BAB 2

LANDASAN TEORI

2.1 Klasifikasi Dokumen

Klasifikasi dokumen adalah sebuah metode perolehan informasi yang untuk menentukan atau mengkategorikan suatu dokumen ke dalam satu atau lebih kelompok. Dokumen yang telah diketahui kategori sebelumnya secara otomatis dikelompokkan berdasarkan isi dokumen tersebut [**5**]. Klasifikasi dokumen bertujuan untuk mengelompokkan dokumen yang tidak terstruktur ke dalam kelompok-kelompok yang menggambarkan isi dari dokumen.

2.2 Preprocessing

Preprocessing adalah sebuah tahapan yang mengubah teks menjadi data yang dapat diolah pada tahap berikutnya. Tahapan preprocessing yang dilakukan pada penelitian ini meliputi *filtering*, *case folding*, *tokenizing*, dan *stopword removal*.

2.2.1 Filtering

Filtering mengacu pada proses memutuskan istilah mana yang harus digunakan untuk merepresentasikan dokumen sehingga dapat digunakan untuk menggambarkan isi dokumen dan membedakan dokumen dengan dokumen lain yang ada pada koleksi [**6**]. Dengan kata lain *filtering* merupakan proses dimana teks selain karakter "a" sampai "z" akan dihilangkan dan hanya menerima spasi.

2.2.2 Case Folding

Case folding adalah tahapan pemrosesan teks dimana semua teks diubah ke dalam case yang sama. Pada penelitian ini semua huruf dalam teks dokumen diseragamkan menjadi huruf kecil [**6**].

2.2.3 Tokenizing

Tokenizing adalah proses pemotongan *string* masukan berdasarkan tiap kata yang menyusunnya menjadi bagian-bagian atau token-token tertentu.

Pada umumnnya setiap kata terindentifikasi atau terpisahkan dengan kata yang lain oleh karakter spasi, sehingga proses *tokenizing* mengandalkan karakter spasi pada dokumen untuk melakukan pemisahan kata [**6**].

2.2.4 Stopword Removal

Proses pembuangan stopword dimaksudkan untuk mengetahui suatu kata masuk ke dalam stopword atau tidak. Stoplist berisi kumpulan kata yang 'tidak relevan', tetapi seringkali muncul dalam sebuah dokumen. Dengan kata lain, stoplist berisi sekumpulan stopwords. Stopwords removal adalah proses menghilangkan kata yang 'tidak relevan' dari sebuah dokumen teks dengan cara membandingkannya dengan stoplist yang ada [**6**].

2.2.5 Term Frequency – Inverse Document (TF-IDF)

Pembobotan *TF-IDF*, *Term Frequency Factor* adalah faktor yang menentukan bobot *term* pada suatu dokumen berdasarkan jumlah kemunculannya dalam dokumen tersebut. Sedangkan, *IDF (Inverse Document Frequency)*, yakni pengurangan dominasi *term* yang sering muncul diberbagai dokumen [**7**].

Pembobotan diperoleh dari frekuensi jumlah kemunculan sebuah kata yang terdapat di dalam sebuah dokumen, *term frequency (tf).* Sebuah kata atau jumlah kemunculan *term* di dalam koleksi dokumen, *inverse document frequency (idf).*

Bobot suatu istilah semakin besar jika istilah tersebut sering muncul dalam suatu dokumen dan semakin kecil jika istilah tersebut muncul dalam banyak dokumen.

Nilai *idf* sebuah *term* (kata) dapat dihitung menggunakan persamaan (2.1) berikut :

$$IDFt = \log(N/df) \quad (2.1)$$

Untuk menghitung bobot (W) masing-masing dokumen terhadap setiap term (kata) dapat menggunakan persamaan berikut:

$$W\,d\,t \;=\; TF * IDFt \quad (2.2)$$

Dimana:

W = bobot dokumen ke-d terhadap kata ke-t

d = dokumen ke-d

t = kata ke-t

tf = banyaknya kata yang dicari pada sebuah dokumen

N = total dokumen

df = banyak dokumen yang mengandung tiap kata

2.2.6 Normalisasi

Normalisasi dilakukan terhadap vektor fitur dokumen untuk menghilangkan pengaruh anggapan bahwa dokumen panjang lebih relevan dibandingkan dokumen pendek. Dengan normalisasi ini dapat membantu menormalkan batas nilai dengan melakukan standarisasi nilai ke dalam interval 0 sampai dengan 1. Sehingga dapat dituliskan sebagai berikut:

$$w(word_i) = \frac{w(word_i)}{\sqrt{w^2(word_1) + w^2(word_2) + \cdots + w^2(word_n)}} \quad (2.3)$$

2.3 K-Means

K-Means merupakan metode data *clustering* yang digolongkan sebagai metode pengklasifikasian yang bersifat *unsupervised*. Pengkategorian metode - metode pengklasifikasian data antara *supervised* dan *unsupervised classification* didasarkan pada adanya dataset yang data itemnya sudah sejak awal mempunyai label kelas dan dataset yang data itemnya tidak mempunyai label kelas. Untuk data yang sudah mempunyai label kelas, metode pengklasifikasian yang digunakan merupakan metode *supervised classification* dan untuk data yang belum mempunyai label kelas, metode pengklasifikasian yang digunakan adalah metode *unsupervised classification*.

