# OTOMATISASI PERINGKASAN TEKS PADA DOKUMEN HUKUM MENGGUNAKAN METODE *LATENT SEMANTIC ANALYSIS*

## UAS METODOLOGI PENELITIAN (BAB III)

**Oleh:**

**MILLENIA RUSBANDI NIM. 1641720029**

**PROGRAM STUDI TEKNIK INFORMATIKA**

**JURUSAN TEKNOLOGI INFORMASI**

**POLITEKNIK NEGERI MALANG**

**2020**

# BAB III. METODOLOGI PENELITIAN

## 3.1 Data

Data yang diolah berupa dokumen hukum yang didapatkan dari Kantor Advokat Ahmad Riyadh Umar Balhmar, Surabaya. Data yang akan diolah tersebut memiliki format pdf.

## 3.2 Metode Pengambilan Data

Metode pengambilan data digunakan untuk mengumpulkan data-data yang dibutuhkan dalam proses penelitian. Pemgambilan data dengan cara antara lain studi literature.

### 3.4.1 Studi Literatur

Studi literatur digunakan untuk mencari informasi pustaka yang berkaitan dengan penelitian ini, yang diperoleh dari jurnal maupun penelitian sebelumnya meliputi metode – metode yang dilakukan dalam melakukan peringkasan dokumen.

### 3.4.2 Observasi.

Dalam pengambilan data, Penulis langsung terjun ke lapangan Kantor Advokat Ahmad Riyadh Umar Balhmar Kota Surabaya untuk melakukan observasi.

### 3.4.3 Wawancara

Penulis melakukan kegiatan tanya jawab dengan pihak Advokat Kota Surabaya. Metode Pengolahan Data

## 3.3 Metode Pengolahan Data

Data atau kalimat yang akan digunakan untuk pengujian metode *pre-processing* dan metode *Latent Semantic Analysis* ditunjukan pada paragraf dibawah ini.

“Emmirianto memosting status tersebut hanya dalam waktu kurang dari 1 hari dan belum berdampak bagi masyarakat. Lalu ia melakukan perbuatan tersebut atas dasar keinginan sendiri dan hal tersebut merupakan bentuk emosi dari terdakwa yang akhirnya mengalahkan akal sehatnya. Sehingga terdakwa tidak kontrol dan merasa jengkel.”

### 3.3.1 Pre-Processing

Pada tahapan ini, data tekstual akan diubah menjadi teks agar dapat diolah oleh sistem. Penelitian ini menggunakan dokumen sebagai inputan awal. Proses yang digunakan antara lain : a. Pembentukan Kalimat

Pembentukan kalimat yaitu pemecahan teks dokumen menjadi kumpulan kalimat berdasarkan delimiter.

*Table 3.1 Pembentukan Kalimat*

| No | Pembentukan Kalimat |
|---|---|
| 1 | Emmirianto memosting status tersebut hanya dalam waktu kurang dari 1 hari dan belum berdampak bagi masyarakat. |
| 2 | Lalu ia melakukan perbuatan tersebut atas dasar keinginan sendiri dan hal tersebut merupakan bentuk emosi dari terdakwa yang akhirnya mengalahkan akal sehatnya. |
| 3 | Sehingga Terdakwa tidak kontrol dan merasa jengkel. |

*b. Case Folding*

*Case folding* merupakan pengubahan huruf pada kalimat menjadi huruf kecil (*lowercase*) dan penghilangan karakter yang tidak valid seperti tanda baca.

*Table 3.2 Case Folding*

| No | *Case Folding* |
|---|---|
| 1 | emmirianto memosting status tersebut hanya dalam waktu kurang dari hari dan belum berdampak bagi masyarakat |
| 2 | lalu ia melakukan perbuatan tersebut atas dasar keinginan sendiri dan hal tersebut merupakan bentuk emosi dari terdakwa yang akhirnya mengalahkan akal sehatnya |
| 3 | sehingga terdakwa tidak kontrol dan merasa jengkel |

*c. Tokenizing*

Pada proses ini, kalimat tersebut dipecah kembali menjadi beberapa kata tunggal penyusunnya.

