4.7 Dokumen Manualisasi 22](#_Toc26960107)

# PENDAHULUAN

## Latar Belakang

Penggunaan internet sebagai sumber informasi telah membawa manusia pada era *one click away*. Apa pun bisa diakses di mana pun kapan pun, baik secara visual maupun tidak. Namun, apakah setiap informasi yang diakses selalu sesuai dengan konteks yang diinginkan? Bisa dikatakan hanya sedikit pengguna yang dapat memahami semua informasi ketika membaca sebuah tulisan panjang (Niu et al., 2016). Kesulitan tersebut akan membuat pengguna untuk membaca ulang, sehingga akan menghabiskan banyak waktu.

Untuk memudahkan pengguna internet dalam mendapatkan informasi yang ringkas dengan tidak merusak atau menghilangkan informasi penting, maka dibutuhkan suatu peringkasan otomatis (Abbasi-ghalehtaki et al., 2016). Berdasarkan metode yang digunakan, peringkasan teks dapat dikategorikan dalam dua bentuk, yaitu ekstraktif dan abstraktif. Ringkasan ekstraktif merupakan ringkasan yang terdiri atas kumpulan dari bagian-bagian penting suatu tulisan yang dapat mewakili keseluruhan teks, sedangkan ringkasan abstraktif merupakan ringkasan yang terdiri dari kalimat baru yang dapat merepresentasikan konteks tulisan dalam bentuk lain. Selain itu, peringkasan teks juga dapat dikelompokkan berdasarkan dokumen yang digunakan menjadi *single document* dan *multi-document* (Fang et al., 2017)*.*

Masalah utama yang muncul setelah melakukan peringkasan adalah kualitas hasil peringkasan. Apakah konteks yang dibicarakan pada hasil ringkasan sudah dapat merepresentasikan tulisan secara utuh. Penelitian oleh Pinandhita (2013) telah melakukan peringkasan ekstraktif teks berbahasa Indonesia dengan menggunakan kata benda yang terdapat dalam sebuah dokumen. Penelitian tersebut menghitung nilai kemiripan (*similarity*) antara kalimat berdasarkan kata benda yang terdapat pada setiap kalimat, lalu memeringkatkan kalimat yang paling penting dengan mengurutkan total nilai kemiripan tersebut. Evaluasi dilakukan dengan membandingkan hasil kinerja beberapa metode *similarity* agar dapat menentukan metode *similarity* terbaik dalam melakukan peringkasan.

Selain itu, peneliti lain telah melakukan penelitian untuk mendapatkan ringkasan secara abstraktif (Niu et al., 2016). Penelitian tersebut menggunakan dokumen teks opini pendek berbahasa China dengan mengelompokkan teks yang mirip lalu meringkasnya. Proses pengelompokan teks yang mirip dilakukan dengan menggunakan metode *K-Means* dengan fitur yang didapatkan dari nilai *word2vec* (nilai hubungan kedekatan antara kata). Hasil pengelompokan teks akan diperingkatkan menggunakan metode *TextRank.* Kalimat yang dianggap penting dan dapat dijadikan ringkasan memiliki peringkat tertinggi. Hasil pemeringkatan akan dijadikan ringkasan dengan menggunakan *encoder-decoder Reccurent Neural Network (RNN)* untuk membentuk kalimat baru.

Penelitian sebelumnya yang menjadi acuan dalam penelitian ini adalah peringkasan esktraktif menggunakan data *statistical* (Mussina, Aubakirov and Trigo, 2018). Data *statistical* merupakan nilai kesamaan *text units* dalam sebuah dokumen. *Text units* dalam dokumen dapat berupa kata, kalimat atau paragraf. Penelitian tersebut menggunakan metode *TextRank* untuk merepresentasikan kalimat dalam sebuah dokumen dan menggunakan beberapa metode *similarity.*

Berdasarkan pemaparan di atas, penulis mengajukan penelitian untuk melakukan peringkasan otomatis dengan objek artikel berita *online* berbahasa Indonesia menggunakan metode pemeringkatan *TextRank* dan BM25 sebagai fungsi *similarity.* BM25 dipilih berdasarkan penelitian sebelumnya yang telah melakukan pengujian menggunakan beberapa fungsi *similarity* dan mendapatkan bahwa BM25 menghasilkan ringkasan yang lebih baik dari pada fungsi *similarity* lainnya (Barrios et al., 2016)*.*

## Rumusan Masalah

Berdasarkan latar belakang yang telah disebutkan, dapat dirumuskan masalah sebagai berikut:

1. Apakah *TextRank* dapat menghasilkan ringkasan ekstraktif otomatis dengan kualitas yang baik?
2. Bagaimana pengaruh *compression rate* dengan kualitas ringkasan?

## Tujuan

Menghasilkan ringkasan ekstraktif secara otomatis dengan kualitas yang baik menggunakan *TextRank*

Membandingkan kualitas ringkasan berdasarkan *compression rate*

## Manfaat

Selain menghasilkan ringkasan secara otomatis, penelitian ini juga dapat digunakan dalam *Question Answering System, Information retrival*, dan ekstraksi informasi.

## Batasan Masalah

Artikel online yang digunakan adalah artikel berita berbahasa Indonesia yang memiliki tulisan sesuai Ejaan Bahasa Indonesia (EBI)

Artikel online berasal dari situs resmi BBC Indonesia dengan topik bencana alam dan lingkungan sebanyak 10 artikel.

## Sistematika Pembahasan

Laporan penelitian ini terdiri atas beberapa bagian yaitu sebagai berikut.

**Bab I Pendahuluan**

Berisikan latar belakang penelitian, rumusan masalah, tujuan penelitian, manfaat, batasan masalah yang diberikan, dan sistematika pembahasan.

**Bab II Landasan Kepustakaan**

Berisikan pembahasan mengenai teori yang dijadikan pedoman pengerjaan, konsep pengerjaan penelitian, metode atau sistem yang diterapkan, pustaka ilmiah yang berkitan dengan *Text Summarize, dan TextRank*.

**Bab III Metode Penelitian**

Berisikan langkah pengerjaan dalam penelitian, teknik yang digunakan, data yang akan digunakan, dan representasi berdasarkan metode yang dipilih untuk menyelesaikan masalah.

**Bab IV Perancangan**

Berisikan proses penyelesaian masalah dengan visualisasi diagram alir, perhitungan manual dari metode yang digunakan, dan rancangan pengujian.

**Bab V Implementasi**

Berisikan implementasi metode yang digunakan dalam menyelesaikan masalah dalam sebuah sistem.

**Bab VI Pengujian dan Analisis**

Berisikan scenario pengujian dan analisis terhadap hasil pengujian.

**Bab VII Penutup**

Berisikan kesimpulan dan saran sebagai rekomendasi untuk penelitian selanjutnya.

# LANDASAN KEPUSTAKAAN

## Kajian Pustaka

Salah satu penelitian yang dilakukan untuk melakukan peringkasan adalah dengan memanfaatkan kata benda yang terdapat pada sebuah dokumen (Pinandhita, 2013). Penelitian tersebut dilakukan untuk mendapatkan ringkasan ekstraktif dengan membandingkan beberapa metode *similarity* yang didapatkan dari kata benda. Dokumen yang digunakan berupa artikel koran dengan topik di luar pertanian yang juga digunakan pada penelitian Miptahudin (2010) dan dokumen dari penelitian Aristoteles (2011). Hasil peringkasan yang didapatkan dari beberapa metode *similarity* dievaluasi untuk mendapatkan metode *similarity* terbaik dalam melakukan peringkasan.

Total percobaan yang dilakukan Pinandhita (2013) adalah sebanyak tujuh kali (dengan mengikutsertakan judul dokumen dan tanpa judul dokumen) yaitu penerapan *PageRank* dengan bobot *cosine similarity*, *PageRank* dengan bobot BM25, *PageRank* dengan bobot *content overlap*, bobot *cosine similarity* tanpa PageRank, bobot BM25 tanpa *PageRank*, bobot *content overlap* tanpa *PageRank*, dan yang terakhir menggunakan bobot koefisien *dice*. Rata rata panjang dokumen yang digunakan pada penelitian tersebut adalah sebanyak 47 kalimat yang mengandung rata rata sebanyak 282 kata benda. Pengujian dilakukan dengan mencari nilai *kappa* berdasarkan kesepakatan hasil ringkasan antara tiga orang dosen Jurusan Sastra Indonesia dan tiga orang mahasiswa. Kualitas ringkasan terbaik didapatkan dari hasil pembobotan BM25 dengan *PageRank* tanpa menggunakan judul dokumen.

Selain mendapatkan ringkasan secara ekstraktif, peneliti lain juga telah melakukan penelitian untuk mendapatkan ringkasan secara abstraktif (Niu et al., 2016). Penelitian tersebut menggunakan dokumen teks opini pendek berbahasa China dengan mengelompokkan teks yang mirip lalu meringkasnya. Proses peringkasan dibagi atas tiga tahap yaitu *clustering* teks (mengelompokkan), pemeringkatan teks, dan peringkasan teks. Proses *clustering* dilakukan dengan menggunakan metode *K-Means* dengan fitur yang telah didapatkan dari nilai *word2vec*. *Word2vec* menunjukan hubungan kedekatan antara suatu kata dengan kata lainnya.

Hasil *clustering* yang telah didapatkan akan diperingkatkan menggunakan model pemeringkatan graf yaitu *TextRank* dengan bantuan pembobotan *similarity* BM25, setelahnya hasil pemeringkatan tertinggi akan diringkas secara abstraktif. Peringkasan abstraktif dilakukan dengan metode *encoder-decoder Reccurent Neural Network (RNN)*. Pengujian kualitas ringkasan yang dihasilkan pada penelitian Niu et al., (2016) dilakukan dengan mencari nilai *precision, recall, F-measure, ROUGE-N, dan ROUGE-L*. Secara berurut rata-rata nilai *precision, recall, dan F-measure* yang didapatkan adalah sebesar 0,94; 0,932; dan 0,933. Nilai ROUGE-N didapatkan dari hasil kesamaan *N-gram* yang serupa dari hasil peringkasan manual dengan hasil ringkasan yang didapatkan sistem. Pada evaluasi ini digunakan ROUGE-1 (*unigrams*) dan ROUGE-2 (*bigrams*).

Peringkasan dokumen juga dapat dilakukan dengan metode klasifikasi kalimat (Fhadli, 2017). Kalimat akan diklasifikasikan sebagai kelas yang termasuk ringkasan dan kelas yang termasuk bukan ringkasan. Fitur yang digunakan dalam pengklasifikasian tersebut adalah fitur statistik dan fitur linguistik, sedangkan dokumen yang digunakan dalam penelitian tersebut adalah literatur Ilmu Komputer berbahasa Indonesia.

Fitur statistik didapatkan dari nilai TF-IDF sebuah kalimat, sedangkan fitur linguistik didapatkan dari banyaknya kata pada judul yang terdapat pada kalimat (*title word*), posisi kalimat pada dokumen (*sentence location*), panjang kalimat (*sentence length*), kalimat yang mengandung akronim atau singkatan (*upper-case word*), kalimat yang memiliki frasa yang dianggap penting seperti "jadi", "hasilnya", dan "kesimpulannya" (*cue phrase*), kalimat yang mengandung kata spesifik yang menyatakan sebuah topik dokumen (*biased word*), dan kalimat yang mengandung kata-kata yang dianggap tidak penting (*occurrence of non-essential information*). Proses pengklasifikasian kalimat dilakukan dengan menggunakan metode Naive Bayes. Naive Bayes bekerja dengan cara menghitung peluang suatu kalimat terhadap kelas dengan bantuan data latih. Hasil kualitas ringkasan diuji dengan melakukan pencarian nilai precision, recall, F-measure, dan relative utility. Hasil rata-rata F-measure dan relative utility yang didapatkan adalah 0,206538 dan 0,116657.

