BAB 2
LANDASAN TEORI

2.1 Sistem Tanya Jawab

Salah satu bentuk interaksi antara manusia dan komputer, yaitu dapat melalui sistem tanya jawab (*question answering system*). Sistem tanya jawab merupakan sistem yang menerima pertanyaan dan memberikan jawaban singkat dengan menggunakan bahasa alami. Sistem tanya jawab dapat membahas permasalahan yang luas, mulai dari kesehatan, olahraga, politik dan sebagainya [8]. Contoh sistem tanya jawab yang populer saat ini, yaitu adalah *chatbot*. Beberapa *chatbot* yang terkenal di antaranya Duolingo, Gymbot, Dinner Ideas, Luka, Cleverbot, dan lain-lain.

Sistem tanya jawab dikembangkan untuk membantu pengguna dalam mencari informasi. Respon atau jawaban yang diberikan oleh sistem tanya jawab dapat bergantung pada kata-kata spesifik dari pertanyaan yang diterima dari pengguna berdasarkan aturan yang telah ditentukan, atau dapat juga dihasilkan dari pelatihan yang dilakukan dengan kecerdasan buatan sehingga sistem tanya jawab dapat melakukan penalaran.

2.2 Kecerdasan Buatan

Kecerdasan buatan atau *Artificial Intelligence* merupakan suatu ilmu yang mempelajari bagaimana membuat komputer melakukan sesuatu pada suatu kejadian atau peristiwa yang mana orang melakukannya dengan baik. Kecerdasan buatan merupakan proses di mana peralatan mekanik dapat melaksanakan kejadian-kejadian dengan menggunakan pemikiran atau kecerdasan seperti manusia [9].

Aplikasi-aplikasi yang dibuat menggunakan kecerdasan buatan dapat berpikir layaknya manusia seperti menentukan keputusan, menarik kesimpulan, rekomendasi keputusan dan otomasi sebuah sistem.

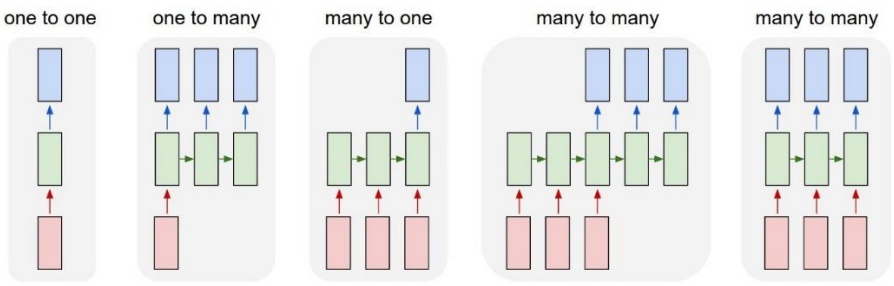
2.3 *Machine Learning*

Machine learning merupakan salah satu bidang kajian dari kecerdasan buatan yang dapat belajar memprediksi sesuatu dari sebuah contoh. Daripada sebuah komputer diberikan sejumlah peraturan (rules) untuk dipecahkan, *machine learning* mampu memberikan model untuk dievaluasi dan diberikan perintah untuk mengubah model ketika terjadi kesalahan [10].

Machine Learning adalah metode analisis data yang dapat membangun model analitis secara otomatis. *Machine Learning* adalah bagian dari kecerdasan buatan yang didasarkan pada gagasan bahwa sistem dapat belajar dari data, mengidentifikasi pola dan membuat keputusan dengan intervensi manusia minimal.

2.4 *Recurrent Neural Network* (RNN)

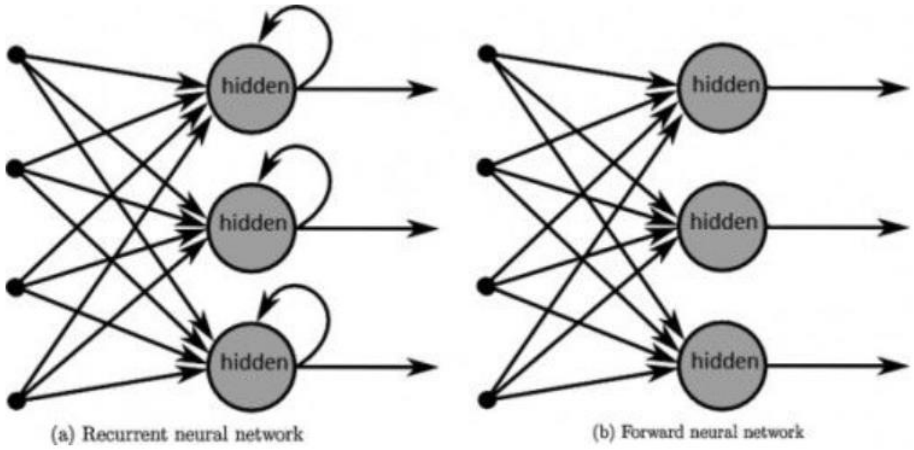
Jaringan saraf berulang atau *recurrent neural network* (RNN) adalah jenis arsitektur jaringan saraf tiruan yang pemrosesannya dipanggil berulang-ulang untuk memroses masukan yang biasanya adalah data sekuensial. RNN masuk dalam kategori *deep learning* karena data diproses melalui banyak lapis (*layer*). RNN telah mengalami kemajuan yang pesat dan telah merevolusi bidang-bidang seperti pemrosesan bahasa alami, pengenalan suara, sintesa musik, pemrosesan data finansial seri waktu, analisis deret DNA, analisis video, dan sebagainya [11]. RNN memiliki beberapa arsitektur sesuai dengan kebutuhannya, yang dapat dilihat pada Gambar 2.1 [12].