*Table 3.3 Tokenizing*

| *Tokenizing* | | | |
|---|---|---|---|
| emmirianto | belum | keinginan | mengalahkan |
| memosting | berdampak | sendiri | akal |
| status | bagi | dan | sehatnya |
| tersebut | masyarakat | hal | sehingga |
| hanya | lalu | tersebut | terdakwa |
| dalam | Ia | bentuk | tidak |
| waktu | melakukan | emosi | kontrol |
| kurang | perbuatan | dari | dan |
| dari | tersebut | terdakwa | merasa |
| hari | atas | yang | jengkel |
| dan | dasar | akhirnya | Dll. |

*d. Stopword Removal*

*Stopword removal* adalah proses penghilangan kata-kata yang tidak merepresentasikan isi dokumen.

*Table 3.4 Stopword Removal*

| ***Stopword Removal*** | | | |
|---|---|---|---|
| emmirianto | berdampak | dasar | emosi |
| memosting | masyarakat | bentuk | terdakwa |
| status | perbuatan | mengalahkan | Dll. |

*e. Stemming*

*Stemming* adalah proses pengembalian kata tunggal yang memiliki imbuhan menjadi kata dasar.

*Table 3.5 Stemming*

| *Stemming* | | | |
|---|---|---|---|
| emmirianto | dampak | dasar | emosi |
| posting | masyarakat | bentuk | dakwa |
| status | buat | kalah | Dll. |

### 3.3.2 Processing

#### 3.3.2.1 TF-IDF

Setelah dokumen diproses dengan cara *preprocessing, tokenizing, filtering* dan *stemming*, selanjutnya dilakukan proses pembobotan kata. Pada Metode ini pembobotan kata dalam sebuah dokumen dilakukan dengan mengalikan nilai TF dan IDF.

*Term frequency* (TF) adalah pengukuran yang paling sederhana dalam metode pembobotan. Pada metode ini, masing-masing *term* diasumsikan mempunyai proporsi kepentingan sesuai jumlah kemunculan dalam teks dokumen. *Term frequency* dapat memperbaiki nilai *recall* pada information retrieval, tetapi tidak selalu memperbaiki nilai *precision* (Tokunaga & Iwayama, 1994). Hal ini disebabkan *term* yang *frequent* cenderung muncul di banyak teks, sehingga *term* tersebut memiliki kekuatan

*Inverse document frequency* (IDF) adalah metode pembobotan term yang lebih condong (fokus) untuk memperhatikan kemunculan *term* pada keseluruhan kumpulan teks. Pada IDF, *term* yang jarang muncul pada keseluruhan koleksi teks dinilai lebih berharga. Nilai kepentingan tiap *term* diasumsikan berbanding terbalik dengan jumlah teks yang mengandung *term* tersebut (Tokunaga & Iwayama, 1994).

*Term frequncy inverse document frequency* (TF•IDF) adalah metode pembobotan yang menggabungkan metode TF dan IDF. Metode ini diusulkan oleh Salton sebagai sebuah kombinasi

metode yang dapat memberikan performansi yang lebih baik, khususnya dalam memperbaiki nilai *recall* dan *precision* (Tokunaga & Iwayama, 1994). Berikut ini merupakan perhitungannya :

$$tf.idf = tf * \log(N/df) \quad (3.1)$$

Keterangan :