Salah satu penelitian yang menjadi acuan penelitian yang akan dilakukan penulis adalah pencarian ringkasan secara ekstraktif oleh Mussina, Aubakirov & Trigo (2018). Ringkasan yang dihasilkan tidak melakukan perubahan terhadap struktur kalimat. Cara yang digunakan untuk mendapatkan ringkasan ekstraktif adalah menggunakan data *statistical,* yaitu dengan menghitung kesamaan *text units*. *Text units* tersebut dapat berupa kata, kalimat atau paragraf. Penelitian tersebut menggunakan dokumen mengenai bencana alam berbahasa Rusia dan Kazakh, yang mana memiliki struktur kalimat yang jelas.

Penelitian Mussina, Aubakirov & Trigo (2018) merepresentasikan dokumen menggunakan metode *TextRank,* dengan kalimat sebagai node dan nilai *similarity* antara kalimat sebagai *edges.* Panjang ringkasan yang diambil adalah sebanyak 30% dari total panjang dokumen. Nilai *similarity* yang digunakan adalah *content overlap,* BM25*,* dan *substring* terpanjang yang muncul antara dua kallimat. Namun, penelitian tersebut tidak menggunakan metode *PageRank* dalam proses pemeringkatan kalimat, melainkan dengan menjumlahkan seluruh nilai *similarity* yang dimiliki kalimat dan mengurutkannya berdasarkan jumlah nilai terbesar. Secara umum, setiap kalimat akan saling berkaitan antara satu dengan yang lainnya, namun terkadang ada kalimat yang tidak memiliki kesamaan dengan kalimat lain. Kalimat tersebut tidak memiliki nilai *similarity* sehingga tidak akan dijadikan ringkasan.

Sebelum melakukan pengambilan ringkasan berdasarkan nilai fungsi *silimarity*, penelitian Mussina, Aubakirov & Trigo (2018) mereduksi kalimat berdasarkan nilai *treshold*. Kalimat yang memiliki jumlah nilai *similarity* kurang dari nilai treshold tidak dijadikan ringkasan. Pengambilan ringkasan diambil sebanyak 30% dari panjang dokumen berdasarkan kalimat yang memiliki nilai *similarity* terbesar. Kalimat terpilih dijadikan ringkasan berdasarkan urutan sesuai dokumen awal. Evaluasi pada penelitian tersebut dilakukan dengan mencari nilai distribusi *key-words,* dengan nilai rata-rata untuk nilai *similarity* menggunakan *content overlap, LongestCommonSubstring*, dan BM25secara berurutan adalah 0,180; 0,175; dan 0,169.

Penelitian lain yang menjadi acuan penelitian ini adalah percobaan untuk membandingkan hasil ringkasan ekstraktif menggunakan beberapa nilai *similarity* dalam *TextRank* (Barrios et al., 2016)*.* Cara mengidentifikasi hubungan antara kalimat satu dengan kalimat lainnya adalah dengan melakukan perhitungan kata yang sama, *cosine distance*, dan kesamaan *query* yang dianggap penting. Pada penelitian tersebut dokumen akan direpresentasikan sebagai graf dengan kalimat sebagai *node* dan nilai *similarity* sebagai *edges* (hubungan antara *nodes*). Fungsi similarity yang digunakan adalah *Longest Common Substring, cosine distance,* BM25*,* dan BM25*+* dengan proses pemeringkatan *PageRank*.

Dokumen yang digunakan pada penelitian Barrios et al., (2016)adalah *Document Understanding Conference (DUC)* yang berjumlah sebanyak 567 dokumen dengan peringkasan sebanyak 20% dari tiap panjang dokumen. Hasil ringkasan dari percobaan tersebut dievaluasi menggunakan metode *ROUGE-N* dengan nilai terbaik didapatkan pada peringkasan menggunakan BM25 dan BM25+.

## Dasar Teori Peringkasan Teks

Ketika jumlah informasi *online* semakin banyak, maka kebutuhan akan sistem yang dapat merangkum satu atau lebih dokumen secara otomatis sangat diperlukan (Radev et al., 2002). Selain dapat membantu dalam mengatasi informasi yang berlebihan, peringkasan otomatis juga berguna dalam penyajian informasi singkat mengingat ukuran perangkat *handy* yang digunakan pembaca (Sankarasubramaniam et al., 2014). Ukuran ringkasan biasanya tidak lebih dari setengah dokumen aslinya. Berdasarkan kebutuhannya ringkasan dokumen terbagi atas (Munot and S. Govilkar, 2013):

Metode

Abstraktif

Ringkasan yang terdiri dari kalimat baru yang dapat merepresentasikan konteks tulisan dalam bentuk kalimat lain.

Ekstraktif

Ringkasan yang terdiri atas kumpulan dari bagian-bagian penting suatu tulisan yang dapat mewakili keseluruhan teks.

Konten

*Generic*

Ringkasan umum yang tidak bergantung pada syarat apapun.

*Query based*

Ringkasan hanya didapatkan berdasarkan *query* yang diinginkan pengguna.

Jumlah dokumen

*Single document*

Ringkasan yang didapatkan dari satu dokumen.

*Multi document*

Ringkasan yang didapatkan dari beberapa dokumen.

Bahasa

*Monolingual*

Ringkasan yang didapatkan dari dokumen dengan bahasa yang sama.

*Multilingual*

Ringkasan yang didapatkan dari beberapa dokumen dengan bahasa yang berbeda.

Dalam penelitian ini peringkasan akan dilakukan secara ekstraktif dengan memilih kalimat yang dianggap penting dan dapat dijadikan ringkasan. Dokumen yang digunakan merupakan *single document* dengan berbahasa Indonesia.

## *Preprocessing*

*Preprocessing* dilakukan sebelum proses pembentukan ringkasan dan mempermudah proses peringkasan. Dalam *preprocessing* terdapat tiga tahapan yaitu segmentasi, *cleaning*, tokenisasi, dan *stemming*.

### Segmentasi

Segmentasi merupakan proses pemecahan teks dokumen menjadi kalimat-kalimat untuk mempermudah pemrosesan dokumen menjadi potongan-potongan yang lebih kecil. Pada penelitian yang akan dilakukan, judul pada artikel berita tidak diikutsertakan dalam percobaan. Contoh hasil segmentasi dokumen ditunjukan pada Tabel 2.1.

Tabel 2.1 Contoh Hasil Segmentasi Dokumen

|  |  |
| --- | --- |
| **Index** | **Kalimat** |
| 1 | Dia mengambilnya dengan senang hati. |
| 2 | Esoknya setelah pulang sekolah, ia melarikan diri. |
| 3 | Ibu yang bekerja sebagai pembantu. |

### *Cleaning*

Pada proses cleaning dilakukan penghilangan tanda baca dan URLkarna dianggap tidak mempengaruhi isi dokumen.

### Stemming

*Stemming* merupakan metode pembentukan kata atau term menjadi kata dasar. Biasanya proses *stemming* dilakukan dengan membuang imbuhan yang terdapat pada term. Contoh hasil stemming dokumen ditunjukan pada Tabel 2.2.

Tabel 2.2 Contoh Hasil Stemming Dokumen

|  |  |
| --- | --- |
| **Index** | **Kalimat** |
| 1 | dia ambil dengan senang hati |
| 2 | esok telah pulang sekolah ia lari diri |
| 3 | ibu yang kerja bagai bantu |

### Tokenisasi

Tokenisasi adalah metode pemecahan teks kalimat menjadi token-token (*term*) yang berurutan. Pada proses ini juga dilakukan penghilangan angka dan kata hubung atau kata yang dianggap tidak mempenggaruhi isi dari konten dokumen menggunakan metode *stopwords removal*. Hal ini juga dilakukan guna meningkatkan performa sistem agar sistem bisa secara efektif dimanfaatkan untuk pengolahan konten yang benar-benar dianggap penting saja. Contoh hasil tokenisasi dokumen ditunjukan pada Gambar 2.1.

Gambar 2.1 Contoh Hasil Tokenisasi Kalimat Index 1

ambil

senang hati

## *Term Weighting*

### *Term Frequency (tf)*

Perhitungan *term frequency* merupakan langkah awal dari *term weighting*. *Term frequency* adalah jumlah kemunculan setiap *term* dalam satu kalimat. Contoh hasil perhitungan term frequency ditunjukan pada Tabel 2.3.

Tabel 2.3 Contoh Hasil Perhitungan *Tf*

|  |  |  |  |
| --- | --- | --- | --- |
| ***Term*** | **Kalimat 1** | **Kalimat 2** | **Kalimat 3** |
| ambil | 1 | 0 | 0 |
| bantu | 0 | 0 | 1 |
| esok | 0 | 1 | 0 |
| hati | 1 | 0 | 0 |
| kerja | 0 | 0 | 1 |
| lari | 0 | 1 | 0 |
| pulang | 0 | 1 | 0 |
| sekolah | 0 | 1 | 0 |
| senang | 1 | 0 | 0 |

### *Document Frequency (df)*

*Document frequency* merupakan jumlah dokumen yang memiliki kemunculan sebuah *term*. Nilai *document frequency* akan digunakan untuk mendapatkan nilai *Inverse document frequency*.

### *Inverse Document Frequency (Idf)*

Nilai *inverse document frequency* akan digunakan untuk perhitungan BM25. Mengacu pada penelitian yang dilakukan (Lv and Zhai, 2011) perhitungan Idf ditunjukan pada Persamaan 2.1 :

Keterangan:

N = Panjang kalimat

= *Document frequency* tiap *term* dalam satu dokumen

Contoh perhitungan *df* dan *idf* ditunjukan pada Tabel 2.4.

Tabel 2.4 Contoh Hasil Perhitungan *Df* dan *Idf*

|  |  |  |
| --- | --- | --- |
| ***Term*** | ***df*** | ***idf*** |
| ambil | 1 | 0,60206 |
| bantu | 1 | 0,60206 |
| esok | 1 | 0,60206 |
| hati | 1 | 0,60206 |
| kerja | 1 | 0,60206 |
| lari | 1 | 0,60206 |
| pulang | 1 | 0,60206 |
| sekolah | 1 | 0,60206 |
| senang | 1 | 0,60206 |

## *TextRank*

*TextRank* merupakan sebuah algoritme berbasis graf yang digunakan untuk menentukan *node* mana yang paling penting dalam suatu graf (Tarau, 1973). Struktur *TextRank* ditunjukan pada Gambar 2.2. Penggunaan *TextRank* dapat dilakukan dalam melakukan penarikan keputusan. Proses melakukan *TextRank* dalam peringkasan teks ekstraktif dimulai dengan mengidentifikasi *single document* yang digunakan. Lalu setiap kalimat direpresentasikan sebagai *node* dan hubungan antara kalimat merupakan fungsi *similarity* yang direpresentasikan sebagai *edges*. *Edges* pada graf dapat memiliki arah maupun tidak. Setelah graf terbentuk, maka selanjutnya akan dilakukan pemeringkatan graf dan mengurutkan *node* yang memiliki nilai pemeringkatan paling tinggi (paling penting).Setelah diurutkan, maka peringkasan dapat diambil berdasarkan peringkat dari kalimat-kalimat tersebut.