Gambar 2.1 Macam-macam Arsitektur RNN

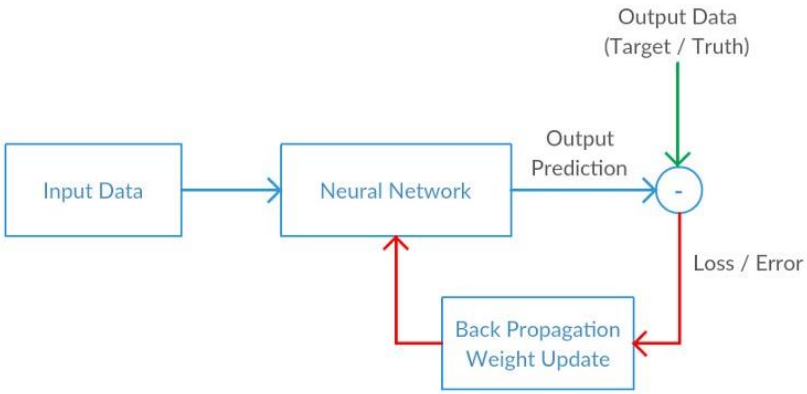
Pada Gambar 2.1, vektor masukan diwakili oleh kotak berwarna merah, vektor keluaran diwakili oleh kotak berwarna biru dan kotak berwarna hijau merupakan vektor *hidden state* RNN. Arsitektur RNN satu ke satu terdiri dari masukan berukuran tetap dan keluaran berukuran tetap, biasanya digunakan untuk klasifikasi gambar. Arsitektur RNN satu ke banyak dapat dimanfaatkan misalnya untuk *image captioning*. Arsitektur RNN yang selanjutnya, yaitu banyak ke satu, dapat digunakan misalnya untuk analisis sentimen dari sebuah pernyataan atau kalimat, lalu dapat juga digunakan untuk melakukan klasifikasi kalimat (*sequence*) ke dalam beberapa kategori. Arsitektur RNN banyak ke banyak dapat dimanfaatkan untuk *Machine Translation* di mana masukannya dapat berupa kalimat berbahasa Inggris yang kemudian dihasilkan keluaran dengan kalimat berbahasa Indonesia [12].

RNN berbeda dengan Feedforward Neural Network yang semua masukan dan keluarannya diasumsikan tidak bergantung satu sama lain. RNN juga memiliki bobot-bobot yang sama untuk setiap proses yang dilakukan. RNN disebut berulang karena melakukan tugas yang sama untuk setiap elemen urutan, dengan keluaran tergantung pada perhitungan sebelumnya. Struktur dari RNN mirip dengan *Feedforward Neural Network*, perbedaannya hanya *hidden state* pada RNN, di mana setiap waktu bergantung pada waktu sebelumnya (siklus). Gambarannya dapat dilihat pada Gambar 2.2 [13].



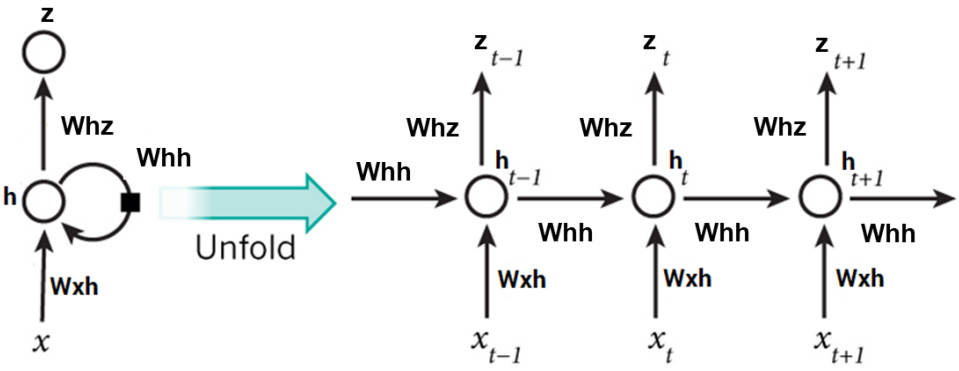
Gambar 2.2 Perbedaan RNN dan *Feedforward* NN

