BAB 2
LANDASAN TEORI

2.1 Analisis Sentimen

Analisis sentimen atau disebut juga *opinion mining* adalah bidang studi untuk menganalisis pendapat, sentimen, evaluasi, penilaian sikap dan emosi terhadap entitas seperti produk, jasa, organisasi, individu, peristiwa, topik dan atribut lainnya. Analisis sentimen dan *opinion mining* berfokus kepada opini yang mengekspresikan sentimen positif dan negatif [1].

Analisis sentimen salah satu masalah penelitian yang populer baik dalam aplikasi kehidupan nyata atau dalam bidang akademik. Secara umum, analisis sentimen sudah dilakukan penelitian pada 3 level:

1. *Document Level*

   Tugas pada level ini adalah untuk mengklasifikasi apakah seluruh pendapat dokumen mengekspresikan sentimen positif atau negatif.

2. *Sentence Level*

   Tugas pada level ini adalah untuk menentukan apakah setiap kalimat menyatakan pendapat positif, negatif, atau netral. Netral biasanya tidak ada pendapat.

3. *Entity* dan *Aspect Level*

   Tugas pada level ini adalah menentukan sentimen (positif atau negatif) berdasarkan target (aspek) dalam suatu kalimat. Seringkali analisis level dokumen atau level kalimat tidak menemukan apa yang sebenernya disukai dan tidak disukai orang.

2.1.1 Analisis Sentimen Berdasarkan Aspek

Analisis sentimen berdasarkan aspek adalah salah satu domain kasus *opinion mining* yang bertujuan untuk mendeteksi polaritas teks tertulis berdasarkan dengan aspek tertentu. Klasifikasi teks opini di level dokumen atau kalimat seringkali tidak cukup karena level tersebut hanya bisa mengindentifikasi

sentimen secara umum. Jika diasumsikan sebuah dokumen atau kalimat memiliki sebuah sentimen positif, tidak berarti semua aspek yang berada di dokumen atau kalimat tersebut semuanya positif begitupun untuk dokumen atau kalimat yang mengandung sentimen negatif. Untuk analisis lebih lengkap perlu ditemukan aspek dan menentukan sentimen positif atau negatif pada setiap aspek [1].

Ada 2 tugas utama untuk dalam melakukan analisis sentimen berdasarkan aspek antara lain:

1. *Aspect Extraction*

   Pada tahap ini mengekstrak aspek yang sudah ditentukan sebelumnya. Misalnya, dalam kalimat "kualitas suara ponsel ini luar biasa", aspeknya adalah "kualitas suara" entitasnya "ponsel ini" . Pada kata "ponsel ini" tidak menunjukan aspek umum karena evaluasi bukan tentang ponsel secara kesuluruhan tetapi hanya tentang "kualitas suara".
2. *Aspect Sentiment Classification*

   Pada tahap ini menentukan apakah pendapat dari berbagai aspek kedalam sentimen positif, negatif atau netral. Pada contoh kalimat "kualitas suara ponsel ini luar biasa" sentimen dari aspek "kualitas suara" adalah positif.

Tujuan analisis sentimen berdasarkan aspek adalah untuk mengidentifikasi aspek dari suatu entitas, dan sentimen yang diungkapkan oleh penulis komentar tentang aspek tersebut [9]. Misalnya, dari kalimat "makanan di restoran ini sangat enak tapi pelayanannya kurang bagus" pada kalimat tersebut kata "makanan" dan "pelayanan" diidentifikasikan sebagai aspek yang kemudian ditentukan sentimennya dengan kata "enak", untuk aspek "makanan" dan mengandung sentimen positif sedangkan kata "kurang bagus", untuk aspek "pelayanan" dan mengandung sentimen negatif. Pada tahap mengidentifikasikan aspek ada beberapa pendekatan yaitu, *frequency based*, *relation based*, *supervised learning*, dan *topic modelling* dan untuk penentuan sentimen terhadap aspek mempunyai 2 pendekatan yaitu, *supervised learning* dan *lexicon based* [1].

Pada penelitian ini algoritma yang digunakan untuk klasifikasi sentimen adalah *supervised learning* sehingga untuk mengembangkan sistem ABSA,

dataset yang sudah di anotasi sangat penting untuk melakukan proses pelatihan dan pengujian. Dataset yang digunakan pada penelitian ini diambil dari penelitian A. Cahyadi.

2.2 Semantic Evaluation (SemEval)

SemEval (*Semantic Evaluation*) adalah serangkaian dari *workshop* yang berfokus pada evaluasi dan perbandingan sistem yang dapat menganalisis berbagai fenomena semantik dalam teks dengan tujuan memperluas *state of the art* dalam analisis semantik dan menciptakan dataset beranotasi berkualitas tinggi dalam meningkatkan tatangan pada semantik bahasa alami [10].