## Fungsi *Similarity* BM25

Dalam merepresentasikan dokumen sebagai graf, *edges* didapatkan dari hasil fungsi *similarity* antar kalimat. Fungsi *similarity* didapatkan dari kemiripan isi kalimat satu dengan kalimat lainnya. Kalimat yang merepresentasikan suatu konteks dalam dokumen akan merekomendasikan kalimat lain yang memiliki konteks yang sama (Tarau, 1973). Fungsi *similarity* yang digunakan dalam penelitian ini adalah BM25. Nilai BM25 didapatkan dari perhitungan bobot *tf* dan *idf* pada setiap kata (*term*). Selain itu juga ditambahkan parameter bebas *k1*dan *b* dengan nilai *k1* sebesar 1.2 dan *b* sebesar 0.75 (Manning et al., 2009). Persamaan BM25 dijabarkan pada Persamaan 2.2:

Keterangan:

= Nilai *idf term t*

*k1* dan *b* = Parameter penskalaan terhadap *tf* dan panjang dokumen

*tftd*  = Frekuensi term t pada kalimat d

*Ld* dan *Lave*  = Panjang kalimat *d* dan rata-rata dari panjang seluruh koleksi kalimat



Gambar 2.2 Struktur *Textrank* Sebagai Graf

## *PageRank*

*PageRank* adalah metode yang digunakan dalam pemeringkatan graf. *PageRank* digunakan oleh Google untuk menentukan tingkat kepentingan halaman web. *PageRank* merupakan nilai mumerical yang menyatakan seberapa penting sebuah halaman web di internet. Singkatnya, perhitungan nilai tersebut bertambah bila halaman tersebut muncul sebagai sebuah *hyperlink* di sebuah halaman web lainnya. Semakin besar nilai yang dimiliki, maka semakin penting web tersebut. Begitu juga dengan kalimat yang saling berhubungan satu sama lain dalam sebuah graf. Kalimat yang penting akan memiliki nilai *PageRank* yang besar.

Inisialisasi awal nilai *PageRank* tiap kalimat ditentukan secara random mulai dari 0 hingga 1. Lalu sejumlah iterasi dilakukan untuk melakukan *update* bobot *PageRank* ditiap kalimat. Persamaan *PageRank* dijabarkan pada Persamaan 2.3:

Keterangan:

= Bobot nilai *PageRank* kalimat i

*d* = *Dampening factor*

= Kalimat *j* yang berhubungan dengan kalimat *i*

= Nilai fungsi *similarity* antara kalimat *j* dan *i*

= Kalimat *k* yang berhubungan dengan kalimat *j*

= Nilai fungsi *similarity* antara kalimat *j* dan *k*

= Bobot nilai *PageRank* kalimat j

*Dampening factor* (*d*) adalah nilai yang telah dihitung oleh Google Engineers dalam sistem PageRank untuk memastikan bahwa bobot *node* akan konvergen pada satu nilai. Nilai *dampening factor* bisa didapatkan dari angka random mulai dari nol hingga satu, namun 0.85 telah menjadi nilai yang umum saat menetapkan nilai *dampening factor.* Pada akhir perhitungan graf kalimat dapat diilustrasikan seperti pada Gambar 2.3 .



Gambar 2.3 Struktur *Textrank* Setelah Perhitungan *Pagerank*

## Evaluasi

Pengujian sistem peringkasan ini akan dilakukan dengan membandingkan nilai *precision, recall,* dan *f-measure* untuk setiap hasil ringkasan. Proses perhitungan tersebut dibantu dengan menggunakan *confussion matrix*.

*Confussion Matrix*

*Confusion Matrix* merupakan informasi mengenai tingkat relevansi hasil peringkasan yang didapatkan oleh sistem. Contoh *confusion matrix* ditunjukkan pada Tabel 2.5.

Tabel 2.5 Contoh *Confusion Matrix*

|  |  |  |
| --- | --- | --- |
|  | ***Relevant*** | ***Not relevant*** |
| ***Retrieved*** | A | B |
| ***Not retrieved*** | C | D |

Keterangan :

A = Semua data yang dihasilkan sistem dan relevan bagi *expert*

B = Semua data yang dihasilkan sistem dan tidak relevan bagi *expert*

C = Semua data yang relevan bagi *expert* dan tidak dihasilkan sistem

D = Semua data yang tidak relevan bagi *expert* dan tidak dihasilkan sistem

Precision

Merupakan nilai ketepatan antara informasi yang dihasilkan sistem dengan hasil informasi yang seharusnya (dianggap benar). Persamaan *precision* ditunjukan pada Persamaan 2.5:

(2.5)

Keterangan:  
A = Semua data yang dihasilkan sistem dan relevan bagi *expert*

B = Semua data yang dihasilkan sistem dan tidak relevan bagi *expert*

Recall

Merupakan tingkat keberhasilan sistem dalam peringkasan otomatis yang menentukan berapa proporsi kalimat yang dipilih oleh pakar yang juga dipilih oleh sistem. Persamaan *recall* ditunjukan pada Persamaan 2.6:

(2.6)

Keterangan:  
A = Semua data yang dihasilkan sistem dan relevan bagi *expert*

C = Semua data yang relevan bagi *expert* dan tidak dihasilkan sistem

F-Measure

Merupakan pengukuran yang menilai timbal balik antara *precision* dan *recall.* Persamaan *F-Measure* ditunjukan pada Persamaan 2.7:

(2.7)

Keterangan:  
P = Precision  
R = Recall

# METODOLOGI

## Jenis Penelitian

Penelitian yang diajukan bersifat Non-implementatif analitik. Yang mana hasil dari penelitian ini akan dibandingkan dengan ringkasan yang diambil oleh pakar untuk mengetahui kualitas ringkasan. Hasil tersebut akan dianalisis menggunakan metode evaluasi.

## Pengambilan Data

Jumlah populasi pada penelitian ini termasuk ke dalam kategori tak terbatas karna dokumen yang digunakan merupakan artikel berita online berbahasa Indonesia. Populasi tak terbatas yaitu sumber datanya tidak dapat ditentukan batasan-batasannya sehingga relatif tidak dapat ditentukan dalam bentuk angka. Teknik pengambilan sampel yang digunakan adalah *simple random sampling*. Yaitu pengambilan sampel dari anggota populasi secara acak tanpa memperhatikan tingkatan, karena populasi bersifat homogen (sama).

Instrumen pengumpulan dokumen pada penelitian ini adalah peneliti sendiri, yang mana dokumen diambil dari artikel berita online BBC Indonesia yang dapat diakses melalui situs resmi BBC Indonesia. Jumlah dokumen yang akan diambil sebagai dokumen pengujian sistem adalah sebanyak 10 artikel berita mengenai bencana alam dan lingkungan.

## Metode Penelitian

Penelitian ini terbagi atas dua tahap, yaitu pembentukan ringkasan dan evaluasi ringkasan yang dihasilkan. Tahap pembentukan ringkasan terdiri atas tahapan *preprocessing*, *term weighting,* penerapan *TextRank dan PageRank,* dan pembentukan ringkasan*.* Hasil ringkasan yang didapatkan dari penelitian ini adalah kalimat yang memiliki nilai *PageRank* yang besar setelah diurutkan. Kalimat tersebut dianggap penting dan dapat mereprenstasikan keseluruhan dokumen sehingga dapat dijadikan ringkasan. Mengacu pada penelitian yang dilakukan (Yeh et al., 2005) total kalimat yang akan diambil sebagai ringkasan ialah sebanyak 5%-30% dari total keseluruhan kalimat. Secara umum, diagram proses peringkasan teks pada penelitian ini ditunjukan pada Gambar 3.1.

## Peralatan Pendukung

Lingkungan implementasi system *hardware* meliputi:

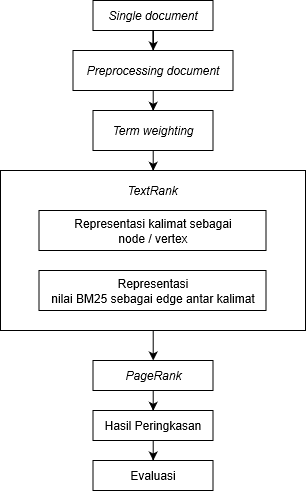
Processor 2.00 GHz

RAM 8.00 GB

Spesifikasi *software* meliputi:

1. Operating System Windows 10 64-bit

2. Jupyter Notebook



Gambar 3.1 Gambaran Umum Proses Peringkasan *Single Document*

## Teknik Analisis Data

Pengujian hasil ringkasan yang dihasilkan sistem akan dilakukan oleh dosen Bahasa Indonesia dari Universitas Brawijaya. Dosen Bahasa Indonesia tersebut dijadikan pakar dan diminta untuk memilih kalimat yang dianggap penting dan dapat dijadikan ringkasan pada sekumpulan dokumen yang diujikan. Hasil pemilihan kalimat pada tiap dokumen akan dibandingkan dengan hasil peringkasan otomatis yang didapatkan oleh sistem yang dibuat menggunakan *TextRank.* Perhitungan *precision, recall,* dan *F-Measure* digunakan untuk mengetahui kualitas ringkasan otomatis tersebut.

# perancangan

## Perancangan Sistem

### Perancangan Umum Sistem Peringkasan Otomatis

Secara umum perancangan sistem dimulai dengan masukan *single document* berupa artikel berita berbahasa Indonesia yang telah disimpan pada file jenis *txt*. Selanjutnya *preprocessing* dilakukan agar dokumen terpecah menjadi potongan kalimat yang hanya berisikan *term-term* penting. Hasil *term* akan dihitung nilai bobotnya sesuai metode *tf* dan *idf*. Setelah mendapatkan bobot tiap *term,* dokumen direpresentasikan sebagai graf dan kalimat direpresentasikan sebagai *node* dalam graf sesuai dengan metode *TextRank*. Perhitungan BM25 akan dilakukan untuk menghitung nilai kemiripan antara kalimat dan pemeringkatan kalimat dilakukan menggunakan metode *PageRank.* Selanjutnya ringkasan dipillih berdasarkan pemeringkatan tersebut. Diagram alir sistem ditunjukan pada Gambar 4.1.





Gambar 4.1 Diagram Alir Sistem

### Perancangan *Preprocessing*

Pada proses *preprocessing,* dokumen berita akan dibersihkan dengan cara menghilangkan URL, angka, symbol, dan *stopword*. *Stopword* merupakan kumpulan kata yang dianggap tidak penting dan tidak mempengaruhi isi dokumen. Setelah pembersihan dokumen, tiap kalimat akan dipecah menjadi *term* atau kata. Alur *preprocessing* ditunjukan pada Gambar 4.2.



Gambar 4.2 Diagram Alir *Preprocessing*

#### Segmentasi

Segmentasi merupakan proses pemisahan dokemen menjadi potongan-potongan kalimat. Proses segmentasi dilakukan dengan bantuan *library* NLTK dengan modul *sent\_tokenize.* Hal ini dilakukan agar pemisahan kalimat tidak hanya mengandalkan tanda titik karna pemisahan bisa terjadi ditengah nama orang atau jalan memiliki titik karna adanya singkatan atau gelar.

#### Cleaning

Pada proses ini dilakukan pembersihan dokumen dengan menghilangkan URL dan simbol. Hal ini diperlukan untuk menghilangkan bagian kalimat yang dianggap tidak penting.