Proses pelatihan pada RNN sama seperti proses pelatihan pada *neural network* pada umumnya. Terdapat tiga langkah utama proses pelatihan pada RNN. Pertama, melakukan proses forward pass dan membuat prediksi. Pada proses ini, dilakukan perhitungan untuk setiap *hidden state* (h_t) berdasarkan setiap masukan (x_t) dan bobot yang telah ditentukan. Setelah ditemukan nilai dari *hidden state*, maka selanjutkan dilakukan perhitungan untuk keluaran atau hasil prediksinya (z_t). Langkah kedua, membandingkan hasil prediksi (z_t) dengan nilai keluaran yang sebenarnya atau disebut juga target, menggunakan *Loss Function*. *Loss Function* menghasilkan nilai kesalahan yang dapat menunjukkan apakah hasil prediksi sesuai target atau bahkan jauh dari target sehingga dapat memberi kesimpulan seberapa baik atau buruk kinerja RNN tersebut. Langkah yang terakhir, dari nilai kesalahan yang dihasilkan oleh *Loss Function* selanjutnya dilakukan proses *Backpropagation Through Time* (BPTT) untuk menghitung gradien untuk setiap langkah waktu dalam jaringan. Proses BPTT dilakukan untuk mencari bobot-bobot dan juga bias yang lebih baik dari proses sebelumnya. Setelah proses BPTT selesai dilakukan, maka dilakukan pembaruan bobot dan juga bias dengan metode *Stochastic Gradient Descent* (SGD). Gambaran langkah-langkah pelatihan *Neural Network* (termasuk RNN), dapat dilihat pada Gambar 2.3.



Gambar 2.3 Langkah-langkah Pelatihan *Neural Network*

Struktur RNN dengan variabel-variabel yang diperlukan untuk melakukan proses perhitungan, dapat dilihat pada Gambar 2.4.



Gambar 2.4 Struktur RNN

Rumus-rumus untuk perhitungan yang terjadi dalam proses pelatihan RNN yang sesuai dengan struktur RNN pada Gambar 2.4 dapat dilihat pada rumus (2.1), (2.2), (2.3), dan (2.4), yaitu sebagai berikut [14].

1) Untuk menghitung nilai *hidden state* untuk waktu t, digunakan rumus (2.1) dengan sebuah fungsi aktivasi tanh yang memiliki rumus (2.2).

$$h_t = \tanh(W_{hh}\, h_{t-1} + W_{xh}\, x_t + b_h) \quad (2.1)$$

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x}-1}{e^{2x}+1} \quad (2.2)$$

Keterangan:

a) h_t adalah nilai *hidden state* pada langkah waktu t. Ini adalah memori dari jaringan. h_t dihitung berdasarkan *hidden state* sebelumnya dan masukan pada langkah atau tahap yang sedang berlangsung.

b) Fungsi $\tanh$ adalah fungsi aktivasi non-linear, selain $\tanh$ dapat juga digunakan fungsi aktivasi non-linear yang lain seperti ReLU.

c) W_{hh} adalah bobot yang digunakan untuk langkah waktu sebelumnya, berupa nilai *random* antara 0 sampai dengan 1.

d) h_{t-1} adalah nilai *hidden state* pada langkah waktu sebelumnya, biasanya diinisialisasi ke semua nol.

e) W_{xh} adalah bobot yang digunakan untuk nilai masukan, berupa nilai *random* antara 0 sampai dengan 1.

f) x_t adalah nilai masukan.

g) b_h adalah nilai bias, berupa nilai *random* antara 0 sampai dengan 1.

2) Untuk menghitung hasil keluaran sebagai prediksi, maka digunakan rumus (2.3) dengan fungsi *softmax* untuk menghitung probabilitas setiap label yang ditebaknya. Kelebihan dari fungsi aktifasi ini adalah nilai keluarannya berupa nilai dengan rentang probabilitas 0 sampai 1 dan apabila setiap hasil fungsi *softmax* dijumlahkan maka bernilai 1.

$$z_t = softmax(W_{hz}\, h_t + b_z) = \frac{\exp(W_{hz}\, h_t + b_z)}{\sum \exp(W_{hz}\, h_t + b_z)} \quad (2.3)$$

Keterangan:

a) z_t adalah keluaran yang dihasilkan setelah proses pada langkah waktu t.
b) $softmax$ adalah nilai eksponensial yang dinormalisasi.
c) W_{hz} adalah bobot yang digunakan untuk menghasilkan nilai keluaran, berupa nilai *random* antara 0 sampai dengan 1.
d) h_t adalah nilai *hidden state* pada langkah waktu t.
e) b_z adalah nilai bias, berupa nilai *random* antara 0 sampai dengan 1.