Tujuan dari SemEval dan Senseval adalah untuk mengevaluasi sistem analisis semantik. "*Semantic Analysis*" mengacu pada analisis formal makna, dan "*computational*" merujuk pada pendekatan yang pada prinsipnya mendukung implementasi yang efektif. SemEval menyediakan forum yang menarik bagi para peneliti untuk mengusulkan masalah penelitian yang menantang dalam semantik dan untuk membangun sistem/teknik untuk mengatasi masalah penelitian tersebut.

Tiga *workshop* pertama Senseval-1 hingga Senseval-3, difokuskan pada *word sense disambiguation,* seiring berjalannya waktu jumlah bahasa yang ditawarkan pada tugas Senseval bertambah dan banyak tim yang berpartisipasi juga bertambah. Dimulai dari *workshop* keempat, SemEval-2007 (SemEval-1), sifat tugas berkembang yang mencakup tugas *semantic analysis* selain tugas *word sense disambiguation*.

- Senseval and SemEval *tasks overview*

Tabel 2.1 menunjukan pertumbuhan workshop dari Senseval ke SemEval dan memberikan gambaran tentang bidang semantik komputasi mana yang dievaluasi di seluruh workshop Senseval/SemEval.

Tabel 2.1 Senseval dan SemEval Taks

Workshop	*No. of*	*Areas of study*	*Languages of*

	Tasks		*Data Evaluated*
Senseval-1	3	*Word Sense Disambiguation (WSD) - Lexical Sample WSD tasks*	English, French, Italian
Senseval-2	12	*Word Sense Disambiguation (WSD) - Lexical Sample, All Words, Translation WSD tasks*	Czech, Dutch, English, Estonian, Basque, Chinese, Danish, English, Italian, Japanese, Korean, Spanish, Swedish
Senseval-3	16 (incl. 2 cancelled)	*Logic Form Transformation, Machine Translation (MT) Evaluation, Semantic Role Labelling, WSD*	Basque, Catalan, Chinese, English, Italian, Romanian, Spanish
SemEval2007	19 (incl. 1 cancelled)	*Cross-lingual, Frame Extraction, Information Extraction, Lexical Substitution, Lexical Sample, Metonymy, Semantic Annotation, Semantic Relations, Semantic Role Labelling, Sentiment Analysis, Time Expression, WSD*	Arabic, Catalan, Chinese, English, Spanish, Turkish
SemEval2010	18 (incl. 1 cancelled)	*Coreference, Cross-lingual, Ellipsis, Information Extraction, Lexical Substitution, Metonymy, Noun Compounds, Parsing, Semantic Relations, Semantic Role Labeling, Sentiment Analysis, Textual Entailment, Time Expressions, WSD*	Catalan, Chinese, Dutch, English, French, German, Italian, Japanese, Spanish
SemEval2012	8	*Common Sense Reasoning, Lexical Simplification, Relational Similarity, Spatial Role Labelling, Semantic Dependency Parsing, Semantic and Textual Similarity*	Chinese, English
SemEval2013	14	*Temporal Annotation, Sentiment Analysis, Spatial Role Labeling, Noun Compounds, Phrasal Semantics, Textual Similarity,*	Catalan, French, German, English, Italian, Spanish

		Response Analysis, Cross-lingual Textual Entailment, BioMedical Texts, Cross and Multilingual WSD, Word Sense Induction, and Lexical Sample	
SemEval2014	10	*Compositional Distributional Semantic, Grammar Induction for Spoken Dialogue Systems, Cross-Level Semantic Similarity, Sentiment Analysis, L2 Writing Assistant, Supervised Semantic Parsing, Clinical Text Analysis, Semantic Dependency Parsing, Sentiment Analysis in Twitter, Multilingual Semantic Textual Similarity*	English, Spanish, French, German, Dutch,
SemEval2015	18 (incl. 1 cancelled)	*Text Similarity and Question Answering, Time and Space, Sentiment, Word Sense Disambiguation and Induction, Learning Semantic Relations*	English, Spanish, Arabic, Italian
SemEval2016	14	*Textual Similarity and Question Answering, Sentiment Analysis, Semantic Parsing, Semantic Analysis, Semantic Taxonomy*	
SemEval2017	12	*Semantic comparison for words and texts, Detecting sentiment, humor, and truth and Parsing semantic structures*	
SemEval2018	12	*Affect and Creative Language in Tweets, Coreference, Information Extraction, Lexical Semantics and Reading Comprehension and Reasoning*	