#### Stemming

Stemming dilakukan untuk menghilangkan imbuhan pada kata dalam kalimat sehingga akan menghasilkan kata dasar. Proses stemming dilakukan dengan bantuan *library* stemming bahasa IndonesiaStemmer Sastrawi. Hasil stemming akan digunakan untuk proses berikutnya yaitu tokenisasi.

#### Tokenisasi

Proses tokenisasi dilakukan untuk memisahkan kalimat menjadi potongan-potongan *term* atau kata. Pada proses ini juga dilakukan penghilangan angka dan *stopword* *removal,* yaitu penghilangan kata yang dianggap tidak penting dan tidak mempengaruhi kata penting yang berkaitan dengan konteks dokumen. Jenis kata yang biasanya dihilangkan pada proses *stopword removal* adalah kata hubung.

### Perancangan *Term Weighting*

Proses *term weighting* dilakukan untuk mendapatkan bobot *Inverse Document Frequency (Idf)* yang akan digunakan pada perhitungan fungsi *similarity* BM25. *Term weighting* dimulai dengan mencari *term* unik pada dokumen, lalu menghitung jumlah munculnya *term*/kata dalam satu kalimat yang disebut juga dengan *frequency (tf).* Nilai *tf* akan digunakan untuk menghitung *document frequency (df)* yaitu jumlah dokumen yang terdapat kemunculan sebuah *term*, dan mencari nilai *Inverse Document Frequency (Idf)* menggunakan Persamaan 2.1*.* Alur *term weighting* ditunjukan pada Gambar 4.3.





Gambar 4.3 Diagram Alir *Term Weighting*

### Perancangan Perhitungan BM25

Perhitungan nilai BM25 dilakukan dengan menetapkan nilai parameter k1 dan b seperti pada Persamaan 2.2. Untuk setiap kalimat akan dilakukan perhitungan BM25 dengan kalimat lain dalam satu dokumen. Kalimat pertama akan dijadikan *query* dalam perhitungan BM25 lalu sistem akan menghitung kemiripan kalimat tersebut dengan sisa kallimat lainnya. Hal tersebut terus dilakukan hingga mencapai kalimat terakhir. Semakin besar nilai BM25antara satu kalimat dengan kalimat lain, maka semakin mirip topik yang dibicarakan pada kedua kalimat tersebut. Perancangan perhitungan BM25 ditunjukan pada Gambar 4.4.







Gambar 4.4 Diagram Alir Perhitungan BM25

### Perancangan Perhitungan *PageRank*

*PageRank* merupakan teknik pemeringkatan berbasis graf dengan memanfaatkan bobot yang dimiliki antara suatu node dengan node yang lain sesuai dengan Persamaan 2.3. Proses perhitungan *PageRank* dilakukan terhadap setiap *node* kalimat sebanyak jumlah iterasi yang ditentukan. Bobot awal *PageRank* untuk setiap kalimat ditetapkan secara random antara nol hingga satu. Perancangan perhitungan nilai *PageRank* ditunjukan pada Gambar 4.5.





Gambar 4.5 Diagram Alir Perhiungan *PageRank*

### Perancangan Mendapatkan Ringkasan

Ringkasan diambil berdasarkan urutan nilai *PageRank* terbesar dari seluruh *node* kalimat. Kalimat akan dipilih sebanyak *compression rate* dari seluruh panjang dokumen dan diurutkan sesuai dengan urutan asli dokumennya. Perancangan pengambilan ringkasan ditunjukan pada Gambar 4.6.



Gambar 4.6 Diagram Alir Pembentukan Ringkasan

## Manualisasi

Manualisasi dilakukan untuk membuktikan dan mempermudah proses implementasi system yang akan dibuat.

### Data Manualisasi

Proses manulaisasi ini dilakukan menggunakan satu dokumen dengan total 10 kalimat. Dokumen tersebut ditunjukan pada Gambar 4.7.

|  |
| --- |
| Sejumlah sekolah di Kota Palembang, Sumatera Selatan, memutuskan memulangkan siswa-siswa mereka lantaran kabut asap semakin tebal menyelimuti kota tersebut. pada Senin (14/10) pagi. Kepala SMP Negeri 7 Palembang, Siti Zubaida, mengatakan keputusan pemulangan ditempuh sesuai dengan instruksi Dinas Pendidikan Kota Palembang. "Pagi ini kami memulangkan siswa karena melihat kabut asap yang tebal dan berdampak buruk terhadap siswa, oleh karenanya atas instruksi Kadiknas Kota Palembang melalui pesan WA Grup meminta siswa dipulangkan dan belajar di rumah masing-masing saja," jelas Siti kepada radio Elshinta. Hal ini diamini Kepala Dinas Pendidikan Kota Palembang, Ahmad Zulinto, yang menyampaikan surat edaran ke semua sekolah. "Hari ini seluruh TK hingga SMP negeri dan swasta sederajat diliburkan, untuk besok dan seterusnya akan diberikan edaran lebih lanjut," kata Ahmad Zulinto kepada kantor berita Antara. Menurutnya, kalau kualitas udara Palembang masih buruk dalam beberapa hari ke depan, kegiatan belajar mengajar di tingkat TK sampai SMP akan tetap diliburkan. Akan tetapi, sebagaimana dipaparkan Kepala Dinas Pendidikan Sumatera Selatan, Widodo, kegiatan belajar mengajar di daerah yang tidak terdampak kabut asap tetap berlangsung. "Daerah yang tidak terdampak kabut asap tetap normal tetap belajar, untuk daerah yang terkategori sedang tetap belajar namun jam masuk sekolah diundur dan kami himbau memakai masker, bagi daerah terkategori parah maka siswa diberikan tugas dengan memaksimalkan kelas daring," kata Widodo kepada Antara. Hal itu belakangan dibenarkan Agus Wibowo, selaku Kepala Pusat Data, Informasi dan Humas Badan Nasional Penanggulangan Bencana (BNPB). "Melalui pesan digital, Kepala Dinas Pendidikan Kota Palembang menginstruksikan kegiatan belajar mengajar di tingkat paud, TK, SD dan SMP negeri dan swasta diliburkan hingga batas yang belum ditentukan," sebut Agus dalam siaran pers. |

Gambar 4.7 Dokumen Manualisasi

### *Preprocessing*

#### Segmentasi

Proses pemotongan dokumen menjadi kalimat-kalimat. Hasil segmentasi ditunjukan pada Tabel 4.1.

Tabel 4.1 Hasil Segmentasi

|  |  |
| --- | --- |
| **Index** | **Kalimat** |
| 1 | Sejumlah sekolah di Kota Palembang, Sumatera Selatan, memutuskan memulangkan siswa-siswa mereka lantaran kabut asap semakin tebal menyelimuti kota tersebut pada Senin (14/10) pagi. |
| 2 | Kepala SMP Negeri 7 Palembang, Siti Zubaida, mengatakan keputusan pemulangan ditempuh sesuai dengan instruksi Dinas Pendidikan Kota Palembang. |
| 3 | "Pagi ini kami memulangkan siswa karena melihat kabut asap yang tebal dan berdampak buruk terhadap siswa, oleh karenanya atas instruksi Kadiknas Kota Palembang melalui pesan WA Grup meminta siswa dipulangkan dan belajar di rumah masing-masing saja," jelas Siti kepada radio Elshinta. |
| 4 | Hal ini diamini Kepala Dinas Pendidikan Kota Palembang, Ahmad Zulinto, yang menyampaikan surat edaran ke semua sekolah. |
| 5 | "Hari ini seluruh TK hingga SMP negeri dan swasta sederajat diliburkan, untuk besok dan seterusnya akan diberikan edaran lebih lanjut," kata Ahmad Zulinto kepada kantor berita Antara. |
| 6 | Menurutnya, kalau kualitas udara Palembang masih buruk dalam beberapa hari ke depan, kegiatan belajar mengajar di tingkat TK sampai SMP akan tetap diliburkan. |
| 7 | Akan tetapi, sebagaimana dipaparkan Kepala Dinas Pendidikan Sumatera Selatan, Widodo, kegiatan belajar mengajar di daerah yang tidak terdampak kabut asap tetap berlangsung. |
| 8 | "Daerah yang tidak terdampak kabut asap tetap normal tetap belajar, untuk daerah yang terkategori sedang tetap belajar namun jam masuk sekolah diundur dan kami himbau memakai masker, bagi daerah terkategori parah maka siswa diberikan tugas dengan memaksimalkan kelas daring," kata Widodo kepada Antara. |
| 9 | Hal itu belakangan dibenarkan Agus Wibowo, selaku Kepala Pusat Data, Informasi dan Humas Badan Nasional Penanggulangan Bencana (BNPB). |
| 10 | "Melalui pesan digital, Kepala Dinas Pendidikan Kota Palembang menginstruksikan kegiatan belajar mengajar di tingkat paud, TK, SD dan SMP negeri dan swasta diliburkan hingga batas yang belum ditentukan," sebut Agus dalam siaran pers. |

#### Cleaning

Proses pembersihan kalimat dari URL dan simbol. Hasil cleaning dapat dilihat pada Tabel 4.2.

Tabel 4.2 Hasil *Cleaning*

|  |  |
| --- | --- |
| **Index** | **Kalimat** |
| 1 | Sejumlah sekolah di Kota Palembang Sumatera Selatan memutuskan memulangkan siswa siswa mereka lantaran kabut asap semakin tebal menyelimuti kota tersebut pada Senin pagi |
| 2 | Kepala SMP Negeri Palembang Siti Zubaida mengatakan keputusan pemulangan ditempuh sesuai dengan instruksi Dinas Pendidikan Kota Palembang |
| 3 | Pagi ini kami memulangkan siswa karena melihat kabut asap yang tebal dan berdampak buruk terhadap siswa oleh karenanya atas instruksi Kadiknas Kota Palembang melalui pesan WA Grup meminta siswa dipulangkan dan belajar di rumah masing masing saja jelas Siti kepada radio Elshinta |
| 4 | Hal ini diamini Kepala Dinas Pendidikan Kota Palembang Ahmad Zulinto yang menyampaikan surat edaran ke semua sekolah |
| 5 | Hari ini seluruh TK hingga SMP negeri dan swasta sederajat diliburkan untuk besok dan seterusnya akan diberikan edaran lebih lanjut kata Ahmad Zulinto kepada kantor berita Antara |
| 6 | Menurutnya kalau kualitas udara Palembang masih buruk dalam beberapa hari ke depan kegiatan belajar mengajar di tingkat TK sampai SMP akan tetap diliburkan |
| 7 | Akan tetapi sebagaimana dipaparkan Kepala Dinas Pendidikan Sumatera Selatan Widodo kegiatan belajar mengajar di daerah yang tidak terdampak kabut asap tetap berlangsung |
| 8 | Daerah yang tidak terdampak kabut asap tetap normal tetap belajar untuk daerah yang terkategori sedang tetap belajar namun jam masuk sekolah diundur dan kami himbau memakai masker bagi daerah terkategori parah maka siswa diberikan tugas dengan memaksimalkan kelas daring kata Widodo kepada Antara |
| 9 | Hal itu belakangan dibenarkan Agus Wibowo selaku Kepala Pusat Data Informasi dan Humas Badan Nasional Penanggulangan Bencana BNPB |
| 10 | Melalui pesan digital Kepala Dinas Pendidikan Kota Palembang menginstruksikan kegiatan belajar mengajar di tingkat paud TK SD dan SMP negeri dan swasta diliburkan hingga batas yang belum ditentukan sebut Agus dalam siaran pers |

#### Stemming

Merupakan proses penghilangan imbuhan yang terdapat dalam kalimat. Hasil stemming dapat dilihat pada Tabel 4.3.