3) Untuk membandingkan hasil prediksi atau nilai keluaran yang telah dihasilkan dengan nilai target atau nilai yang sebenarnya, maka digunakan sebuah rumus *Cost Function* (2.4). Rumus *Cost Function* yang digunakan pada penelitian ini adalah *Mean Squared Error* (MSE). MSE adalah salah satu fungi untuk mengetahui seberapa besar nilai *error* yang dihasilkan oleh sebuah *machine learning* setelah melalui proses pelatihan.

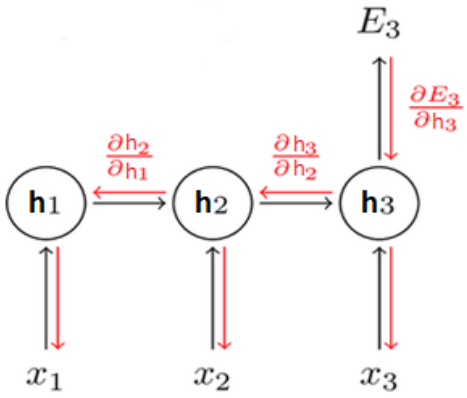
$$E = \frac{1}{n}\sum_{i=1}^{n}(y_i - z_i)^2 \quad (2.4)$$

Keterangan:

a) E adalah fungsi *error* atau fungsi *cost MSE*
b) y adalah nilai vektor target
c) z adalah nilai hasil keluaran dari perhitungan
d) n adalah jumlah kelas

2.5 Backpropagation Through Time (BPTT)

Backpropagation Through Time umumnya sama seperti *backpropagation* biasa pada *neural network*. Satu-satunya perbedaan utamanya adalah pada RNN *backpropagation* harus dilakukan pada setiap langkah waktu yang ada sepanjang jalur RNN yang berulang. *Backpropagation* mengacu pada dua hal, yang pertama merupakan sebuah metode matematika yang digunakan untuk menghitung turunan dan penerapan *chain rule*. Yang kedua merupakan sebuah algoritma pelatihan untuk memperbarui bobot jaringan untuk meminimalkan kesalahan [15]. Cara menghitung BPTT adalah dengan mencari nilai gradien mulai dari nilai *error* atau nilai *loss* hingga ke parameter (bobot dan bias) yang ingin diubah, seperti ilustrasi pada Gambar 2.5.



Gambar 2.5 Ilustrasi *Backpropagation Through Time*

Gradien adalah nilai yang digunakan untuk menyesuaikan parameter atau bobot yang sesuai dalam sebuah jaringan, agar jaringan tersebut dapat belajar. Dalam proses BPTT inilah terjadi proses perhitungan gradien. Semakin besar gradien, maka semakin besar penyesuaian, dan demikian pula sebaliknya. BPTT menggunakan konsep *chain rule*, yaitu hubungan antara

beberapa turunan (seperti rantai). Berikut adalah *chain rule* BPTT yang terdiri dari beberapa rumus beserta masing-masing turunannya, yaitu terurut (2.5) sampai dengan (2.17).

Rumus (2.6) dan (2.7) digunakan pada *chain rule* (2.5), yaitu untuk menghitung gradien bobot W_{hz}.

$$\frac{\partial E}{\partial W_{hz}} = \frac{\partial E}{\partial z}\ \frac{\partial z}{\partial W_{hz}} \tag{2.5}$$

$$\frac{\partial E}{\partial z} = -(y - z) \tag{2.6}$$

$$\frac{\partial z}{\partial W_{hz}} = (z)(1 - z)(h) \tag{2.7}$$

Rumus (2.6), (2.9), (2.10) dan (2.11) digunakan pada *chain rule* (2.8), yaitu untuk menghitung gradien bobot W_{hh}.

$$\frac{\partial E}{\partial W_{hh}} = \sum_{k=1}^{t-1} \frac{\partial E}{\partial z}\ \frac{\partial z}{\partial h_t}\ \frac{\partial h_t}{\partial h_k}\ \frac{\partial h_{t-1}}{\partial W_{hh}} \tag{2.8}$$

$$\frac{\partial z}{\partial h_t} = (z)(1 - z)(W_{hz}) \tag{2.9}$$

$$\frac{\partial h_t}{\partial h_k} = (h)(1 - h)(W_{hh}) \tag{2.10}$$

$$\frac{\partial h_{t-1}}{\partial W_{hh}} = (h)(1 - h)(h_{t-1}) \tag{2.11}$$

Rumus (2.6), (2.9), (2.10) dan (2.13) digunakan pada *chain rule* (2.12), yaitu untuk menghitung gradien bobot W_{xh}.