*Tf* : Jumlah *term* tersebut *N* : Total dokumen

*df* : Jumlah dokumen yang mengandung suatu *term*

*Table 3.6 Contoh Perhitungan DF*

| **Kata penting** | **Frekuensi Kata pada Kalimat** | | | **df** |
|---|---|---|---|---|
| | **Kalimat 1** | **Kalimat 2** | **Kalimat 3** | |
| emmirianto | 1 | 1 | 0 | 2 |
| posting | 1 | 0 | 0 | 1 |
| status | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| dampak | 1 | 0 | 0 | 1 |
| masyarakat | 1 | 0 | 0 | 1 |
| buat | 0 | 1 | 0 | 1 |
| dasar | 0 | 1 | 0 | 1 |
| bentuk | 0 | 1 | 0 | 1 |
| emosi | 0 | 1 | 1 | 2 |
| dakwa | 0 | 1 | 0 | 1 |
| kalah | 0 | 1 | 0 | 1 |
| sehat | 0 | 1 | 0 | 1 |
| kontrol | 0 | 0 | 1 | 1 |
| jengkel | 0 | 0 | 1 | 1 |

*Table 3.7 Contoh Perhitungan TF-IDF*

| W | | |
|---|---|---|
| **tf*idf** | | |
| 1.176091 | 1.176091 | 0 |
| 1.477121 | 0 | 0 |
| 1.477121 | 0 | 0 |
| 1.477121 | 0 | 0 |
| 1.477121 | 0 | 0 |
| 1.477121 | 0 | 0 |
| 0 | 1.477121 | 0 |
| 0 | 1.477121 | 0 |
| 0 | 1.477121 | 0 |
| 0 | 1.477121 | 0 |
| 0 | 1.176091 | 1.176091 |
| 0 | 1.477121 | 0 |
| 0 | 1.477121 | 0 |
| 0 | 0 | 1.477121 |
| 0 | 0 | 1.477121 |

### 3.3.2.2 Latent Semantic Analysis

Adapun langkah-langkah LSA sebagai berikut (Mandar & Gunawan, 2017):

a. Membentuk matriks $A_{mn}$.

$$A = USV_T \quad (3.2)$$

A adalah matriks dokumen yang mewakili kalimat atau kata yang dikenal dengan matriks Amn. Dari perhitungan TF-IDF, dilakukan pembentukan matriks A dengan cara memasukkan nilai TF-IDF. Selanjutnya dilakukan perhitungan menggunakan metode *Singular Value Decomposition* (SVD). Seperti pada rumus 7.2. Akan didapat maktriks U mendiskripsikan

matriks orthogonal $m \times m$ yang dikenal dengan istilah *Left Singular Vector* . *Right Singular Vektor* (V) merupakan matriks orthogonal $n \times n$ yang diperoleh dari *eigen vector* matriks $A^TA$, sedangkan matriks diagonal S dihasilkan dari *eigen value* matriks $A^TA$ yang diakarkan.

Table 3.8 Hasil Perhitungan Matriks $A^TA$ (X)

| Kalimat 1 | Kalimat 2 | Kalimat 3 |
|---|---|---|
| 12,29262666 | 1,38319065 | 0 |
| 1,38319065 | 15,85770451 | 1,38319065 |
| 0 | 1,38319065 | 5,746965052 |

b. Perhitungan *Eigen Value*

Setelah dilakukan perhitungan $A^TA$, dilanjutkan dengan melakukan perhitungan *eigen value*. Dengan Rumus :

$$\det |X- \lambda I| \quad (3.3)$$

*Table 3.9 Hasil Perhitungan X-λI*

| Kalimat 1 | Kalimat 2 | Kalimat 3 |
|---|---|---|
| 12,2926266553501 - λ | 1,38319065 | 0 |
| 1,38319065 | 15,8577045061218 - λ | 1,38319065 |
| 0 | 1,38319065 | 5,74696505191634 - λ |

Polinomial yang didapat dari det |X-□I| yaitu $-\lambda^3 + 33{,}8973\ \lambda^2 - 352{,}885\ \lambda + 1085{,}76$.