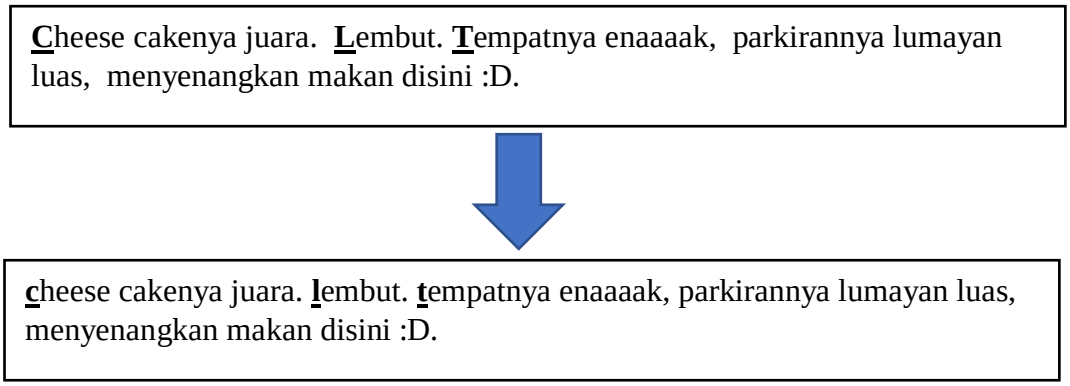
2.3 *Preprocessing*

Preprocessing adalah proses pengolahan data asli yang sebelumnya tidak terstruktur menjadi terstruktur [11]. Proses ini bertujuan agar data yang akan digunakan dalam proses klasifikasi sentimen bersih dari *noise.* Oleh karena itu, dibutuhkan proses yang dapat mengubah data yang sebelumnya tidak terstruktur menjadi data yang terstruktur untuk diproses ke tahap selanjutnya. Adapun tahapan dari *Preprocessing* pada penelitian ini antara lain, *case folding, filtering,*

word normalization, tokenization, dan stopword removal, penambahan token aspek, dan *one hot encoding.*

2.3.1 *Case folding*

Case folding merupakan proses penyeragaman bentuk huruf untuk semua teks pada dokumen baik itu menjadi huruf kecil (*lowercase*) atau huruf kapital (*uppercase*). Pada penelitian ini bentuk huruf memiliki bentuk yang beragam sehingga *case folding* dilakukan untuk menghilangkan *noise*. Contoh pada *case folding* ditunjukan pada Gambar 2.1.

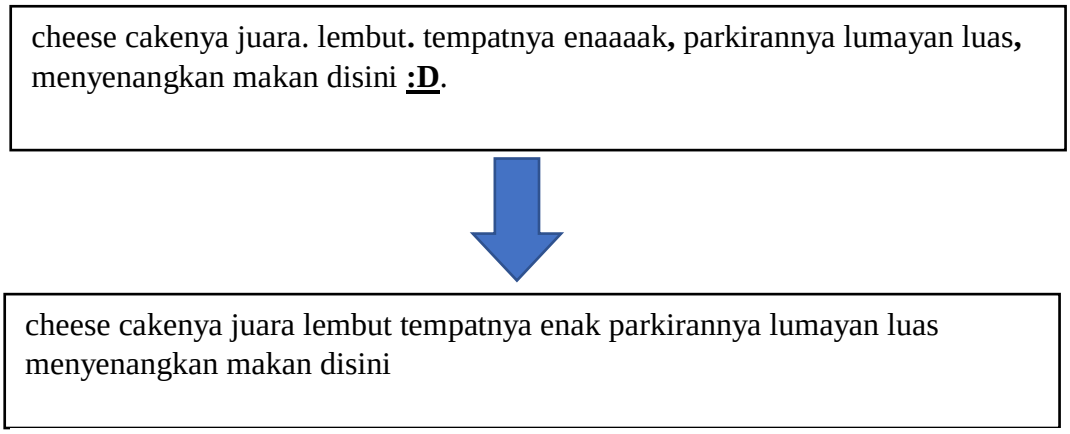


Gambar 2.1 Contoh Proses *Case folding*

Pada Gambar 2.1 kata “Cheese”, “Lembut”, dan “Tempatnya” diubah menjadi huruf kecil pada proses *case folding.*

2.3.2 *Filtering*

Filtering merupakan proses membersihkan simbol-simbol yang tidak diperlukan dalam dokumen seperti tanda baca, angka, dan *emoticon.* Pada penelitian ini *emoticon* seperti :D, :), :(, ^^, :3 akan dihilangkan. Selain itu tanda baca seperti titik (.), koma (,) dan tanda baca lainnya akan dihilangkan. Jika