$$\frac{\partial E}{\partial W_{xh}} = \sum_{k=1}^{t-1} \frac{\partial E}{\partial z}\ \frac{\partial z}{\partial h_t}\ \frac{\partial h_t}{\partial h_k}\ \frac{\partial h_{t-1}}{\partial W_{xh}} \tag{2.12}$$

$$\frac{\partial h_{t-1}}{\partial W_{xh}} = (h)(1-h)(x) \tag{2.13}$$

Rumus (2.6) dan (2.15) digunakan pada *chain rule* (2.14), yaitu untuk menghitung gradien bias b_z.

$$\frac{\partial E}{\partial b_z} = \frac{\partial E}{\partial z}\frac{\partial z}{\partial b_z} \tag{2.14}$$

$$\frac{\partial z}{\partial b_z} = = (z)(1-z)(1) \tag{2.15}$$

Rumus (2.6), (2.9), (2.10) dan (2.17) digunakan pada *chain rule* (2.16), yaitu untuk menghitung gradien bobot b_h.

$$\frac{\partial E}{\partial b_h} = \sum_{k=1}^{t-1} \frac{\partial E}{\partial z}\frac{\partial z}{\partial h_t}\frac{\partial h_t}{\partial h_k}\frac{\partial h_{t-1}}{\partial b_h} \tag{2.16}$$

$$\frac{\partial h_{t-1}}{\partial b_h} = (h)(1-h)(1) \tag{2.17}$$

2.6 Stochastic Gradient Descent (SGD)

Penggunaan *Stochastic Gradient Descent* dalam *neural network* dimotivasi oleh nilai *cost* atau *loss* yang tinggi dan mengharuskan menjalankan *backpropagation* setelah dilakukan pelatihan. SGD dapat mengatasi nilai loss yang tinggi dengan cara memperbarui parameter atau bobot, serta bias, setelah dilakukan *backpropagation* atau pada RNN *backpropagation through time*. SGD melakukan pembaruan parameter atau bobot, serta bias untuk setiap contoh dengan rumus (2.18).

$$\theta' = \theta - \alpha \frac{\partial E}{\partial \theta} \tag{2.18}$$

Keterangan:

a) θ adalah nilai bobot atau bias yang diperbarui
b) θ' adalah nilai bobot atau bias yang baru

c) α adalah nilai learning rate

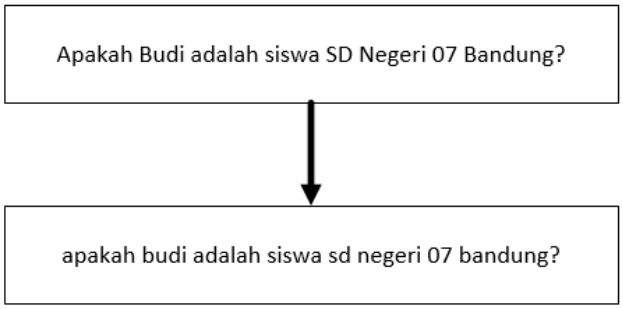
d) $\frac{\partial E}{\partial \theta}$ adalah nilai gradien dari *cost function*

2.7 *Text Pre-processing*

Text pre-processing adalah bagian penting dari setiap sistem pemrosesan bahasa alami, karena karakter, kata, dan kalimat yang diidentifikasi pada tahap ini adalah unit dasar atau awal sebelum pemrosesan lebih lanjut [16]. *Text pre-processing* dilakukan karena data teks sering mengandung berbagai macam format khusus atau berbeda-beda seperti format angka, format tanggal dan kata-kata yang tidak membantu dan dapat dihilangkan sehingga memudahkan proses selanjutnya.

2.8 *Case Folding*

Case folding adalah proses penyetaraan huruf untuk semua teks pada dokumen, yaitu dengan mengubah keseluruhan huruf menjadi bentuk yang sama, baik menjadi huruf kapital semua atau huruf kecil semua. *Case folding* dilakukan pada penelitian ini karena dokumen mengadung berbagai macam bentuk penulisan huruf, sehingga kata yang harusnya memiliki arti yang sama dapat dianggap berbeda oleh komputer dan akhirnya menghasilkan informasi yang tidak sesuai. Contoh proses *case folding* terdapat pada Gambar 2.6, di mana dilakukan proses *case folding* mengubah seluruh huruf menjadi huruf kecil.

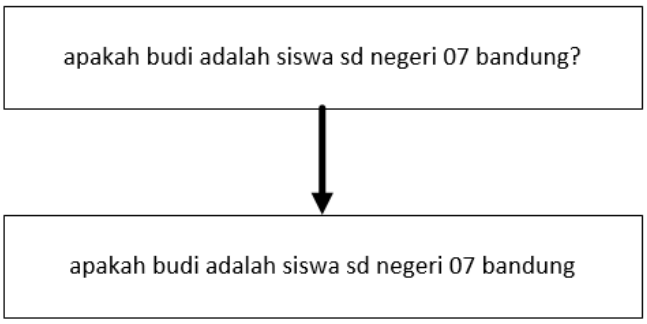


Gambar 2.6 Contoh Proses *Case Folding*

Pada contoh proses *case folding* pada Gambar 2.6, kata "'Budi" diubah menjadi "budi", "SD" diubah menjadi "sd", "Negeri" diubah menjadi "negeri", dan "Bandung" diubah menjadi "bandung".

