Dokumentasi Program Mesin Sentiment Analysis dengan TF-IDF dan SVM

Oleh:
Muhammad Azka Adhisetama

Ucapan Terima Kasih Kepada:
Teguh Bharata Adji, PhD

Departemen Teknik Elektro dan Teknologi Informasi
Universitas Gadjah Mada

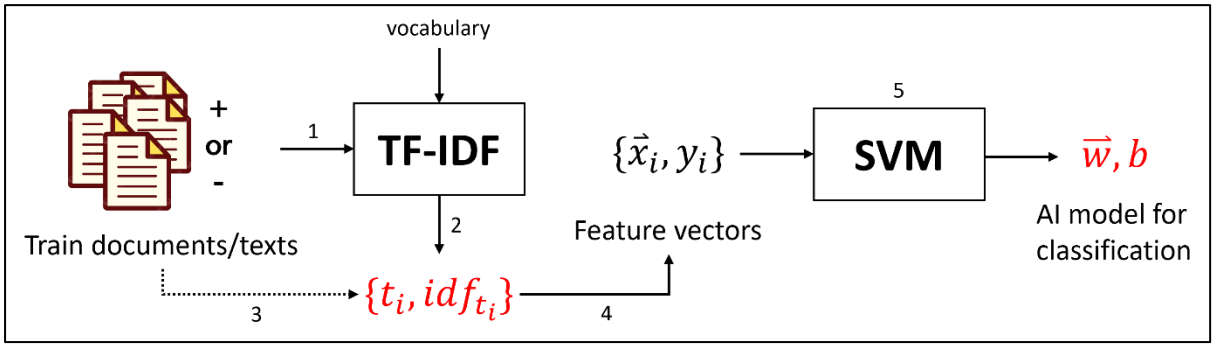
Daftar Isi

Abstrak

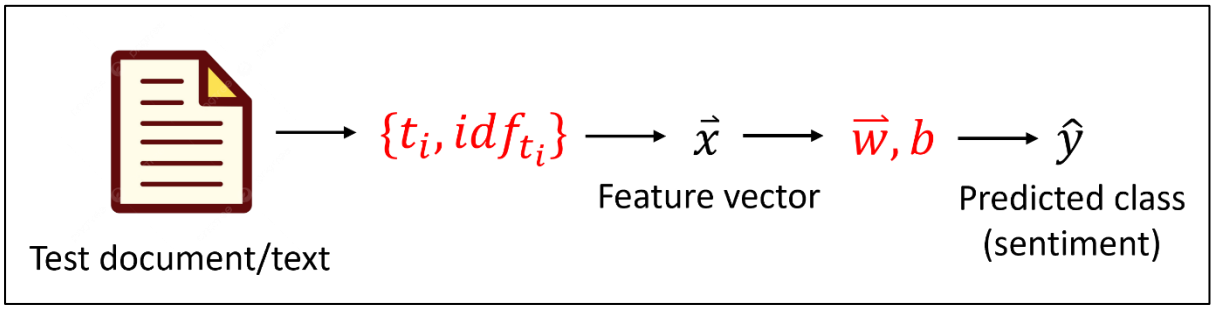
Sentiment analysis atau analisis sentimen adalah subtopik dari ranah pemrosesan bahasa alami (NLP). Analisis sentimen berkaitan dengan klasifikasi sebuah teks berdasar sentimennya, misalkan sentimen positif dan sentimen negatif. Oleh karena itu, pada dasarnya, analisis sentimen adalah sebuah *classification problem*. Ide dari program ini adalah mencoba menyelesaikan *classification problem* dari analisis sentimen dengan menggunakan *machine learning*.

Metode *machine learning* yang digunakan pada program ini adalah *Support Vector Machine* (SVM). Akan tetapi, masukan data *training* dari SVM adalah kumpulan angka atau *feature vector* yang dilabeli dengan kelas. Sementara masukan dalam analisis sentimen adalah sebuah teks yang dilabeli sentimen. Maka untuk mengubah teks masukan tersebut menjadi sebuah *feature vector*, digunakan metode *Term Frequency – Inverse Document Frequency* (TF-IDF). TF-IDF mengubah teks menjadi kumpulan *score* untuk setiap *term* dalam *vocabulary* yang didefinisikan. Sehingga setiap teks dapat direpresentasikan sebagai sebuah vektor berdimensi N dimana N adalah ukuran dari *vocabulary* dan setiap *term* merepresentasikan satu dimensi dalam vektor tersebut.