Penentuan jumlah *cluster* untuk dataset yang dianalisa umumnya dilakukan secara *supervised* atau ditentukan dari awal oleh pengguna, walaupun dalam penerapannya ada beberapa metode yang sering dipasangkan dengan metode *K-Means* [**8**]. Adapun algoritma dari *K-means* sebagai berikut:

1. Menentukan jumlah *cluster*
2. Menentukan nilai *centroid*

   Dalam menentukan nilai centroid untuk awal iterasi, nilai awal centroid dilakukan secara acak. Sedangkan jika menentukan nilai centroid yang merupakan tahap dari iterasi, maka digunakan rumus sebagai berikut.

$$\bar{v}_{ij} = \frac{1}{N_i} \sum_{k=0}^{N_i} x_{kj} \ , \qquad (2.4)$$

dimana:

$\bar{v}_{ij}$ = centroid/ rata-rata *cluster* ke-*i* untuk variabel ke-*j*

N_i = jumlah data yang menjadi anggota *cluster* ke-*i*

i,*k* = indeks dari *cluster*

j = indeks dari variabel

x_{kj} = nilai data ke-*k* yang ada di dalam *cluster* tersebut untuk variabel ke-*j*

3. Menghitung jarak jarak antara data dengan pusat *cluster*

   Untuk menghitung jarak tersebut dapat menggunakan *Euclidean Distance*. *Euclidean* sering digunakan karena penghitungan jarak dalam *distance space* ini merupakan jarak terpendek yang bisa didapatkan antara dua titik yang diperhitungkan. Berikut persamaan dengan *Euclidean Distance*.

   $$D_e = \sqrt{(x1 - y1)^2 + (x2 - y2)^2 + \ ... \ + (xi - yi)^2} \qquad (2.5)$$

   De = *euclidean distance*.

   i = jumlah data

   x = bobot dokumen.

   y = pusat *cluster*.

4. Pengelompokkan Data

   Untuk menentukan anggota *cluster* adalah dengan memperhitungkan jarak minimum objek. Nilai yang diperoleh dalam keanggotaan data pada *distance* matriks adalah 0 atau 1, dimana nilai 1 untuk data yang dialokasikan ke *cluster* dan nilai 0 untuk data yang dialokasikan ke *cluster* yang lain.

5. Kembali ke tahap 2, lakukan perulangan hingga nilai *centroid* yang dihasilkan tetap dan anggota *cluster* tidak berpindah ke *cluster* lain.

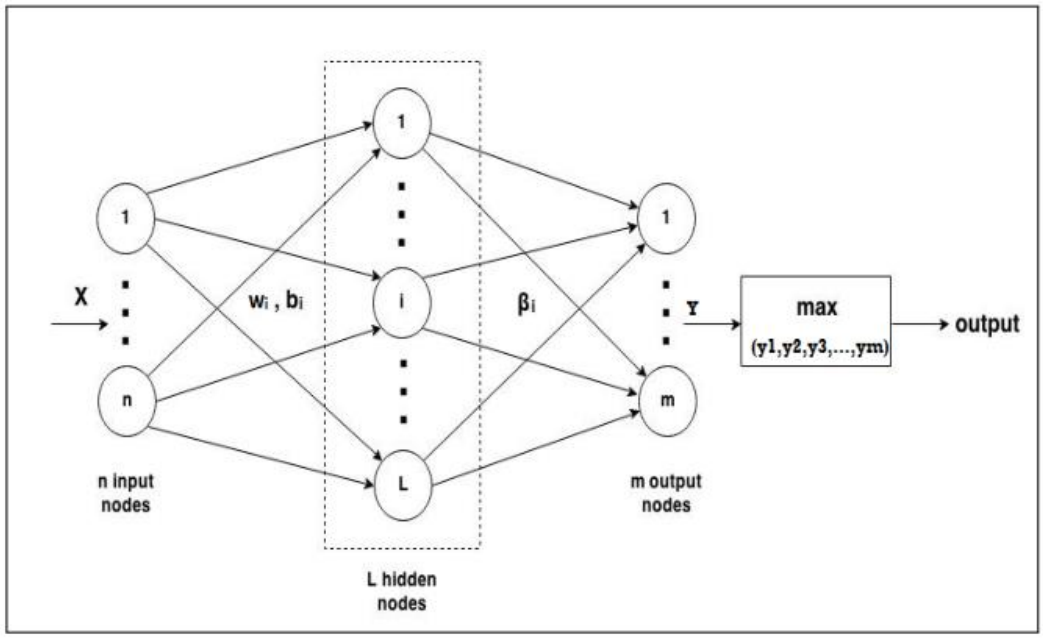
2.4 Extreme Learning Machines (ELM)