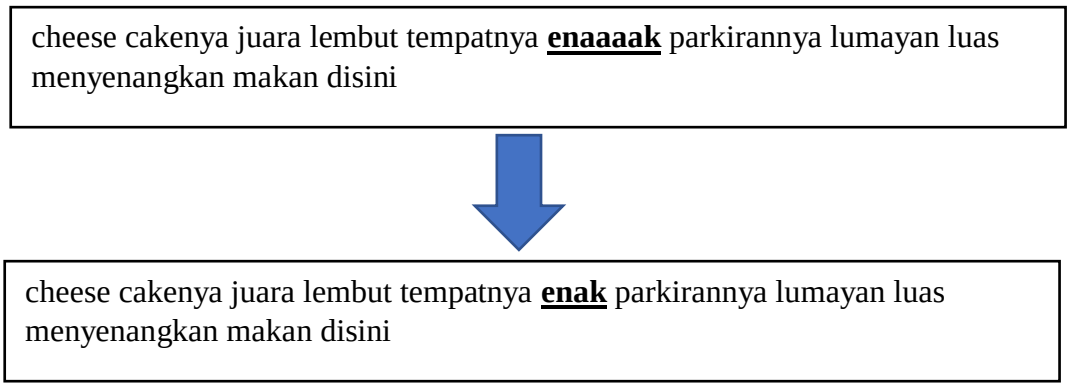
Gambar 2.2 Contoh Proses *Filtering*

terdapat angka akan dihilangkan, jika terdapat spasi yang lebih dari satu maka akan diganti menjadi satu spasi. Contoh *Filtering* ditujukan pada Gambar 2.2, yaitu pada kalimat yang sebelumnya telah dilakukan proses *case folding*.

Pada Gambar 2.2, semua tanda baca titik (.), koma (,) dan emoticon ":D" dihilangkan.

2.3.3 *Word Normalization*

Word normalization merupakan proses mengubah kata tidak baku menjadi kata baku [12]. Pada penelitian ini dataset yang digunakan diambil penelitian A. Cahyadi sering kali ditemukan kata yang tidak baku yang bisa menjadi *noise* pada proses selanjutnya. Pada proses *word normalization* kata yang terdapat pada list kata tidak baku akan diubah menjadi kata baku, jika terdapat kata yang hurufnya berulang seperti, "eeenaaaak" atau "enaaak" akan dihapus huruf berulang tersebut yang menghasilkan kata "enak" tanpa pengulangan huruf. Contoh proses *word normalization* ditujukan Gambar 2.3. Pada penelitian ini menggunakan library inaNLP [12] untuk proses *word normalization*.

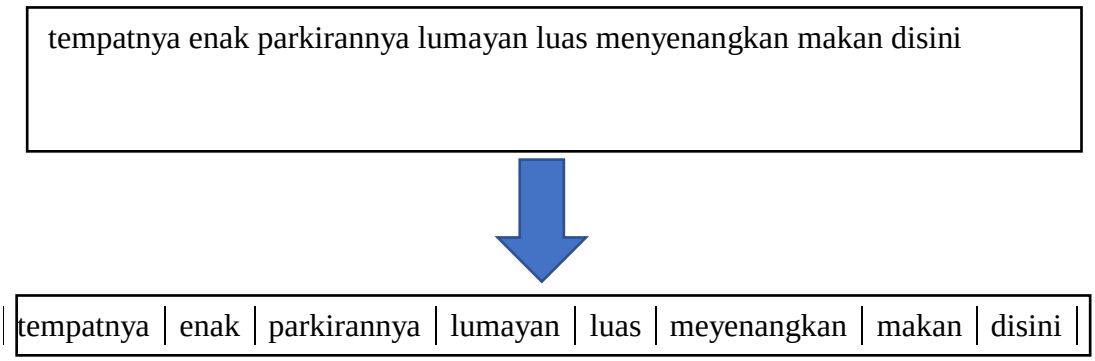


Gambar 2.3 Contoh Proses *Word Normalization*

Pada Gambar 2.3, kata “enaaaak” diubah menjadi kata “enak” karena mempunyai huruf yang berulang.

2.3.4 Tokenization

Tokenization merupakan proses pemisahan setiap kata dalam sebuah kalimat [13]. Proses pemisahan kata dilakukan berdasarkan spasi diantara setiap kata dalam kalimat. Hasil dari proses *tokenization* akan digunakan sebagai masukan dalam tahap transformasi teks ke dalam bentuk angka. Pada penelitian ini proses *tokenization* ditunjukan pada Gambar 2.4.