Dengan menganggap vektor yang dihasilkan oleh TF-IDF sebagai sebuah *feature vector* dan sentimen dari teks sebagai kelas yang melabeli *feature vector* tersebut, penyelesaian analisis sentimen dengan SVM dapat dilakukan. Alur proses cara kerja mesin dalam tahap *training* dan *testing* dapat dilihat pada Gambar 1 dan Gambar 2.



Gambar 1: Alur Training dari Mesin Analisis Sentimen dengan TF-IDF dan SVM



Gambar 2: Alur Prediction dari Mesin Analisis Sentimen dengan TF-IDF dan SVM

Dokumentasi Algoritma

TF-IDF

TF-IDF adalah sebuah algoritma untuk mengubah sebuah teks yang disebut *document* menjadi sebuah vektor. Vektor tersebut memiliki dimensi N yaitu ukuran dari *vocabulary*. Dengan mendefinisikan *vocabulary*, kumpulan *document* dapat digunakan untuk menghitung nilai dari *Inverse Document Frequency* (IDF) untuk setiap *term*. Dengan mendefinisikan *vocabulary* pula sebuah *document* dapat dihitung nilai dari *Term Frequency* (TF) untuk setiap *term*-nya. Perkalian setiap elemen vektor TF dari sebuah *document* dan IDF dari hasil *training* akan menghasilkan representasi vektor dari sebuah *document* tersebut.

$$\overrightarrow{TF} \odot \overrightarrow{IDF} = \vec{x} = \langle tf_{t_0} idf_{t_0}, tf_{t_1} idf_{t_1}, \dots, tf_{t_n} idf_{t_n} \rangle$$

Persamaan 1: Perhitungan Feature Vector dengan TF-IDF

Metode perhitungan TF dan IDF berbeda-beda dan tidak ada rumus pasti nya. Rumus yang digunakan bebas ditentukan dan umumnya dipilih menyesuaikan konteks dari permasalahan yang sedang diselesaikan.

Term Frequency

Perhitungan TF pada program ini menggunakan metode *count* biasa, yaitu dengan menghitung frekuensi munculnya sebuah *term* dalam sebuah *document*.

$$\overrightarrow{TF} = \left\langle tf_{t_0}, tf_{t_1}, \dots, tf_{t_n} \middle| tf_{t_i} = count(t_i) \right\rangle$$

Persamaan 2: Perhitungan TF dalam Program

Inverse Document Frequency

Perhitungan IDF pada program ini menggunakan *inverse document frequency* secara logaritmik natural. Untuk himpunan *document* $d \in D$ dan *term* dalam document $t \in d$, maka:

$$\overrightarrow{IDF} = \left\langle idf_{t_0}, idf_{t_1}, \dots, idf_{t_n} \middle| idf_{t_i} = \ln\left(\frac{|D|}{\sum_{d \in D : t \in d} count(t_i, d)}\right) \right\rangle$$

Persamaan 3: Perhitungan IDF dalam Program

SVM

Support Vector Machine adalah algoritma penyelesaian permasalahan klasifikasi dalam machine learning. Konsep dari SVM adalah mencari sebuah *hyperplane* berdimensi N-1 dimana N adalah dimensi dari *feature vector* untuk memisahkan titik-titik data dengan label yang berbeda. *Hyperplane* ini didefinisikan dengan sebuah vektor $\vec{w}$ yang normal dengan *hyperplane* tersebut dan sebuah skalar b. Sebuah titik $\vec{u}$ berada di *hyperplane* apabila terpenuhi $\vec{u} \cdot \vec{w} + b = 0$.