2.9 *Filtering*

Filtering adalah proses melakukan penyaringan dan menghilangkan simbol-simbol yang ada di dalam dokumen. Proses *filtering* dilakukan untuk mencegah terjadinya salah pemahaman pada komputer. Kata yang terdapat simbol, di depan, di belakang atau pun di antara huruf-hurufnya, akan membuat kata tersebut dianggap oleh komputer memiliki makna yang berbeda dari yang seharusnya. Misalnya, pada Gambar 2.6, kata "bandung?" akan dianggap berbeda oleh komputer dengan kata "bandung". Berikut adalah hasil proses *filtering* yang dilakukan pada Gambar 2.6, yaitu pada Gambar 2.7.



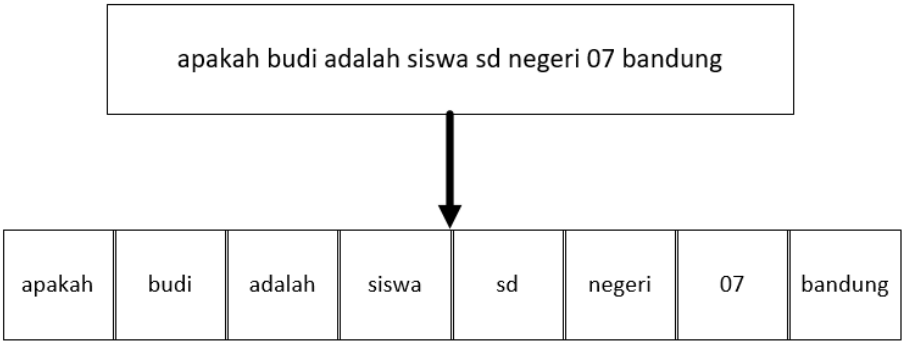
Gambar 2.7 Contoh Proses *Filtering*

Hasilnya, simbol "?" pada kata "bandung?" dihilangkan, sehingga menjadi "bandung".

2.10 *Tokenization*

Tokenization merupakan proses pemisahan setiap masing-masing kata dalam sebuah kalimat. Proses pemisahan tersebut dilakukan berdasarkan spasi

yang terdapat di antara setiap kata. Potongan kata atau disebut token kata digunakan untuk memudahkan proses selanjutnya, yaitu mengubah setiap kata menjadi vektor kata atau menjadi bentuk angka agar dapat dipahami oleh komputer. Proses *tokenization* dicontohkan pada Gambar 2.8, yaitu pada kalimat yang sebelumnya telah dilakukan proses *case folding* dan proses *filtering*.



Gambar 2.8 Contoh Proses *Tokenization*

2.11 *Removing Stopwords*

Removing stopwords merupakan proses penghilangan kata tidak penting yang terdapat dalam *stoplist*. Kata-kata yang dihilangkan tersebut menunjukan kurang relevansinya dengan teks dan didefinisikan dalam *stopword list*. *Stopword list* yang digunakan pada penelitian ini menggunakan *stopword list* yang digunakan pada penelitian sebelumnya yang dilakukan oleh Tommy Edityantama Hutapea [17], yang dapat dilihat pada Tabel 2.1.

Tabel 2.1 *Stopword List*

ID	Kata		ID	Kata		ID	Kata
1	yang		10	ada		19	menjadi
2	untuk		11	jika		20	masih
3	saya		12	cara		21	aja
4	harus		13	tetap		22	itu
5	saja		14	dengan		23	dalam
6	terdapat		15	hanya		24	pada
7	setiap		16	seperti		25	ini
8	di		17	atau		26	ke

9	tentang		18	dari		27	yg

Contoh dari proses *removing stopword* ditampilkan pada Gambar 2.9.

apakah		apakah
budi		budi
adalah		
siswa		siswa
sd		sd
negeri		negeri
07		07
bandung		bandung

Gambar 2.9 Contoh Proses *Removing Stopword*

2.12 TF-IDF

Metode TF-IDF merupakan metode untuk menghitung bobot setiap kata yang paling umum digunakan pada *information retrieval*. Metode ini juga terkenal efisien, mudah dan memiliki hasil yang akurat. Metode ini akan menghitung bobot setiap *term* dalam sebuah dokumen dengan rumus (2.19)

$$IDF_{t_i} = log(\frac{D}{df_i}) \quad (2.19)$$

Keterangan:

a) D adalah total keseluruhan dokumen.
b) df_i adalah banyaknya dokumen yang mengandung kata ke–i yang dicari.