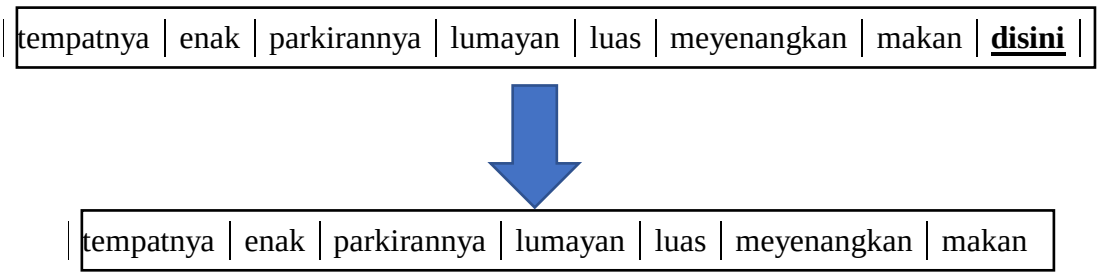


Gambar 2.4 Contoh Proses *Tokenization*

Pada Gambar 2.4 kalimat akan dipisahkan berdasarkan spasi menjadi token kata.

2.3.5 *Stopword Removal*

Stopword removal merupakan proses penghilangan kata-kata umum yang biasanya muncul dalam jumlah besar dan dianggap tidak memiliki makna seperti kata-kata di, yang, ke, oleh dan lain-lain. Proses penghilangkan *stopword* berfungsi untuk mengurangi jumlah kata yang akan diproses yang tidak ada relevansinya dengan teks [13]. Pada penelitian ini kata akan dihilangkan jika terdapat pada list *stopword*, *list stopword* yang digunakan diambil dari library inaNLP [12]. Proses *stopword removal* ditunjukan pada Gambar 2.5.

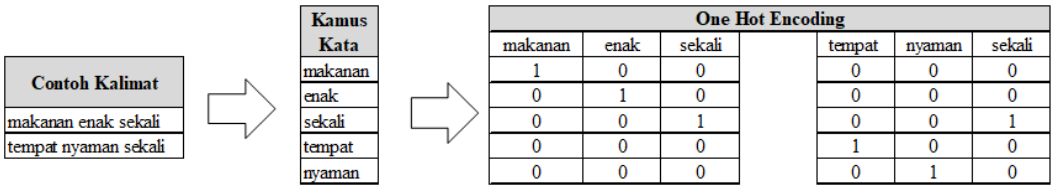


Gambar 2.5 Contoh Proses *Stopword Removal*

Pada Gambar 2.5 kata “disini” dihilangkan karena terdapat pada list *stopword*.

2.3.6 *One Hot Encoding*

Algoritma *machine learning* atau *neural network* tidak dapat memproses data kategorikal secara langsung. Variable input dan output yang akan di proses oleh *machine learning* harus berupa numerik. Salah satu teknik untuk merubah variabel kategorikal menjadi variabel numerik adalah *one hot encoding*. *One hot encoding* merupakan proses mengubah satu variabel dengan n kamus kata dan nilai d unik, ke d variabel biner dengan n kamus kata masing-masing [14]. Setiap d variabel biner akan bernilai bernilai 1, jika terdapat dalam indeks n kamus data dan sisanya akan menjadi 0.



Gambar 2.6 Contoh *One Hot Encoding*

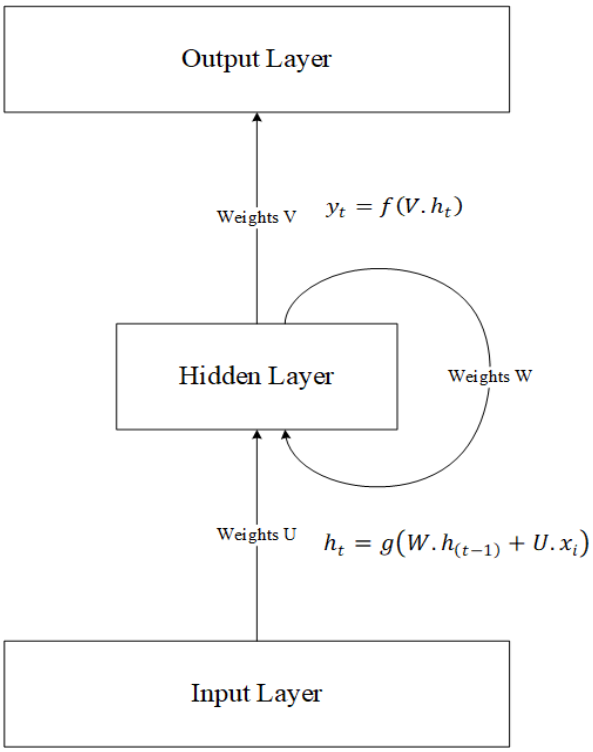
Pada Gambar 2.6 terdapat 2 kalimat, pertama dari 2 kalimat tersebut akan dibuat kamus kata berbeda (unik) kemudian akan di lakukan proses *one hot encoding* pada masing-masing kata dalam kalimat. Kata yang terdapat pada kamus kata akan bernilai 1 dan sisanya bernilai 0 seperti kata "makanan" pada kalimat pertama bernilai 1 dan sisanya bernilai nol.