Definisikan y_i sebagai label dari $\vec{x}_i$ bernilai 1 apabila sentimen positif dan -1 apabila sentimen negatif. Maka *decision function* untuk prediksi y dari SVM adalah:

$$\hat{y} = sign(\vec{x} \cdot \vec{w} + b)$$

Persamaan 4: Decision Function dari SVM

Persamaan Optimisasi SVM

Optimisasi dari SVM adalah mencari nilai $\vec{w}$ dan b dimana *distance* d maksimum. Distance d adalah jarak dari $\vec{x}_+$ dan $\vec{x}_-$ yang diproyeksikan ke arah normal dari *hyperplane,* dimana $\vec{x}_+$ dan $\vec{x}_-$adalah titik terdekat untuk kelas + dan -, disebut *support vector*. Karena $\vec{w}$ normal dengan *hyperplane*, maka d adalah proyeksi $\vec{x}_+ - \vec{x}_-$ dengan unit vektor dari $\vec{w}$

$$d = (\vec{x}_+ - \vec{x}_-)\frac{\vec{w}}{\|\vec{w}\|}$$

Dengan mensubtitusikan $\vec{x}_+$ dan $\vec{x}_-$ dari Persamaan 4, didapat bahwa d hanya bergantung pada $\vec{w}$.

$$d = \frac{c}{\|\vec{w}\|}$$

Sehingga optimisasi SVM sama saja mencari panjang $\vec{w}$ terkecil.

$$\max(d) \rightarrow \min(\|\vec{w}\|) \rightarrow \min\left(\frac{1}{2}\|\vec{w}\|^2\right)$$

Menggunakan optimisasi dengan *lagrange multiplier*, kemudian mendiferensiasi parsial persamaan tersebut terhadap d dan $\vec{w}$, didapatkan persamaan optimisasi beserta *constraint* dari persamaan tersebut.

$$L = \frac{1}{2}\|\vec{w}\|^2 - \sum \alpha_i[y_i(\vec{x}_i \cdot \vec{w} + b) - 1]$$

Persamaan 5: Persamaan Lagrange Multiplier untuk Optimisasi SVM

$$\frac{\partial L}{\partial \vec{w}} = \vec{w} - \sum \alpha_i \vec{x}_i = 0$$

Persamaan 6: Turunan Parsial L terhadap vektor weight

$$\frac{\partial L}{\partial b} = \sum \alpha_i y_i = 0$$

Persamaan 7: Turunan Parsial L terhadap bias

Dengan mensubstitusikan $\vec{w}$ dari Persamaan 6 dan b dari Persamaan 4 ke Persamaan 5, didapat persamaan optimisasi dimana yang dicari adalah kumpulan *multiplier* α untuk setiap data yang ada.

$$L = \sum \alpha_i - \frac{1}{2}\sum_i \sum_j \alpha_i \alpha_j y_i y_j (\vec{x}_i \cdot \vec{x}_j)$$

Persamaan 8: Bentuk Lagrangian Dual dari Persamaan Optimisasi SVM

Kesimpulan dari berbagai persamaan tadi menunjukkan bahwa mengoptimisasi SVM dapat dilakukan dengan menemukan kombinasi nilai α maksimal sehingga L pada Persamaan 8 bernilai minimum dengan *constraint* (batasan) Persamaan 7.

Sequential Minimal Optimization

Ide dasar dari *Sequential Minimal Optimization* (SMO) adalah mengoptimisasi α satu-satu dengan menganggap α yang lain adalah konstan. Sehingga Persamaan 8 yang awalnya fungsi dengan dependen terhadap $\{\alpha_1, \alpha_2, \dots, \alpha_n\}$ hanya menjadi dependen terhadap α_i. Kemudian titik maksimum dapat dicari dengan menurunkan fungsi tersebut seperti persamaan optimisasi pada umumnya.

Permasalahan dalam ide tersebut adalah adanya batasan pada Persamaan 7. Mengganti hanya satu nilai α akan melanggar batasan tersebut. Oleh karena itu, ide dari algoritma SMO adalah memilih bukan satu α, namun sepasang $\{\alpha_r, \alpha_s\}$ dimana α_s dianggap sebagai fungsi dari α_r agar Persamaan 7 tetap berlaku.

$$\sum \alpha_i y_i = 0 \rightarrow \alpha_s = -\frac{\alpha_r y_r + \sum_{i \neq \{r,s\}} \alpha_i y_i}{y_s} = -\frac{\alpha_r y_r + \gamma_{r,s}}{y_s}$$

Persamaan 9: a_s Sebagai Fungsi dari a_r Sehingga Constraint Terpenuhi

Substitusikan Persamaan 9 ke Persamaan 8 dengan menganggap semua α selain α_r dan α_s sebagai konstan akan menghasilkan fungsi kuadrat dengan variabel independen α_r.