Tabel 4.3 Hasil Stemming

|  |  |
| --- | --- |
| **Index** | **Kalimat** |
| 1 | jumlah sekolah di kota palembang sumatera selatan putus pulang siswa siswa mereka lantar kabut asap makin tebal limut kota sebut pada senin pagi |
| 2 | kepala smp negeri 7 palembang siti zubaida kata putus pulang tempuh sesuai dengan instruksi dinas didik kota palembang |
| 3 | pagi ini kami pulang siswa karena lihat kabut asap yang tebal dan dampak buruk hadap siswa oleh karena atas instruksi kadiknas kota palembang lalu pesan wa grup minta siswa pulang dan ajar di rumah masing masing saja jelas siti kepada radio elshinta |
| 4 | hal ini amin kepala dinas didik kota palembang ahmad zulinto yang sampai surat edar ke semua sekolah |
| 5 | hari ini seluruh tk hingga smp negeri dan swasta derajat libur untuk besok dan terus akan beri edar lebih lanjut kata ahmad zulinto kepada kantor berita antara |
| 6 | turut kalau kualitas udara palembang masih buruk dalam beberapa hari ke depan giat ajar ajar di tingkat tk sampai smp akan tetap libur |
| 7 | akan tetapi bagaimana papar kepala dinas didik sumatera selatan widodo giat ajar ajar di daerah yang tidak dampak kabut asap tetap langsung |
| 8 | daerah yang tidak dampak kabut asap tetap normal tetap ajar untuk daerah yang kategori sedang tetap ajar namun jam masuk sekolah undur dan kami himbau pakai masker bagi daerah kategori parah maka siswa beri tugas dengan maksimal kelas daring kata widodo kepada antara |
| 9 | hal itu belakang benar agus wibowo selaku kepala pusat data informasi dan humas badan nasional tanggulang bencana bnpb |
| 10 | lalu pesan digital kepala dinas didik kota palembang instruksi giat ajar ajar di tingkat paud tk sd dan smp negeri dan swasta libur hingga batas yang belum tentu sebut agus dalam siar pers |

#### Tokeniasasi

Pada proses ini dilakukan pemisahan tiap kata dan menghilangkan kata yang terdapat dalam *stopword* dan angka*.* Hasil tokenisasi dapat dilihat pada Tabel 4.4*.*

Tabel 4.4 Hasil Tokenisasi

|  |  |
| --- | --- |
| **Index** | **Kalimat** |
| 1 | sekolah', 'kota', 'palembang', 'sumatera', 'selatan', 'putus', 'pulang', 'siswa', 'lantar', 'kabut', 'asap', 'tebal', 'limut', 'senin', 'pagi' |
| 2 | kepala', 'smp', 'negeri', 'palembang', 'siti', 'zubaida', 'putus', 'pulang', 'tempuh', 'sesuai', 'instruksi', 'dinas', 'didik', 'kota' |
| 3 | pagi', 'pulang', 'siswa', 'lihat', 'kabut', 'asap', 'tebal', 'dampak', 'buruk', 'hadap', 'instruksi', 'kadiknas', 'kota', 'palembang', 'pesan', 'wa', 'grup', 'ajar', 'rumah', 'siti', 'radio', 'elshinta' |
| 4 | amin', 'kepala', 'dinas', 'didik', 'kota', 'palembang', 'ahmad', 'zulinto', 'surat', 'edar', 'sekolah' |
| 5 | tk', 'smp', 'negeri', 'swasta', 'derajat', 'libur', 'besok', 'edar', 'ahmad', 'zulinto', 'kantor', 'berita' |
| 6 | kualitas', 'udara', 'palembang', 'buruk', 'giat', 'ajar', 'tingkat', 'tk', 'smp', 'libur' |
| 7 | papar', 'kepala', 'dinas', 'didik', 'sumatera', 'selatan', 'widodo', 'giat', 'ajar', 'daerah', 'dampak', 'kabut', 'asap', 'langsung' |
| 8 | daerah', 'dampak', 'kabut', 'asap', 'normal', 'ajar', 'kategori', 'jam', 'masuk', 'sekolah', 'undur', 'himbau', 'pakai', 'masker', 'parah', 'siswa', 'tugas', 'maksimal', 'kelas', 'daring', 'widodo' |
| 9 | agus', 'wibowo', 'kepala', 'pusat', 'data', 'informasi', 'humas', 'badan', 'nasional', 'tanggulang', 'bencana', 'bnpb' |
| 10 | pesan', 'digital', 'kepala', 'dinas', 'didik', 'kota', 'palembang', 'instruksi', 'giat', 'ajar', 'tingkat', 'paud', 'tk', 'sd', 'smp', 'negeri', 'swasta', 'libur', 'batas', 'agus', 'siar', 'pers' |

### *Term Weighting*

#### Term Frequency (tf)

Pada proses ini akan dihitung jumlah kemunculan tiap kata pada setiap kalimat. Hasil perhitungan *tf* dapat dilihat pada Tabel 4.5.

Tabel 4.5 Hasil Perhitungan *tf*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ***Term*** | **Kalimat** | | | | | | | | | |
| **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |
| agus | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| ahmad | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| ajar | 0 | 0 | 1 | 0 | 0 | 2 | 2 | 2 | 0 | 2 |
| amin | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| asap | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| badan | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| batas | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| bencana | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| .. | .. | .. | .. | .. | .. | .. | .. | .. | .. | .. |
| .. | .. | .. | .. | .. | .. | .. | .. | .. | .. | .. |
| udara | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| undur | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| wa | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| wibowo | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| widodo | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| zubaida | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| zulinto | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| **Panjang kalimat** | 17 | 15 | 25 | 11 | 12 | 11 | 15 | 25 | 12 | 23 |
| **Rata-rata panjang kalimat** | 16,6 | | | | | | | | | |

#### Document Frequency (df)

Pada proses ini akan dihitung jumlah kemunculan kata pada seluruh kalimat. Hasil perhitungan *df* dapat dilihat pada Tabel 4.6.

Tabel 4.6 Hasil Perhitungan *df*

|  |  |
| --- | --- |
| ***Term*** | ***df*** |
| agus | 2 |
| ahmad | 2 |
| ajar | 5 |
| amin | 1 |
| asap | 4 |
| badan | 1 |
| batas | 1 |
| bencana | 1 |
| .. | .. |
| .. | .. |
| udara | 1 |
| undur | 1 |
| wa | 1 |
| wibowo | 1 |
| widodo | 2 |
| zubaida | 1 |
| zulinto | 2 |

#### Inverse Document Frequency (idf)

Perhitungan *idf* dilakukan berdasarkan Persamaan 2.1 dengan memanfaatkan hasil dari perhitungan *df* sebelumnya. Hasil perhitungan *idf* dapat dilihat pada Tabel 4.7.

Contoh perhitungan *idf* untuk kata ‘agus’, ‘ahmad’, dan ‘ajar’:

Tabel 4.7 Hasil Perhitungan *idf*

|  |  |
| --- | --- |
| ***Term*** | ***idf*** |
| agus | 0,740363 |
| ahmad | 0,740363 |
| ajar | 0,342423 |
| amin | 1,041393 |
| asap | 0,439333 |
| badan | 1,041393 |
| batas | 1,041393 |
| bencana | 1,041393 |
| .. | .. |
| .. | .. |
| udara | 1,041393 |
| undur | 1,041393 |
| wa | 1,041393 |
| wibowo | 1,041393 |
| widodo | 0,740363 |
| zubaida | 1,041393 |
| zulinto | 0,740363 |

### Perhitungan BM25

Perhitungan BM25 akan dilakukan sesuai dengan Persamaan 2.2 dengan menentukan nilai parameter *k1* dan *b.* Nilai *k1* dan *b* yang digunakan adalah sebesar 1.2 dan *b* sebesar 0.75 (Manning et al., 2009). Untuk setiap kalimat yang akan dihitung kepiripannya dengan kalimat lain, maka kalimat tersebut dianggap sebagai *query* yang akan dihitung nilai kemiripannya dengan kalimat lainnya. Hasil perhitungan BM25 antara kalimat ditunjukan pada Tabel 4.8.

Contoh perhitungan BM25 antara kalimat 1 dan kalimat 2:

Tabel 4.8 Hasil Perhitungan BM25

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | **Dokumen** | | | | |
| **1** | **2** | **3** | **4** | **5** |
| ***Query*** | **1** |  | 2,086709 | 3,935707 | 1,357244 | 0 |
| **2** | 2,020233 |  | 2,261879 | 2,119218 | 1,131922 |
| **3** | 4,394107 | 2,674144 |  | 0,702632 | 0 |
| **4** | 1,287097 | 1,999734 | 0,501789 |  | 2,505069 |
| **5** | 0 | 1,044801 | 0 | 2,576688 |  |
| **6** | 0,260672 | 0,829409 | 1,115175 | 0,305387 | 1,768339 |
| **7** | 2,33636 | 1,271213 | 1,479159 | 1,416586 | 0 |
| **8** | 2,199502 | 0 | 2,279128 | 0,654612 | 0 |
| **9** | 0 | 0,356479 | 0 | 0,397245 | 0 |
| **10** | 0,728334 | 3,631969 | 1,866365 | 2,119218 | 3,239779 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | **Dokumen** | | | | |
| **6** | **7** | **8** | **9** | **10** |
| ***Query*** | **1** | 0,305387 | 2,456241 | 1,662959 | 0 | 0,523152 |
| **2** | 0,815057 | 1,271213 | 0 | 0,386204 | 2,932162 |
| **3** | 1,684469 | 1,986118 | 2,075131 | 0 | 2,074821 |
| **4** | 0,305387 | 1,271213 | 0,467495 | 0,386204 | 1,577885 |
| **5** | 1,818895 | 0 | 0 | 0 | 2,481174 |
| **6** |  | 1,071385 | 0,412171 | 0 | 3,133325 |
| **7** | 1,174798 |  | 3,270635 | 0,386204 | 1,966903 |
| **8** | 0,520186 | 3,527626 |  | 0 | 0,424772 |
| **9** | 0 | 0,356479 | 0 |  | 0,935273 |
| **10** | 4,157976 | 2,342597 | 0,412171 | 1,221227 |  |

### Perhitungan *PageRank*

Dalam perhitungan bobot *PageRank,* bobot awal akan diinisialisasi dengan nilai random antara nol hingga satu, lalu untuk iterasi pertama perhitungan bobot akan diperbarui menggunakan Persamaan 2.3. Setelah perhitungan bobot *PageRank* untuk semua kalimat dilakukan dalam satu iterasi, bobot *PageRank* akan diganti dengan nilai baru yang telah dihitung. Contoh inisialisasi bobot awal *PageRank* ditunjukan pada Tabel 4.9 dan hasil perhitungan bobot PageRank untuk dua iterasi ditunjukan pada Tabel 4.10.