2.4 *Recurrent Neural Network*

Recurrent Neural Network (RNN) adalah jaringan dengan *loop,* yang memungkinkan informasi bertahan dalam jaringan, RNN memiliki koneksi umpan balik ke jaringan itu sendiri yang memungkinkan aktivasi mengalir kembali dalam satu lingkaran mempelajari urutan dan informasi sebelumnya [2]. RNN sangat *powerful* dalam memodelkan data sekuensial, ucapan atau teks dan diterapkan pada data non-sekuensial untuk melatih secara non-sekuensial, RNN bisa digunakan dalam kasus *image, video captioning, word prediction, word translation, image processing, speech recognition, natural language processing, music processing* dan sebagainya.

2.4.1 *Elman Recurrent Neural Network*

Elman Recurrent Neural Netwrok (ERNN) merupakan salah satu dari arsitektur Recurrent Neural Network (RNN), *Elman Recurrent Neural Network* (ERNN) termasuk ke dalam *Simple Recurrent Neural Network* (SRNN) [2]. ERNN melakukan proses *learning* dengan membuat salinan neuron layer hidden pada layer input disebut sebagai *context input (hidden layer sebelumnya)*, sehingga salinan ini berfungsi sebagai perpanjangan dari *input layer*. Fungsi dari *context input* ini adalah untuk menyimpan status ataupun keadaaan sebelumnya dari *hidden layer*, untuk kemudian disampaikan kembali kepada *hidden layer*. Hubungan antara *context input* dan *hidden layer* adalah terhubung penuh (*fully connected*) dan diberi bobot 1. Selain ERNN arsitektur yang termasuk *Simple Recurrent Neural Network* adalah *Jordan Recurrent Neural Network*, perbedaannya adalah model Jordan melakukan proses *learning* dengan membuat salinan *output layer* pada *input layer* yang disebut sebagai *state layer*, Dengan proses ini, hasil output pada iterasi sebelumnya akan menjadi bagian dari input



Gambar 2.7 Arsitektur *Elman Recurrent Neural Network*

pada iterasi selanjutnya.

Pada gambar 2.7 *Elman Recurrent Neural Network* memerlukan 3 parameter bobot diantarannya, V bobot dari *input layer* ke *hidden layer*, U bobot dari *context layer* (*hidden layer* sebelumnya) ke *hidden layer*, dan W bobot dari *hidden layer* ke *output layer*. Pada proses pelatihan ketiga bobot tadi akan diperbaharui hingga mendapatkan bobot yang optimal. Dalam melakukan *forward propagation* ERNN menggunakan persamaan (2.1).

$$h_t = g\left(W.h_{(t-1)} + U.x_i\right) \quad \textbf{(2.1)}$$

dimana:

h_t = *hidden layer* pada *timestep* ke-t

W = bobot dari neuron hidden sebelumnya menuju ke neuron hidden selanjutnya

x_i = nilai neuron input ke-i

U = bobot dari neuron *input* menuju ke neuron hidden

t = *timestep*

g = *activation function* untuk mendapat nilai keluaran pada *hidden layer*, pada penelitian ini *activation function* yang digunakan pada *hidden layer* adalah *Tanh*

Sedangkan nilai *output layer* waktu ke t didapatkan dengan persamaan (2.2).

$$y_t = f(V.h_t) \quad \textbf{(2.2)}$$

dimana :

y_t = *output layer* pada waktu ke t

V = bobot dari neuron hidden menuju ke neuron output

h_t = nilai neuron hidden ke-t

f = *activation function* untuk mendapat nilai keluaran pada *output layer*, pada penelitian ini *activation function* yang digunakan pada *output layer* adalah

Softmax

2.4.2 *Activation Function* (Fungsi Aktivasi)

Fungsi aktivasi sangat umum digunakan dalam *neural network*. Alasan utama menggunakan fungsi aktivasi adalah agar *neural network* mengenali data yang *non-linear*, karena output yang dihasilkan dari neural network jarang sekali bersifat *linear* [15]. Fungsi aktivasi juga digunakan untuk mengaktifkan dan menonaktifkan *neuron*, karakteristik *neural network* ditentukan oleh bobot dan input-output fungsi aktivasi yang diterapkan. Terdapat banyak jenis fungsi aktivasi pada neural network, namun yang di gunakan dalam penelitian ini fungsi aktivasi *hyperbolic tangent* dan *softmax*.