Extreme learning machine merupakan jaringan saraf tiruan *feedforward* dengan satu *hidden layer* atau biasa disebut dengan istilah *single hidden layer feedforward neural network (SLFNs)* [**9**].

Metode *ELM* dibuat untuk mengatasi kelemahan-kelemahan dari jaringan saraf tiruan *feedforward* terutama dalam hal *learning speed*. Algoritma *ELM* tidak melatih bobot *input* ataupun bias, *ELM* melatih untuk memperoleh bobot keluarannya dengan menggunakan *norm-least-squares solution* dan *moore-penrose inverse* pada sistem linier secara umum.

Dengan menemukan *node* yang memberikan nilai *output* maksimal, dan parameter-parameter seperti *input weight* dan bias dipilh secara *random*, sehingga *ELM* memiliki *learning speed* yang cepat dan mampu menghasilkan *good generalization performance.*

2.4.1 Arsitektur Algoritma ELM



Gambar 2. 1 Arsitektur ELM

Untuk sampel N (x_i, y_i), di mana $x_i = [x_{i1}, x_{i2}, ..., x_{in}]^T \in R^n$ dan $y_i = [y_{i1}, y_{i2}, ..., yi_m]^T \in R^m$, sedemikian rupa sehingga $(x_i, y_i) \in R^n \times R^m$ dimana (i = 1,2, ...,

N), dengan L sebagai *hidden nodes*, dan fungsi aktivasi g (x). Fungsi keluaran ELM untuk input x yang diberikan adalah [**4**]:

$$g_L(x_j) = \sum_{i=1}^{L} \beta_i g(w_i \cdot x_j + b_i) = y_j, j = 1, ..., N \tag{2.6}$$

Dimana, (w_i, b_i), i = 1, ..., L dengan parameter secara acak dimana $w_i = [w_{i1}, w_{i2} ... w_{in}]$ adalah bobot vektor yang menghubungkan semua node *input* ke simpul tersembunyi. b_i adalah bias dari simpul tersembunyi. $\beta = [\beta_1, ..., \beta_L]^T$ adalah beratnya vektor antara node tersembunyi dan node *output*. g (x) adalah vektor output yang memetakan ruang input n-dimension untuk fitur ruang dimensi L. Fungsi aktivasi g(x) yang digunakan adalah softsign, fungsi ini memberikan batasan keluaran antara (0,1) rumusnya sebagai berikut:

$$g(w_i x_j + b_i) = \frac{wi.xj+bi}{1+|wi.xj+bi|} \tag{2.7}$$

Format dari Persamaan (2.6) dapat dituliskan sebagai berikut:

$$H\beta = Y \tag{2.8}$$

Dimana,

$$\mathbf{H} = \begin{bmatrix} g(w_1 \cdot x_1 + b_1) & \cdots & g(w_L \cdot x_1 + b_L) \\ g(w_1 \cdot x_2 + b_1) & \cdots & g(w_L \cdot x_2 + b_L) \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ \cdot & \cdots & \cdot \\ g(w_1 \cdot x_N + b_1) & \cdots & g(w_L \cdot x_N + b_L) \end{bmatrix}_{N \times L} \tag{2.9}$$

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_{11} & \cdots & \beta_{1m} \\ \beta_{21} & \cdots & \beta_{2m} \\ & \cdot & \\ & \cdot & \\ & \cdot & \\ \beta_{L1} & \cdots & \beta_{Lm} \end{bmatrix}_{L \times m} \quad \boldsymbol{Y} = \begin{bmatrix} y_{11} & \cdots & y_{1m} \\ y_{21} & \cdots & y_{2m} \\ & \cdot & \\ & \cdot & \\ & \cdot & \\ y_{N1} & \cdots & y_{Nm} \end{bmatrix}_{N \times m} \tag{2.10}$$

Pelatihan kesalahan terkecil (konsep serupa di SVM) dan norma bobot keluaran terkecil dapat dicapai dengan ELM dan dapat direpresentasikan sebagai berikut:

$$minimize: \ \| \mathbf{H}\boldsymbol{\beta} - \mathbf{Y} \|^2 \ and \ \| \boldsymbol{\beta} \| \tag{2.11}$$

Solusi kuadrat terkecil minimal dari sistem linier di atas diberikan oleh:

$$\boldsymbol{\beta} = \boldsymbol{H}^{+}\boldsymbol{Y} \tag{2.12}$$

dimana H^+ adalah *Moore-Penrose invers* matriks *H*.