Tabel 4.9 Inisialisasi Awal Bobot *PageRank*

|  |  |
| --- | --- |
| **Kalimat** | ***PageRank*** |
| 1 | 0,400827866 |
| 2 | 0,863170087 |
| 3 | 0,389187762 |
| 4 | 0,924094751 |
| 5 | 0,157640608 |
| 6 | 0,714980958 |
| 7 | 0,216858534 |
| 8 | 0,237221536 |
| 9 | 0,076112858 |
| 10 | 0,841401681 |

Contoh perhitungan bobot PageRank untuk kalimat 1 iterasi 1:

Tabel 4.10 Hasil Perhitungan Bobot *PageRank*

|  |  |  |  |
| --- | --- | --- | --- |
| **Kalimat** | **Inisialisasi awal** | **Iterasi 1** | **Iterasi 2** |
| 1 | 0,400827866 | 0,564107084 | 0,645031343 |
| 2 | 0,863170087 | 0,626411969 | 0,727372658 |
| 3 | 0,389187762 | 0,738073776 | 0,77323643 |
| 4 | 0,924094751 | 0,469729805 | 0,657956046 |
| 5 | 0,157640608 | 0,619960421 | 0,472058505 |
| 6 | 0,714980958 | 0,418234927 | 0,563115694 |
| 7 | 0,216858534 | 0,658518382 | 0,69574379 |
| 8 | 0,237221536 | 0,408885441 | 0,540094241 |
| 9 | 0,076112858 | 0,239364391 | 0,582118663 |
| 10 | 0,841401681 | 0,995877088 | 1,101678699 |

### Mendapatkan Ringkasan

Setelah melakukan pembaruan bobot *PageRank* sebanyak iterasi yang ditentukan, kalimat akan diurutkan berdasarkan bobot *PageRank*. Jumlah kalimat yang dipilih untuk dijadikan ringkasan akan ditentukan berdasarkan *compression rate.* Pada Tabel 4.11 ditunjukan jumlah kalimat yang diambil sebanyak 30% dari total panjang dokumen dan dibulatkan keatas. Kalimat yang terpilih kemudian diurutkan kembali berdasarkan urutan aslinya sesuai dengan dokumen awal seperti pada Tabel 4.12.

Tabel 4.11 Hasil Pengurutan Bobot *PageRank*

|  |  |
| --- | --- |
| **Kalimat** | **Iterasi 2** |
| 10 | 1,101679 |
| 3 | 0,773236 |
| 2 | 0,727373 |
| 7 | 0,695744 |
| 4 | 0,657956 |
| 1 | 0,645031 |
| 9 | 0,582119 |
| 6 | 0,563116 |
| 8 | 0,540094 |
| 5 | 0,472059 |

Tabel 4.12 Hasil Ringkasan Dokumen

|  |  |
| --- | --- |
| **Index** | **Kalimat** |
| 2 | Kepala SMP Negeri 7 Palembang, Siti Zubaida, mengatakan keputusan pemulangan ditempuh sesuai dengan instruksi Dinas Pendidikan Kota Palembang. |
| 3 | "Pagi ini kami memulangkan siswa karena melihat kabut asap yang tebal dan berdampak buruk terhadap siswa, oleh karenanya atas instruksi Kadiknas Kota Palembang melalui pesan WA Grup meminta siswa dipulangkan dan belajar di rumah masing-masing saja," jelas Siti kepada radio Elshinta. |
| 10 | "Melalui pesan digital, Kepala Dinas Pendidikan Kota Palembang menginstruksikan kegiatan belajar mengajar di tingkat paud, TK, SD dan SMP negeri dan swasta diliburkan hingga batas yang belum ditentukan," sebut Agus dalam siaran pers. |

## Perancangan evaluasi

Evaluasi dilakukan dengan membandingkan hasil peringkasan manual yang dilakuakn pakar dengan hasil peringkasan yang dihasilkan sistem. Jumlah *compression rate* yang digunakan adalah 10%, 15%, 20%, dan 25%. Rancangan evaluasi dapat dilihat pada Tabel 4.8.

Tabel 4.13 Perancangan Evaluasi Sistem Compression Rate 5%

|  |  |  |  |
| --- | --- | --- | --- |
| **Dokumen** | ***Precision*** | ***Recall*** | ***F-Measure*** |
| Dokumen 1 |  |  |  |
| .. |  |  |  |
| .. |  |  |  |
| Dokumen 10 |  |  |  |

Tabel 4.14 Perancangan Evaluasi Sistem Compression Rate 10%

|  |  |  |  |
| --- | --- | --- | --- |
| **Dokumen** | ***Precision*** | ***Recall*** | ***F-Measure*** |
| Dokumen 1 |  |  |  |
| .. |  |  |  |
| .. |  |  |  |
| Dokumen 10 |  |  |  |

Tabel 4.15 Perancangan Evaluasi Sistem Compression Rate 20%

|  |  |  |  |
| --- | --- | --- | --- |
| **Dokumen** | ***Precision*** | ***Recall*** | ***F-Measure*** |
| Dokumen 1 |  |  |  |
| .. |  |  |  |
| .. |  |  |  |
| Dokumen 10 |  |  |  |

Tabel 4.16 Perancangan Evaluasi Sistem Compression Rate 30%

|  |  |  |  |
| --- | --- | --- | --- |
| **Dokumen** | ***Precision*** | ***Recall*** | ***F-Measure*** |
| Dokumen 1 |  |  |  |
| .. |  |  |  |
| .. |  |  |  |
| Dokumen 10 |  |  |  |

# IMPLEMENTASI

## Implementasi *Preprocessing*

Pada implementasi *preprocessing* terdapat tahapan segmentasi, *cleaning, stemming, stopword removal*, dan tokenisasi. Pada tahap ini dibantu dengan *library* NLTK, *pandas, Sastrawi,* dan *re.* Tahapan segmentasi menggunakan fungsi sent\_tokenize dari library NLTK. Tahapan cleaning dibantu dengan menggunakan *library re.* Untuk melakukan penghilangan kata berimbuhan dibantu dengan fungsi factory.create\_stemmer()dari *library* Sastrawi. Sedangkan untuk tahapan tokenisasi menggunakan fungsi word\_tokenize dari *library* NLTK. Implementasi *preprocessing* ditunjukan pada Kode Sumber 5.1.

Kode Sumber 5.1 Implementasi *Preprocessing*

|  |  |
| --- | --- |
| *Preprocessing* | |
| 1 | import pandas as pd |
| 2 | import re |
| 3 | from nltk.tokenize import sent\_tokenize |
| 4 | from nltk.tokenize import word\_tokenize |
| 5 |  |
| 6 | # import StemmerFactory class |
| 7 | from Sastrawi.Stemmer.StemmerFactory import StemmerFactory |
| 8 |  |
| 9 | # create stemmer |
| 10 | factory = StemmerFactory() |
| 11 | stemmer = factory.create\_stemmer() |
| 12 |  |
| 13 | #this function does cleaning, tokenize, remove stopwords, and stemming |
| 14 | def get\_clean\_corpus(corpus, stopwords): |
| 15 |  |
| 16 | #segmentasi |
| 17 | temp = sent\_tokenize(corpus) |
| 18 | corpus = pd.DataFrame(temp, columns=['dokumen']) |
| 19 |  |
| 20 | clean\_corpus = [] |
| 21 | token = [] |
| 22 | for index, sentence in enumerate(corpus['dokumen']): |
| 23 | term = word\_tokenize(corpus['dokumen'][index]) |
| 24 |  |
| 25 | #deleting URL |
| 26 | deleted\_URL = [temp for temp in term if not re.match(r"\w+(?:(\.(\w+)\.(\w+)))|\w+(?:(\.(\w+)))", str(temp))] |
| 27 |  |
| 28 | #deleting symbol |
| 29 | deleted\_symbol = [re.sub(r"[\-\+\=\:\;\"\\\@\[\]\,\_!;.':#$%^&\*()<>?/\|}{~:]"," ",str(temp)) for temp in deleted\_URL ] |
| 30 |  |
| 31 | #stemming |
| 32 | stemmed\_sentence = stemmer.stem(" ".join(deleted\_symbol)) |
| 33 |  |
| 34 | tokens = word\_tokenize(stemmed\_sentence) |
| 35 |  |
| 36 | for i in range(len(tokens)): |
| 37 | for index, word in enumerate(tokens): |
| 38 | #delete stopwprds |
| 39 | if word in stopwords: |
| 40 | del tokens[index] |
| 41 |  |
| 42 | #delete number |
| 43 | if word.isdigit(): |
| 44 | del tokens[index] |
| 45 |  |
| 46 | clean\_corpus.append(" ".join(tokens)) |
| 47 | token.append(list(dict.fromkeys(tokens))) |
| 48 |  |
| 49 | corpus['clean\_corpus'] = clean\_corpus |
| 50 | corpus['terms'] = token |
| 51 |  |
| 52 | return corpus |

Penjelasan dari Kode Sumber 5.1:

Baris 1-7 : Melakukan *import library pandas, re,* NLTK, dan *Sastrawi*

Baris 10-11 : Inisialisasi fungsi *StremmerFactory* dan membuat *object stemmer*

Baris 14 : Deklarasi fungsi get\_clean\_corpus dengan parameter dokumen dan daftar *stopword*

Baris 17 : Segmentasi dokumen

Baris 18 : Inisialisasi *dataframe* corpus dengan argumen hasil segmentasi

Baris 22 : Melakukan perulangan untuk setiap kalimat pada hasil segmentasi

Baris 23 : Melakukan tokenisasi kalimat

Baris 26 : Melakukan penghilangan URL untuk setiap hasil tokenisasi menggunakan *regex* dengan bantuan fungsi *match* dari *library re*

Baris 29 : Melakukan penghilangan simbol pada hasil penghilangan URL

Baris 32 : Melakukan penghilangan imbuhan dengan bantuan object stemmer

Baris 34 : Melakukan tokenisasi kembali dari hasil penghilangan imbuhan

Baris 36-44 : Melakukan perulangan untuk setiap kata dari hasil tokenisasi, apabila kata muncul pada daftar *stopword* maka kata akan dihapus dan apabila kata mengandung angka maka kata juga akan dihapus

Baris 46 : Menyimpan hasil kata yang digabung menjadi dengan pemisahan spasi ke dalam *list* clean\_corpus

Baris 47 : Menyimpan hasil *list* kata ke dalam *list* token

Baris 49 : Menambahkan kolom clean\_corpus ke *dataframe* corpus

Baris 50 : Menambahkan kolom token ke *dataframe* corpus

Baris 52 : Mengembalikan hasil fungsi get\_clean\_corpus dengan *dataframe* corpus

## Implementasi *Term Weighting*

Pada implementasi *term weighting* terdapat tahapan pengambilan *term* unik, menghitung *term frequency*, menghitung *document frequency (df)*, dan *inverse document frequency (idf)*. Pada tahap ini dibantu dengan library pandas dan *math* dalam proses perhitungan *idf.* Implementasi term weighting ditunjukan pada Kode Sumber 5.2.