Kemudian rumus yang digunakan untuk menghitung bobot (w) *term* dari masing-masing dokumen terhadap term menggunakan persamaan dapat dilihat pada rumus (2.20).

$$w_i = tf_i * IDF_{t_i} \quad (2.20)$$

Keterangan:

a) tf_i adalah *term* frekuensi/ frekuensi kata.
b) W_i adalah bobot dokumen ke-d terhadap term ke-t.

2.13 *Normalization*

Data yang bersifat *sequence* perlu diskala saat melatih *neural network*, demikian pula *recurrent neural network*. Ketika sebuah jaringan memiliki data tak berskala yang memiliki rentang nilai besar, maka adalah data yang besar tersebut akan memperlambat proses *training* dan konvergensi jaringan, bahkan dalam beberapa kasus mencegah jaringan melakukan *training* secara efektif. Normalisasi dilakukan untuk melakukan penskalaan data dari rentang asli sehingga semua nilai hanya berada dalam rentang 0 dan 1. Proses normalisasi dilakukan dengan rumus (2.21) [18].

$$y = \frac{(x - \min)}{(\max - \min)} \quad (2.21)$$

Keterangan:

a) y adalah nilai data masukan yang telah dinormalisasi.
b) x adalah nilai data masukan sebelum dinormalisasi.
c) min adalah nilai terkecil dari seluruh data masukan sebelum dinormalisasi.
d) max adalah nilai terbesar dari seluruh data masukan sebelum dinormalisasi.

2.14 *One Hot Encoding*

Machine learning tidak dapat mengenali data berupa kategorikal secara langsung. Label berupa kategorikal yang harus diubah agar dapat dikenali oleh *machine learning*. Maka dari itu *one hot encoding* digunakan untuk merepresentasikan data kategorikal menjadi dikenali oleh *machine learning* [19].

Misalnya terdapat label data Waktu Autodebet, Informasi Autodebet, Waktu PMB, dan Informasi PMB. Total label data tersebut adalah empat. Keempat label data tersebut dapat diubah menjadi sebuah index yang unik. Lalu, setelah menjadi masing-masing index yang berbeda, selanjutnya seluruh label data tersebut diubah menjadi vektor sesuai dengan index yang dimilikinya. Contoh hasil *One Hot Encoding* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Hasil *One Hot Encoding*

Label Data	Index	Vektor
Waktu Autodebet	0	$\begin{bmatrix}1\\0\\0\\0\end{bmatrix}$
Informasi Autodebet	1	$\begin{bmatrix}0\\1\\0\\0\end{bmatrix}$
Waktu PMB	2	$\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$
Informasi PMB	3	$\begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

2.15 *Database*

Database adalah suatu kumpulan file yang terkait atau tabel yang berisi data. *Database* adalah kumpulan logis terorganisir data terkait dirancang dan dibangun untuk tujuan tertentu, teknologi untuk menarik bersama-sama

fakta-fakta yang memungkinkan untuk dipilih dan dicampurkan untuk mencocokkan data. Data dalam *database* memiliki beberapa makna yang melekat. Dengan kata lain, berbagai macam data tidak benar disebut *database*. Sebuah *database* dapat dari berbagai ukuran dan tingkat kerumitan, dan itu dapat dipertahankan secara manual atau dengan perangkat lunak pada komputer [20].

2.16 *Entity Relationship Diagram* (ERD)

ERD merupakan suatu model jaringan yang menggunakan susunan data yang disimpan pada sistem secara abstrak. ERD juga menggambarkan hubungan antara satu entitas yang memiliki sejumlah atribut dengan entitas lain dalam suatu sistem yang terintegrasi. ERD digunakan oleh perancang sistem untuk memodelkan data yang nantinya akan dikembangkan menjadi basis data. ERD ini juga merupakan model konseptual yang dapat mendeskripsikan hubungan antara file yang digunakan untuk memodelkan struktur data serta hubungan antar data [21].

2.17 *Unified Modeling Language* (UML)

UML adalah salah satu bahasa pemodelan untuk memodelkan suatu perangkat lunak agar perangkat lunak tersebut dapat tergambar secara detail. Keuntungan menggunakan UML yaitu perangkat lunak yang dimodelkan akan terdokumentasi dengan baik maka ketika perangkat lunak sudah berjalan lama dan berganti developer maka developer yang baru akan mudah mengerti dengan membaca UML tersebut [22].

Pada UML terdapat beberapa diagram yang digunakan pada penelitian ini, yaitu [22]:

1. *Use case diagram*

   Use case diagram adalah diagram untuk mengambarkan fungsi yang ada pada sebuah perangkat lunak. Tujuan dari *use case diagram* untuk

menggambarkan relasi antara fungsi pada perangkat lunak dengan aktor atau bisa disebut pengguna.