H (disebut ruang fitur ELM, juga dikenal sebagai *hidden layer output matrix*) merupakan matriks bobot penghubung *input layer* dan *hidden layer* maka matriks *H* mempunyai ukuran *N* x *L*. Penentuan nilai elemen-elemen matriks tersebut dilakukan secara random. Kemudian setiap nilai *input* tersebut diproses pada *hidden layer* menggunakan fungsi aktivasi dan nilai tersebut dihimpun dalam sebuah matriks *H* dengan ordo *N* x *L*.

2.5 UML

Unified Modeling Language (UML) adalah himpunan struktur dan teknik untuk pemodelan desain program berorientasi objek *(OOP)* serta aplikasinya. *UML* adalah metodologi untuk mengembangkan sistem *OOP* dan sekelompok perangkat *tool* untuk mendukung pengembangan sistem tersebut. *UML* merupakan dasar bagi perangkat (*tool*) desain berorientasi objek dari *IBM*.

UML adalah suatu bahasa yang digunakan untuk menentukan, memvisualisasikan, membangun, dan mendokumentasikan suatu sistem

informasi. *UML* dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek. Namun demikian *UML* dapat digunakan untuk memahami dan mendokumentasikan setiap sistem informasi. Ini merupakan standar terbuka yang menjadikannya sebagai bahasa pemodelan yang umum dalam industri perangkat lunak dan pengembangan sistem [**10**].

UML menyediakan beberapa macam diagram untuk memodelkan aplikasi berorientasi objek. Adapun beberapa diagram yang sering digunakan adalah *Use Case, Activity Diagram, Sequence Diagram,* serta *Class Diagram*.

2.5.1 Use Case Diagram

Use case diagram digunakan untuk memodelkan proses bisnis berdasarkan perspektif pengguna sistem. *Use case diagram* terdiri atas diagram untuk *use case* dan *actor*. *Actor* merepresentasikan orang yang akan mengoperasikan atau orang yang berinteraksi dengan sistem aplikasi [**10**].

Use case merepresentasikan operasi-operasi yang dilakukan oleh *actor*. *Use case* digambarkan berbentuk elips dengan nama operasi dituliskan di dalamnya. *Actor* yang melakukan operasi dihubungkan dengan garis lurus ke *use case*.

2.5.2 Activity Diagram

Activity diagram adalah representasi grafis dari seluruh tahapan alur kerja. Diagram ini mengandung aktivitas, pilihan tindakan, perulangan dan hasil dari aktivitas tersebut. Pada pemodelan *UML*, diagram ini dapat digunakan untuk menjelaskan proses bisnis dan alur kerja operasional secara langkah demi langkah dari komponen suatu sistem [**10**].

Activity diagram pada dasarnya menggambarkan skenario secara grafis. Serta *activity diagram* cukup serupa dengan diagram alir (*flow chart*), yang membedakan mungkin hanya *swimlane* yang menunjukkan suatu *state* berada pada objek/kelas tertentu. Keunggulan dari *activity diagram* adalah bahwa diagram tersebut lebih mudah dipahami dibanding skenario. Selain itu, dengan menggunakan *activity diagram*, dapat melihat dibagian manakah sistem dari suatu skenario akan berjalan.

2.5.3 Sequence Diagram

Diagram *sequence* merupakan salah satu diagram interaksi (*interaction diagram*) yang menjelaskan bagaimana sistem bekerja. Diagram ini diatur berdasarkan waktu. Obyek-obyek yang berkaitan dengan proses berjalannya operasi diurutkan dari kiri ke kanan berdasarkan waktu terjadinya dalam pesan yang terurut [**10**].

2.5.4 Class Diagram

Class diagram menggambarkan komponen dasar suatu perangkat lunak berorientasi objek. *Class* merepresentasikan suatu *Class* individu atau pun beberapa *Class* yang terkait satu sama lain. Dengan demikian diagram kelas menunjukkan arsitektur sebuah sistem [**10**].

2.6 Akurasi

Untuk menghitung akurasi dari ELM dilakukan perhitungan dengan rumus [**11**] :

$$\text{Akurasi} = \frac{Jumlah\ klasifikasi\ benar}{total\ data\ yang\ diuji} \text{ x } 100\% \qquad (2.13)$$