Kode Sumber 5.2 Implementasi *Term Weighting*

|  |  |
| --- | --- |
| *Term weighting* | |
| 1 | import math |
| 2 | def get\_term\_weighting\_score(cleaning\_result): |
| 3 |  |
| 4 | #getting all the terms |
| 5 | terms = [] |
| 6 | for index, sentence in enumerate(cleaning\_result['terms']): |
| 7 | terms += [temp for temp in sentence if temp not in terms] |
| 8 | terms.sort() |
| 9 |  |
| 10 | #getting frequency for every sentences |
| 11 | terms\_frequency = pd.DataFrame() |
| 12 | for index, term in enumerate(cleaning\_result['terms']): |
| 13 | frequency\_each\_sentence = [] |
| 14 | for i, d in enumerate(terms): |
| 15 | temp = term.count(d) |
| 16 | frequency\_each\_sentence.append(temp) |
| 17 | terms\_frequency[str(index+1)] = frequency\_each\_sentence |
| 18 |  |
| 19 | terms\_frequency['terms'] = terms |
| 20 | terms\_frequency.set\_index('terms', inplace= True) |
| 21 |  |
| 22 | #getting df for every terms |
| 23 | df\_idf = pd.DataFrame(terms\_frequency.sum(axis=1), columns=['df\_term']) |
| 24 | df\_idf['terms'] = terms |
| 25 | df\_idf.set\_index('terms', inplace= True) |
| 26 |  |
| 27 | #getting idf for every terms |
| 28 | N = len(terms\_frequency.columns) |
| 29 | terms\_idf = [] |
| 30 | for i, d in df\_idf.iterrows(): |
| 31 | idf\_score = math.log((N+1)/((df\_idf['df\_term'][i])), 10) |
| 32 | terms\_idf.append(idf\_score) |
| 33 | df\_idf['idf\_term'] = terms\_idf |
| 34 |  |
| 35 | return terms\_frequency, df\_idf |

Penjalasan dari Kode Sumber 5.2:

Baris 1 : Melakukan *import library math*

Baris 2 : Deklarasi fungsi get\_term\_weighting\_score dengan parameter hasil dari preprocessing

Baris 5-8 : Mendapatkan seluruh *term* unik dari seluruh *list* *term* pada tiap kalimat dari hasil *preprocessing* lalu mengurutkan term sesuai urutan abjad

Baris 11 : Inisialisasi *dataframe* terms\_frequency

Baris 12-17 : Untuk setiap *term* dari *term* unik akan dihitung jumlah kemunculan *term* tersebut pada setiap *list term* per kalimat dan menyimpan hasil *tf* pada *dataframe* terms\_frequency

Baris 19-20 : Menambahkan kolom term unik pada *dataframe* terms\_frequency dan menjadikan *term* unik sebagai *index* *dataframe* terms\_frequency

Baris 23-25 : Menghitung *df* dari *dataframe* terms\_frequency

Baris 28 : Menghitung jumlah kalimat yang terdapat pada dokumen

Baris 30-33 : Menghitung *idf* dari hasil *df*

Baris 35 : Mengembalikan hasil fungsi get\_clean\_corpus dengan *dataframe* terms\_frequency dan df\_idf

## Implementasi Kalimat Sebagai *Node*

Dalam mengimplementasikan *TextRank* untuk melakukan pringkasan, dokumen akan direpresentasikan sebagai graf sedangkan kalimat dalam dokumen akan direpresentasikan sebagai *node* dan nilai bobot *similarity* BM25 akan direpresentasikan sebagai *edges* (hubungan antar *node*)*.* Oleh sebab itu diperlukan pembuatan *class* kalimat. Setiap objek kalimat akan memiliki atribut dan method untuk menghitung bobot *similarity* BM25 dan bobot *PageRank.* Implementasi kalimat sebagai node ditunjukan pada Kode Sumber 5.3.

Kode Sumber 5.3 Implementasi Kalimat Sebagai *Node*

|  |  |
| --- | --- |
| *Class* Sentence | |
| 1 | class Sentence: |
| 2 | np.random.seed(0) |
| 3 | def \_\_init\_\_(self, id, full\_sentence, clean\_sentence, tokens): |
| 4 | self.list\_bm25 = {} |
| 5 | self.pagerank\_score = random.random() |
| 6 | self.id = id |
| 7 | self.full\_sentence = full\_sentence |
| 8 | self.clean\_sentence = clean\_sentence |
| 9 | self.tokens = tokens |
| 10 | self.sentence\_len = len(clean\_sentence.split()) |
| 11 | self.pagerank\_score\_new = 0 |
| 12 |  |
| 13 | def calculate\_bm25(self, raw\_frequency, idf, doc, slen\_ave): |
| 14 | k1 = 1.2 |
| 15 | b = 0.75 |
| 16 | total\_bm25 = 0 |
| 17 | for query in self.tokens: |
| 18 | tf = raw\_frequency.at[str(query),str(doc.id)] |
| 19 | idff = idf.at[str(query),'idf\_term'] |
| 20 | temp = idff \* ((k1+1) \* tf) / (k1\*( (1-b) + (b \*(doc.sentence\_len/slen\_ave)) ) + tf) |
| 21 | total\_bm25 += temp |
| 22 | self.list\_bm25[doc.id] = total\_bm25 |
| 23 |  |
| 24 | def calculate\_new\_pagerank(self, doc): |
| 25 | d=0.85 |
| 26 | sum\_InVi = 0 |
| 27 | for item in doc: |
| 28 | if self.id is not item.id: |
| 29 | Wji = self.list\_bm25[item.id] |
| 30 | total\_Wjk = sum(item.list\_bm25.values()) |
| 31 | sum\_InVi += Wji/total\_Wjk\*item.pagerank\_score |
| 32 | self.pagerank\_score\_new = (1-d)+(d\*sum\_InVi) |

Penjelasan dari Kode Sumber 5.3:

Baris 1 : Deklarasi *class* Sentence

Baris 2 : Digunakan agar angka random yang pada saat inisialisasi bobot awal *PageRank* tidak berubah ketika menjalankan program berulang kali dalam satu waktu

Baris 3 : *Constructor class* Sentence dengan parameter id kalimat, kalimat utuh, kalimat setelah *preprocessing*, dan *term* yang terdapat pada kalimat

Baris 4-11 : Inisialisasi nilai atribut ketika melakukan pembuatan objek Sentence

Baris 13 : Deklarasi fungsi calculate\_BM25 dengan parameter *tf, idf,* kalimat lain, dan panjang rata-rata dokumen

Baris 14-16 : Inisialisasi bobot k1, b, dan nilai awal total\_BM25

Baris 17-21 : Menghitung bobot BM25 setiap term dengan kalimat lain lalu menambahkan nilai bobotnya dengan variable total\_BM25

Baris 22 : Menyimpan nilai kemiripan BM25 kalimat dengan setiap kalimat lainnya

Baris 24 : Deklarasi fungsi calculate\_new\_pagerank dengan parameter seluruh dokumen

Baris 25-26 : Inisialisasi bobot d dan nilai awal sum\_InVi

Baris 27-31 : Melakukan perhitungan bobot sum\_InVi untuk seluruh kalimat pada dokumen selain objek kalimat itu sendiri

Baris 32 : Menyimpan nilai baru *PageRank* dengan nilai (1-d)+(d\*sum\_InVi)

## Implementasi Graf Dokumen

Untuk satu dokumen yang akan diringkas akan dijadikan objek *graf* dokumen yang memiliki atribut dan beberapa perhitungan untuk mendapatkan ringkasan. Proses ini dimulai dengan pempentukan objek Sentence untuk setiap kalimat yang terdapat pada dokumen. Lalu melakukan perhitungan panjang rata-rata kalimat yang akan digunakan untuk perhitungan BM25. Selanjutnya setiap kalimat akan dihitung nilai *similarity* BM25 dengan kalimat lainnya. Dalam suatu dokumen tentu saja akan terjadi kasus yang mana suatu kalimat tidak memiliki nilai *similarity* dengan kalimat manapun, untuk itu dilakukan seleksi kalimat yang akan disimpan sebagai *outlier*. Kalimat yang sama sekali tidak memiliki bobot *similarity* dengan kalimat lainnya akan dibuang dari dokumen.

Selanjutnya akan dilakukan perhitungan bobot *PageRank* sebanyak batas berhenti yang ditentukan. Pada implementasi ini kondisi berhenti dilakukan sebanyak iterasi. Setelah melakukan update bobot *PageRank,* selanjutnya dialkukan pengambilan ringkasan berdasarkan nilai *PageRank* tertinggi sebanyak *compression rate* yang ditentukan. *Compression rate* digunakan untuk mengambil seberapa banyak total kalimat yang akan dipilih sebagai ringkasan berdasarkan nilai *PageRank* tertinggi. Selanjutnya ringkasan didapatkan dari kalimat terpilih dan diurutkan kembali sesuai urutannya pada dokumen asli. Implementasi graf dokumen ditunjukan pada Kode Sumber 5.4.

Kode Sumber 5.4 Implementasi Graf Dokumen

|  |  |
| --- | --- |
| *Class* Graph | |
| 1 | class Graph: |
| 2 | def \_\_init\_\_(self, result\_doc, raw\_frequency, idf, cr): |
| 3 | self.raw\_frequency = raw\_frequency |
| 4 | self.idf = idf.drop(columns=['df\_term']) |
| 5 | self.result\_doc = result\_doc |
| 6 | self.total\_doc = len(result\_doc['dokumen']) |
| 7 | self.slen\_ave = 0 |
| 8 | self.summarize = [] |
| 9 | self.doc = [] |
| 10 | self.outlier = [] |
| 11 | self.compression\_rate = cr |
| 12 |  |
| 13 | for index, item in self.result\_doc.iterrows(): |
| 14 | self.doc.append(Sentence((index+1), item['dokumen'], item['clean\_corpus'], item['terms'])) |
| 15 |  |
| 16 | #calculate len average |
| 17 | temp\_len\_doc = 0 |
| 18 | for item in self.doc: |
| 19 | temp\_len\_doc += item.sentence\_len |
| 20 | self.slen\_ave = temp\_len\_doc/len(self.doc) |
| 21 |  |
| 22 | #calculate bm25 for each object sentence |
| 23 | for item in self.doc: |
| 24 | for item2 in self.doc: |
| 25 | if item.id is not item2.id: |
| 26 | item.calculate\_bm25(raw\_frequency= self.raw\_frequency, idf=self.idf, doc=item2, slen\_ave=self.slen\_ave) |
| 27 |  |
| 28 | ##CHECKING IF BM25 SCORE IS 0 (OUTLIER SENTENCE) |
| 29 | for index, item in enumerate(self.doc): |
| 30 | if sum(item.list\_bm25.values()) <= 0: |
| 31 | self.outlier.append(self.doc.pop(index)) |
| 32 |  |
| 33 | #calculate pagerank |
| 34 | for i in range(4): |
| 35 | for item in self.doc: |
| 36 | item.calculate\_new\_pagerank(self.doc) |
| 37 |  |
| 38 | #update pagerank score |
| 39 | for item in self.doc: |
| 40 | item.pagerank\_score = item.pagerank\_score\_new |
| 41 |  |
| 42 | #getting the summarize |
| 43 | sorted\_doc = sorted(self.doc, key=lambda x: x.pagerank\_score, reverse=True) |
| 44 | top\_pagerank = [] |
| 45 | for item in range(math.ceil(self.total\_doc\*self.compression\_rate)): |
| 46 | top\_pagerank.append(sorted\_doc[item]) |
| 47 |  |
| 48 | sorted\_sum = sorted(top\_pagerank, key=lambda x: x.id) |
| 49 |  |
| 50 | temp\_summarize = [item.full\_sentence for item in sorted\_sum] |
| 51 |  |
| 52 | self.summarize = temp\_summarize |

Penjelasan dari Kode Sumber 5.4:

Baris 1 : Deklarasi *class* Graph

Baris 2 : *Constructor class* Graph dengan parameter dokumen, *terms frequency, df&idf*, dan bobot *compression rate*

Baris 3-11 : Inisialisasi nilai atribut ketika melakukan pembuatan objek Graph

Baris 13-14 : Untuk setiap kalimat yang terdapat dalam dokumen dilakukan pembuatan objek Sentence dan disimpan sebagai atribut doc *(list of object)*

Baris 17-20 : Menghitung rata-rata panjang kalimat dan menyimpan nilai tersebut sebagai atribut slen\_ave