Dengan akar akar persamaan sebagai berikut :

*Table 3.10 Hasil Perhitungan det |X-λI| (Eigen Value)*

| λ | Hasil |
|---|---|

| | |
|---|---|
| $\lambda_1$ | 16,491 |
| $\lambda_2$ | 11,85 |
| $\lambda_3$ | 5,556 |

Dari masing-masing *eigen value* dilakukan perhitungan untuk mencari *eigen vector*-nya dan dilanjutkan dengan normalisasi. Rumus normalisasi *eigen vector* :

$$\underline{x}_1^* = \frac{\underline{x}_1}{\left(\underline{x}_1^T \underline{x}_1\right)^{1/2}} = \frac{\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}}{\left( (x_1\ x_2\ x_3) \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \right)^{1/2}} = \frac{\begin{pmatrix} -x_3 \\ x_3 \\ x_3 \end{pmatrix}}{\left( (-x_3\ x_3\ x_3) \begin{pmatrix} -x_3 \\ x_3 \\ x_3 \end{pmatrix} \right)^{1/2}} \qquad (3.4)$$

Dari hasil normalisasi tersebut disatukan membentuk matriks V dan di *transpose*-kan. Seperti contoh dibawah ini.

*Table 3.11 Matriks $V^T$*

| Kalimat 1 | Kalimat 2 | Kalimat 3 |
|---|---|---|
| 0,310572622 | 0,942769607 | 0,121367684 |
| -0,950134866 | 0,304126009 | 0,068928272 |
| 0,028072398 | -0,136722469 | 0,990211547 |

c. Membentuk matriks S

Pembentukan mastriks S yaitu dengan cara mengurutkan nilai tertinggi *eigen value* kemudian diakarkan. Hasil matriks dapat dilihat pada tabel dibawah ini. *Table 3.12 Matriks S*

| Matriks S | | |
|---|---|---|
| 4,060911228 | 0 | 0 |
| 0 | 3,442382896 | 0 |
| 0 | 0 | 2,357116883 |

d. Hasil ringkasan

Hasil ringkasan ditentukan berdasarkan skor tertinggi dari perhitungan *length* pada setiap nilai matriks $V^T$ dengan menggunakan rumus :

$$S_k = \sqrt{\sum^{n}_{i=1}(V^t)^2_{k1} \, . \, S_1{}^2} \qquad (3.5)$$

Di mana $Sk$ adalah panjang vektor k pada kalimat yang dimodifikasi oleh laten vector n adalah jumlah ruang demensi baru. Hasil dari length terbesar pada setiap dokumen kalimat akan dijadikan ringkasan. Hasil dari perhitungan dapat dilihat pada tabel dibawah ini.

*Table 3.13 Hasil skor perhitungan $S_k$*

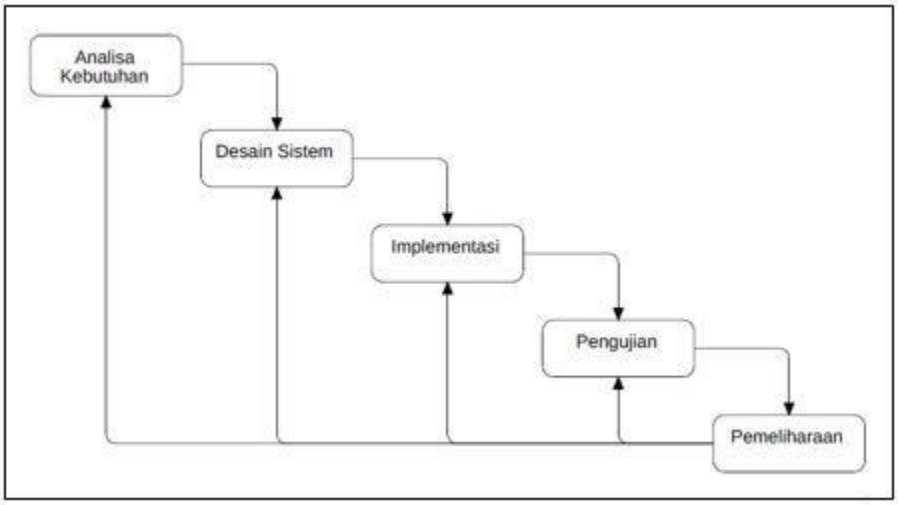
| | Hasil Skor |
|---|---|
| Kalimat 1 | 1,575819146 |
| **Kalimat 2** | **1,954901801** |
| Kalimat 3 | 1,668110957 |

Didapatkan hasil bahwa kalimat 2 akan dijadikan sebagai ringkasan dikarenakan memiliki hasil skor tertinggi.