2.4.2.1 Tanh (Hyperbolic Tangent)

Fungsi aktivasi *hyperbolic tangent* adalah tipe aktivasi fungsi dalam *deep learning* dan memiliki beberapa varian, fungsi aktivasi *hyperbolic tangent* dikenal juga sebagai fungsi *tanh* yang menghasilkan nilai -1 sampai 1 [15]. Fungsi *tanh* memberikan kinerja pelatihan yang lebih baik untuk *multi layer neural network* dibandingkan fungsi *sigmoid*. Fungsi *tanh* telah banyak digunakan dalam *recurrent neural network* untuk kasus *natural language processing* dan *speech recognition*. Persamaan dari fungsi tanh dapat dilihat pada persamaan (2.3).

$$tanh(x) = \frac{\sin(x)}{\cos(x)} = \left(\frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}\right) = \left(\frac{e^{2x} - 1}{e^{2x} + 1}\right) \quad \textbf{(2.3)}$$

Adapun untuk turunan aktivasi fungsi *tanh* yang nantinya akan digunakan pada proses BPTT dalam algoritma ERNN [16].

$$derivative(tanh) = (1 - o^{2}) \quad \textbf{(2.4)}$$

Dimana o adalah output dari *hidden layer*.

Pada penelitian ini fungsi aktifasi *tanh* digunakan untuk menghitung nlai keluaran pada *hidden layer* untuk waktu t .

2.4.2.2 Softmax

Fungsi aktivasi *softmax* digunakan untuk menghitung probabilitas pada hasil output, yang terjadi di *output layer* dimana akan diambil nilai probabilitas paling besar sebagai prediksi. *Softmax* digunakan untuk menghitung probabilitas distribusi dari vektor bilangan real. Fungsi softmax menghasilkan output yang merupakan kisaran nilai antara 0 dan 1, dengan jumlah probabilitas sama dengan 1 [15]. Persamaan dari fungsi sofmax dapat dilihat pada persamaan (2.5).

$$softmax(xi) = \frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad \textbf{(2.5)}$$

Adapun turunan dari softmax yang nantinya akan digunakan pada proses BPTT algoritma ERNN [16].

$$derivative(softmax) = (y_t - label_t) \otimes s_t \quad \textbf{(2.6)}$$

Dimana y_t adalah prediksi label ERNN ke t, $label_t$ adalah label sebenarnya (true label) ke t dan s_t *hidden layer* ke t.

Pada penelitian ini fungsi aktivasi *softmax* digunakan untuk menghitung nilai keluaran pada *output layer* waktu t.

2.4.3 *Loss Function*

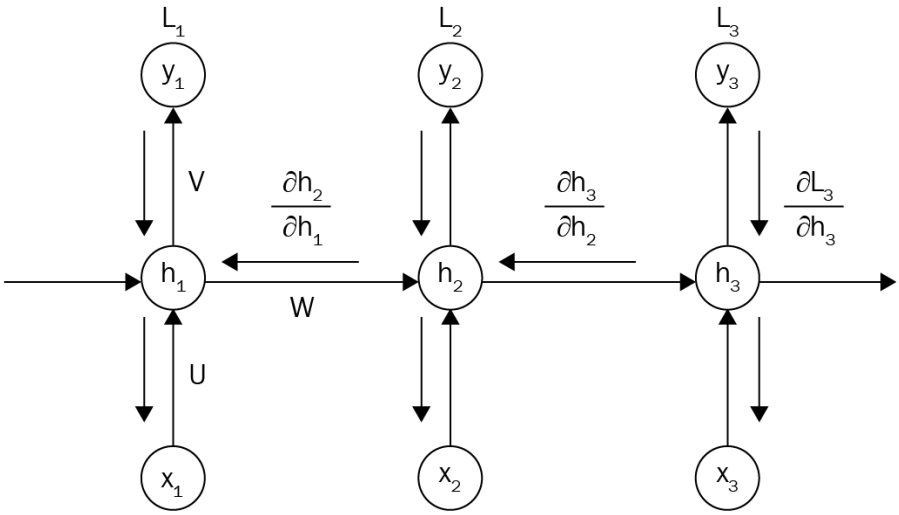
Loss function bertujuan untuk membandingkan nilai hasil prediksi dengan nilai target atau nilai yang sebenarnya [17]. Ada beberapa jenis *loss function* diantaranya *Mean Square Error* (MSE) dan *Cross Entropy* (CE). Namun dalam praktiknya, CE lebih cepat dalam konvergensi dan mendapatkan hasil lebih baik dalam tingkat kesalahan klasifikasi [17]. *Cross entropy* biasa digunakan pada *neural network* dan dalam kasus *multi class classification*. Pada penelitian ini loss function yang digunakan untuk menghitung nilai error hasil prediksi menggunakan *Cross Entropy* (CE). Persamaan *Cross Entropy* dapat dilihat pada persamaan (2.7).