2. *Use case scenario*

   Use case scenario adalah deskripsi dari sebuah *use case*. Penjelasan ini diperlukan untuk mendetailkan fungsi agar lebih dimengerti oleh pembaca terutama *developer*.

3. *Activity diagram*

   Activity diagram digunakan untuk menggambarkan alur dari proses, bisnis proses, alur pada *use case diagram*.

4. *Class diagram*

   Class diagram digunakan untuk menggambarkan *class* apa saja yang nantinya akan ada pada perangkat lunak tersebut. *Class* yang digambarkan berasal dari objek-objek pada perangkat lunak tersebut.

5. *Sequence diagram*

   Sequence diagram digunakan untuk menggambarkan komunikasi antar *class* yang dibuat pada *class* diagram dan juga komunikasi antar *class* tersebut dengan aktor.

2.18 Flow Chart

Flow chart atau Bagan Alir Program merupakan bagan yang menjelaskan secara rinci langkah-langkah dari proses program. *Flow chart* dibuat menggunakan simbol-simbol yang terdapat pada daftar simbol sebelumnya. *Flow chart* ini akan menggambarkan logika setiap langkah proses dari sebuah program, dimana logika ini bisa dianalogikan dengan logika yang ada pada kehidupan sehari-hari [23].

2.19 MySQL

MySQL adalah sebuah perangkat lunak sistem manajemen basis data *Structured Query Language* (SQL) atau DBMS yang *multithread*, *multi-user*,

dengan sekitar enam juta instalasi di seluruh dunia. MySQL AB membuat MySQL tersedia sebagai perangkat lunak gratis di bawah lisensi GNU *General Public License* (GPL), tetapi mereka juga menjual di bawah lisensi komersial untuk kasus-kasus dimana penggunaannya tidak cocok dengan penggunaan GPL [24].

2.20 Python

Pada saat ini bahasa pemrograman python sering digunakan untuk melakukan penelitian atau pembuatan aplikasi machine learning dikarenakan banyak library yang mendukung dan bahasa pemrograman python sama seperti matematika. Kelebihan dari bahasa pemrograman python yaitu banyaknya library yang sedang dikembangkan, mudah untuk dipelajari dan dapat diintegrasikan dengan bahasa pemrograman lain seperti C, C++ atau java. Walaupun begitu bahasa pemrograman python memiliki beberapa kelemahan yaitu lemah dalam bidang teknologi mobile, lemah dalam pengembangan database layer dan run-time *error* (*error* hanya terlihat pada saat program sudah dijalankan) [25].

Python adalah bahasa pemrograman yang bersifat *open source*. Bahasa pemrograman ini dioptimalisasikan untuk *software quality*, *developer productivity*, *program portability*, dan *component integration*. Python telah digunakan untuk mengembangkan berbagai macam perangkat lunak, seperti *internet scripting*, *systems programming*, *user interfaces*, *product customization*, *numberic programming*, dan lain-lain [26].

2.21 Hypertext Markup Language (HTML)

Hypertext Markup Language (HTML) adalah serangkaian kode program yang merupakan dasar dari representasi visual sebuah halaman web. Di dalamnya berisi kumpulan informasi yang disimpan dalam *tag* tertentu, dimana *tag* tersebut digunakan untuk melakukan format terhadap informasi

yang dimaksud. Berbagai pengembangan telah dilakukan terhadap kode HTML dan telah melahirkan teknologi-teknologi baru di dalam dunia pemrograman web. Kendati demikian, sampai sekarang HTML tetap berdiri kokoh sebagai dasar dari bahasa web seperti PHP, ASP, JSP dan lainnya. Bahkan secara umum, mayoritas situs web yang ada di Internet pun masih tetap menggunakan HTML sebagai teknologi utama mereka [27].

2.22 Confusion Matrix

Confusion matrix adalah sebuah tabel matriks yang menyatakan jumlah data uji yang benar diklasifikasikan dan jumlah data uji yang salah diklasifikasikan. Berikut adalah persamaan untuk menghitung akurasi pada confusion matrix, yaitu pada rumus (2.22) [28].

$$Akurasi = \frac{TP + TN}{TP + FN + FP + TN} x100\% \quad (2.22)$$

Keterangan:

a) *True Positive* (TP), yaitu jumlah dokumen dari kelas 1 yang benar dan diklasifikasikan sebagai kelas 1.
b) *True Negative* (TN), yaitu jumlah dokumen dari kelas 0 yang benar diklasifikasikan sebagai kelas 0.
c) *False Positive* (FP), yaitu jumlah dokumen dari kelas 0 yang salah diklasifikasikan sebagai kelas 1.
d) *False Negative* (FN), yaitu jumlah dokumen dari kelas 1 yang salah diklasifikasikan sebagai kelas 0.