Baris 23-26 : Menghitung BM25 setiap objek Sentence dengan memanggil fungsi calculate\_bm25

Baris 29-31 : Melakukan seleksi kalimat ketika tidak memiliki nilai *similarity* sama sekali dengan kalimat lain. Objek Sentence tersebut akan disimpan pada atribut outlier

Baris 34-36 : Melakukan perhitungan bobot *PageRank* untuk sisa kalimat yang telah diseleksi dengan memanggil fungsi calculate\_new\_pagerank

Baris 39-40 : Melakukan update bobot *PageRank* dengan bobot baru

Baris 43 : Mengurutkan kalimat berdasarkan nilai bobot *PageRank* terbesar

Baris 44-46 : Kalimat dengan bobot *PageRank* terbessar akan dipilih sebanyak *compression rate* dan disimpan sebagai top\_pagerank

Baris 48 : Mengurutkan kalimat terpilih berdasarkan urutan idnya (sesuai urutannya pada dokumen asli) dan disimpan sebagai sorted\_sum

Baris 50-52 : Mengambil kalimat utuh dari hasil sorted\_sum dan menyimpannya sebagai atribut summarize

## Implementasi *Main* Sistem Peringkasan

Sistem peringkasan dimulai dengan melakukan import dokumen yang akan diringkas yang telah disimpan dalam bentuk *file txt*. Selanjutnya dilakukan import dokumen *stopword* tala yang akan digunakan pada *preprocessing.* Proses akan dilakutkan dengan *preprocessing* dan perhitungan *term weighting.* Percobaan pada system yang dibuat bergantung pada bobot *compression rate* yang diujikan. Percobaan pada penelitian ini menggunakan bobot *compression rate* sebanyak empat jenis, yaitu 5%, 10%, 20%, dan 30%. Main sistem yang dibuat pada penelitian ini ditunjukan pada Kode Sumber 5.5.

Kode Sumber 5.5 Main Sistem

|  |  |
| --- | --- |
| *Main* system peringkasan | |
| 1 | input\_document = open("bbc contoh beneran.txt", "r") |
| 2 | document= input\_document.readline() |
| 3 |  |
| 4 | stopword = open("stopword\_list\_tala.txt", "r") |
| 5 | stopwords = stopword.read().split("\n") |
| 6 |  |
| 7 | cleaning\_result = get\_clean\_corpus(corpus=document, stopwords=stopwords) |
| 8 |  |
| 9 | terms\_frequency, df\_idf = get\_term\_weighting\_score(cleaning\_result=cleaning\_result) |
| 10 |  |
| 11 | cr = [0.05, 0.10, 0.20, 0.30] |
| 12 | for crr in cr: |
| 13 | print("cr : ", crr) |
| 14 | percobaan = Graph(result\_doc=cleaning\_result, raw\_frequency=terms\_frequency, idf=df\_idf, cr=crr) |
| 15 | for ringkasan in percobaan.summarize: |
| 16 | print(ringkasan) |

Penjelasan dari Kode Sumber 5.5:

Baris 1-2 : Melakukan import dokumen artikel berita dalam *file txt* dan menyimpannya dengan nama variable document

Baris 4-5 : Melakukan import *stopword* tala dalam *file txt* dan menyimpannya dengan nama *variable* stopwords

Baris 7 : Memanggil fungsi get\_clean\_corpus dengan parameter stopwords dan document untuk *preprocessing* dan menyimpan hasil kembaliannya sebagai *variable* cleaning\_result

Baris 9 : Memanggil fungsi get\_term\_weighting\_score dengan parameter cleaning\_result untuk *term weighting* dan menyimpan hasil kembaliannya sebagai *variable* terms\_frequency dan df\_idf

Baris 11 : Inisialisasi bobor *compression rate* sebagai *variable* *list* cr

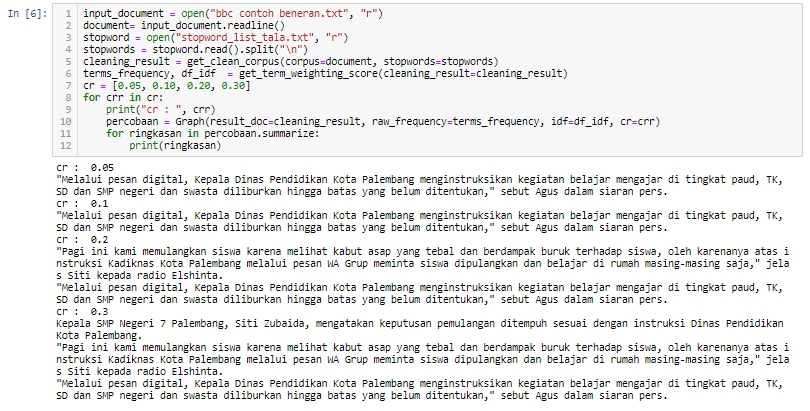
Baris 12 : Perulangan dilakukan sebanyak *item* dalam *list* cr

Baris 13 : Mencetak nilai cr

Baris 14 : Membuat objek graf dengan parameter dokumen, *terms frequency, df&idf*, dan bobot *compression rate* dan menyimpan objek tersebut dengan nama percobaan

Baris 15-16 : Mencetak hasil ringkasasn untuk setiap kalimat yang terdapat pada atribut summarize dari objek percobaan

## Hasil Implementasi Sesuai Manualisasi



# PENGUJIAN DAN ANALISIS

Tabel 6.1 Hasil Pengujian *Compression Rate* 5%

|  |  |  |  |
| --- | --- | --- | --- |
| **Dokumen** | ***Precision*** | ***Recall*** | ***F-Measure*** |
| Dokumen 1 | 1 | 1 | 1 |
| Dokumen 2 | 1 | 1 | 1 |
| Dokumen 3 | 1 | 1 | 1 |
| Dokumen 4 | 1 | 1 | 1 |
| Dokumen 5 | 1 | 1 | 1 |
| Dokumen 6 | 1 | 1 | 1 |
| Dokumen 7 | 1 | 1 | 1 |
| Dokumen 8 | 1 | 1 | 1 |
| Dokumen 9 | 1 | 1 | 1 |
| Dokumen 10 | 1 | 1 | 1 |

Tabel 6.2 Hasil Pengujian *Compression Rate* 10%

|  |  |  |  |
| --- | --- | --- | --- |
| **Dokumen** | ***Precision*** | ***Recall*** | ***F-Measure*** |
| Dokumen 1 | 1 | 1 | 1 |
| Dokumen 2 | 1 | 1 | 1 |
| Dokumen 3 | 1 | 1 | 1 |
| Dokumen 4 | 1 | 1 | 1 |
| Dokumen 5 | 1 | 1 | 1 |
| Dokumen 6 | 1 | 1 | 1 |
| Dokumen 7 | 1 | 1 | 1 |
| Dokumen 8 | 1 | 1 | 1 |
| Dokumen 9 | 1 | 1 | 1 |
| Dokumen 10 | 1 | 1 | 1 |

Tabel 6.3 Hasil Pengujian *Compression Rate* 20%

|  |  |  |  |
| --- | --- | --- | --- |
| **Dokumen** | ***Precision*** | ***Recall*** | ***F-Measure*** |
| Dokumen 1 | 1 | 1 | 1 |
| Dokumen 2 | 1 | 1 | 1 |
| Dokumen 3 | 1 | 1 | 1 |
| Dokumen 4 | 1 | 1 | 1 |
| Dokumen 5 | 1 | 1 | 1 |
| Dokumen 6 | 1 | 1 | 1 |
| Dokumen 7 | 1 | 1 | 1 |
| Dokumen 8 | 1 | 1 | 1 |
| Dokumen 9 | 1 | 1 | 1 |
| Dokumen 10 | 1 | 1 | 1 |

Tabel 6.4 Hasil Pengujian *Compression Rate* 30%

|  |  |  |  |
| --- | --- | --- | --- |
| **Dokumen** | ***Precision*** | ***Recall*** | ***F-Measure*** |
| Dokumen 1 | 1 | 1 | 1 |
| Dokumen 2 | 1 | 1 | 1 |
| Dokumen 3 | 1 | 1 | 1 |
| Dokumen 4 | 1 | 1 | 1 |
| Dokumen 5 | 1 | 1 | 1 |
| Dokumen 6 | 1 | 1 | 1 |
| Dokumen 7 | 1 | 1 | 1 |
| Dokumen 8 | 1 | 1 | 1 |
| Dokumen 9 | 1 | 1 | 1 |
| Dokumen 10 | 1 | 1 | 1 |

DAFTAR RUJUKAN

Abbasi-ghalehtaki, R., Khotanlou, H. and Esmaeilpour, M., 2016. Fuzzy evolutionary cellular learning automata model for text summarization. *Swarm and Evolutionary Computation*, [online] 30, pp.11–26. Available at: <http://dx.doi.org/10.1016/j.swevo.2016.03.004>.

Barrios, F., López, F., Argerich, L. and Wachenchauzer, R., 2016. Variations of the Similarity Function of TextRank for Automated Summarization. [online] Available at: <http://arxiv.org/abs/1602.03606>.

Fang, C., Mu, D., Deng, Z. and Wu, Z., 2017. Word-sentence co-ranking for automatic extractive text summarization. *Expert Systems with Applications*, [online] 72, pp.189–195. Available at: <http://dx.doi.org/10.1016/j.eswa.2016.12.021>.

Fhadli, M., Fauzi, M.A. and Afirianto, T., 2017. Peringkasan Literatur Ilmu Komputer Bahasa Indonesia Berbasis Fitur Statistik dan Linguistik menggunakan Metode Gaussian Naïve Bayes. 1(4), pp.307–319.

Lv, Y. and Zhai, C., 2011. Lower-bounding term frequency normalization. *International Conference on Information and Knowledge Management, Proceedings*, pp.7–16.

Munot, N. and S. Govilkar, S., 2013. Comparative Study of Text Summarization in Indian Languages. *International Journal of Computer Applications*, 75(6), pp.17–21.

Mussina, A., Aubakirov, S. and Trigo, P., 2018. Automatic Document Summarization based on Statistical Information. (Data), pp.71–76.

Niu, J., Zhao, Q., Wang, L., Chen, H., Atiquzzaman, M. and Peng, F., 2016. OnSeS: A novel online short text summarization based on BM25 and neural network. *2016 IEEE Global Communications Conference, GLOBECOM 2016 - Proceedings*, pp.1–6.

Pinandhita, R.R., 2013. Peringkas Dokumen Berbahasa Indonesia Berbasis Kata Benda Dengan BM25.

Radev, D. R., Hovy, E., & McKeown, K., 2002. Introduction to the special issue on summarization. *Computational Linguistics*, 28(4), pp.399–408.

Sankarasubramaniam, Y., Ramanathan, K. and Ghosh, S., 2014. Text summarization using Wikipedia. *Information Processing and Management*, [online] 50(3), pp.443–461. Available at: <http://dx.doi.org/10.1016/j.ipm.2014.02.001>.

Tarau, R.M. and P., 1973. TextRank: Bringing Order into Texts. *Comparative Biochemistry and Physiology -- Part B: Biochemistry and*, [online] 45(4). Available at: <http://www.aclweb.org/anthology/W04-3252>.

Yeh, J.Y., Ke, H.R., Yang, W.P. and Meng, I.H., 2005. Text summarization using a trainable summarizer and latent semantic analysis. *Information Processing and Management*, 41(1), pp.75–95.