## 3.4 Metode Pengembangan Perangkat Lunak

Pengembangan perangkat lunak dapat diartikan sebagai proses membuat suatu perangkat lunak baru untuk menggantikan perangkat lunak lama secara keseluruhan atau memperbaiki perangkat lunak yang telah ada. Agar lebih cepat dan tepat dalam mendeskripsikan solusi dan mengembangkan perangkat lunak juga hasilnya mudah dikembangkan dan dipelihara, maka pengembangan perangkat lunak memerlukan suatu metodologi khusus. Metodologi pengembangan perangkat lunak adalah suatu proses pengorganisasian kumpulan metode dan konvensi notasi yang telah didefinisikan untuk mengembangkan perangkat lunak.

Pada penelitian ini, penulis menggunakan metode *waterfall* dalam pengembangan perangkat lunak. Berikut adalah tahapan – tahapan dalam metode *waterfall* :




*Gambar **Error! No text of specified style in document.**.1 Metode Waterfall*

### 3.4.1 Analisis Kebutuhan

Kondisi, kriteria, syarat atau kemampuan yang harus dimiliki oleh perangkat lunak untuk memenuhi apa yang disyaratkan atau diinginkan pengguna.

### 3.4.2 Desain Sistem

Tahapan perancangan sistem mengalokasikan kebutuhan-kebutuhan sistem baik perangkat keras maupun perangkat lunak dengan membentuk arsitektur sistem secara keseluruhan. Perancangan perangkat lunak melibatkan identifikasi dan penggambaran abstraksi sistem dasar perangkat lunak dan hubungannya.

### 3.4.3 Implementasi

Tahap ini merupakan tahap setelah dilakukannya perancangan sistem. Perancangan yang telah dibuat akan diterjemahkan menjadi kode pemrograman dengan menggunakan bahasa php berbasis website.

### 3.4.4 Pengujian

Untuk pengujian unit digunakan *black box testing. Black Box Testing* berfokus pada spesifikasi fungsional dari perangkat lunak. Pengujian yang dilakukan masing-masing *unit*. Setelah integrasi seluruh sistem diuji untuk mengecek setiap kegagalan maupun kesalahan.

Untuk pengujian pada ringkasan dapat dilakukan dengan mencari nilai *recall, precision* dan *f-measure*. *Recall* ialah kemampuan untuk mengambil peringkat teratas yang sebagian besar relevan (benar). *Precision* adalah berapa banyak dokumen yang berhasil diambil oleh sistem, sedangkan untuk mengukur kualitas *recall* dan *precision* menggunakan *f-measure*.

$$Precision = \frac{\sum Kalimat\ Ringkasan\ Sistem \cap Kalimat\ Ringkasan\ Manual}{\sum Kalimat\ Ringkasan\ Sistem} \quad (3.6)$$

$$Recall = \frac{\sum Kalimat\ Ringkasan\ Sistem \cap Kalimat\ Ringkasan\ Manual}{\sum Kalimat\ Ringkasan\ Manual} \quad (3.7)$$

$$F - measure = 2 \text{ x } \frac{Precision\ x\ Recall}{(Precision + Recall)} \quad (3.8)$$

### 3.4.5 Pemeliharaan

Pada tahap pemeliharaan sistem, aplikasi yang telah dijalankan akan di lakukan perbaikan jika terdapat kesalahan yang tidak ditemukan pada langkah sebelumnya. Perbaikan implementasi *unit* sistem dan peningkatan jasa sistem sebagai kebutuhan baru.