$$\begin{aligned} L = f(\alpha_r) = {} & \frac{1}{2}(\vec{x}_r \cdot \vec{x}_r + \vec{x}_s \cdot \vec{x}_s - 2(\vec{x}_r \cdot \vec{x}_s)){\alpha_r}^2 \\ & + \left(\gamma_{r,s} y_r (\vec{x}_r \cdot \vec{x}_s - \vec{x}_s \cdot \vec{x}_s) + y_r \left((\vec{x}_r - \vec{x}_s) \cdot \sum_{i \neq \{r,s\}} \alpha_i y_i \right) + y_r y_s \right. \\ & \left. - 1 \right) \alpha_r + \gamma_{r,s} \left(\frac{1}{2} \gamma_{r,s} \vec{x}_s \cdot \vec{x}_s + \vec{x}_s \cdot \sum_{i \neq \{r,s\}} \alpha_i y_i - y_s \right) \\ & = a{\alpha_r}^2 + b\alpha_r + c \end{aligned}$$

Persamaan 10: Fungsi Optimisasi dari a_r Sebagai Persamaan Kuadrat

Untuk mencari titik maksimum dari α_r, cari titik kritis pada $f(\alpha_r)$ dengan persamaan diferensial terhadap α_r.

$$\begin{aligned} & \frac{df(\alpha_r)}{d\alpha_r} = 0 \rightarrow \alpha_{r\,max} = \frac{-b}{2a} \\ & = \frac{-\left(\gamma_{r,s} y_r (\vec{x}_r \cdot \vec{x}_s - \vec{x}_s \cdot \vec{x}_s) + y_r \left((\vec{x}_r - \vec{x}_s) \cdot \sum_{i \neq \{r,s\}} \alpha_i y_i \right) + y_r y_s - 1 \right)}{(\vec{x}_r \cdot \vec{x}_r + \vec{x}_s \cdot \vec{x}_s - 2(\vec{x}_r \cdot \vec{x}_s))} \end{aligned}$$

Persamaan 11: Titik Maksimum a_r

Agar nilai α terkontrol, definisikan *hyperparameter* C yaitu *tolerance* sebagai batas maksimal dari α.

$$0 \leq \alpha_r, \alpha_s \leq C$$

Nilai C yang terlalu kecil menyebabkan optimisasi tidak konvergen, sementara nilai C yang terlalu besar membuat *overfitting* dan model menjadi rentan terhadap *outlier*. Nilai *hyperparameter* C yang digunakan pada program ini adalah 10.

Menggabungkan batas tersebut dengan batasan pada Persamaan 7 menghasilkan batasan atas dan bawah untuk α_r yaitu:

1. Untuk $y_r = y_s \rightarrow y_r y_s = 1$ ($\vec{x}_r$ dan $\vec{x}_s$ memiliki label yang sama)
$$\max\left(0, -y_r\left(y_s C - \gamma_{r,s}\right)\right) \leq \alpha_r \leq \min\left(C, y_r \gamma_{r,s}\right)$$
2. Untuk $y_r \neq y_s \rightarrow y_r y_s = -1$ ($\vec{x}_r$ dan $\vec{x}_s$ memiliki label yang berbeda)
$$\max\left(0, y_r \gamma_{r,s}\right) \leq \alpha_r \leq \min\left(C, -y_r\left(y_s C - \gamma_{r,s}\right)\right)$$

Persamaan 12: Batas Atas dan Bawah a_r

Algoritma Iterasi dalam Sequential Minimal Optimization

Inisialisasi

1. Set $\{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ menjadi angka *placeholder* yang memenuhi Persamaan 7. Dalam program ini, nilai awal dari semua α diberi nilai nol
2. Set nilai dari *hyperparameter* C. Dalam program ini, C diberi nilai 10

Iterasi

1. Pilih indeks $\{r, s\}$ secara acak
2. Hitung batasan untuk α_r dengan Persamaan 12
3. Hitung titik maksimum optimal α_r dengan Persamaan 11
4. Batasi titik tersebut dengan batasan yang sudah di hitung
5. Update nilai α_r
6. Update nilai α_s dengan Persamaan 9