$$L_{CE}(\hat{y}, y) = -\sum_{i=1} y_i log \hat{y}_i \quad \textbf{(2.7)}$$

2.4.4 *Backpropagation Through Time* (BPTT)

Backpropagation through time merupakan algoritma yang biasa digunakan untuk memperbaharui bobot pada Recurrent Neural Network. *Pada Recurrent Neural Network* nilai error dapat di *backpropagate* lebih jauh daripada neural network biasa. Prinsip dasar dari BPTT adalah *unfolding* [18]. Secara konseptual, BPTT bekerja dengan membuka gulungan (*unfolding*) setiap input layer pada setiap *timestep* (t), kemudian nilai error dihitung dan diakumulasikan untuk setiap *timestep* (t). Jaringan kemudian diputar kembali untuk memperbaharui bobot.

Pada pelatihan RNN ada 3 bobot yang akan diperbaharui dimana U, bobot dari *input layer* ke *hidden layer*, W, bobot dari *hidden layer* sebelumnya ke *hidden layer* selanjutnya, dan V, bobot dari *hidden layer* ke *output layer*.



Gambar 2.8 Backpropagation Through Time

Pada ERNN dalam melakukan BPTT akan menghitung turunan bobot U, W, dan V. Adapun rumus penurunan dari setiap bobot adalah sebagai berikut.

1. **Turunan bobot V**

Dalam menghitung turunan bobot V akan menggunakan rumus pada persamaan (2.8).

$$\frac{\partial E}{\partial V} = \sum_t \frac{\partial E_t}{\partial V} \quad \textbf{(2.8)}$$

$$\frac{\partial E_t}{\partial V} = (y_t - label_t) \otimes h_t \quad \textbf{(2.9)}$$

dimana :

$\frac{\partial E_t}{\partial V}$ = Turunan dari nilai error ke t terhadap bobot V

E_t = Nilai Error ke t

V = Bobot dari *hidden layer* ke *output layer*

y_t = Nilai dari *output layer* ke t

$label_t$ = vektor dari label sebenarnya ke t

t = *timestep* atau urutan token

2. Turunan bobot W

Dalam menghitung turunan bobot W dimana setiap *hidden state* h_t terhubung dengan *hidden state* sebelumnya $h_{(t-1)}$ dan begitu seterusnya hingga mencapai *hidden state* pertama h_1 yang terhubung dengan h_0 . Maka dalam penurunan bobot W harus memperhatikan seluruh *hidden state* . Rumus turunan bobot W ditunjukan persamaan (2.10).

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W} \quad \textbf{(2.10)}$$

$$\frac{\partial E_t}{\partial W} = \left(\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial W}\right) + \left(\frac{\partial E_t}{\partial h_{(t-1)}}\frac{\partial h_{(t-1)}}{\partial W}\right) + \left(\frac{\partial E_t}{\partial h_{(t-2)}}\frac{\partial h_{(t-2)}}{\partial W}\right) + \cdots + \left(\frac{\partial E_t}{\partial h_1}\frac{\partial h_1}{\partial W}\right)$$

$$\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial W} = \delta_t \otimes h_{t-1} \quad \textbf{(2.11)}$$

dimana :

$\frac{\partial E_t}{\partial W}$ = Turunan dari nilai error ke t terhadap bobot W

E_t = Nilai Error ke t

W = Bobot dari *hidden layer* sebelumnya ke *hidden layer* sesudahnya

h_t = *Hidden state* ke t

t = *timestep* atau urutan token

Dimana δ_t didapat dari penurunan [18]. Rumus penurunan δ_t .

$$\delta_t = [V^T.(y_t - label_t)] \times (1 - {h_t}^2) \quad \textbf{(2.12)}$$

dimana:

V = bobot dari *hidden layer* ke *output layer*

y_t = Nilai dari *output layer* ke t

$label_t$ = vektor dari label sebenarnya ke t

h_t = *Hidden state* ke t

t = *timestep* atau urutan token

3. Turunan bobot U

Pada penurunan bobot U mempunyai kesamaan dengan penurunan bobot W. Letak perbedaannya jika bobot W melakukan penurunan terhadap h_t sedangkan bobot U melakukan penurunan terhadap x_t. Rumus turunan bobot W ditunjukan persamaan (2.13).

$$\frac{\partial E}{\partial U} = \sum_t \frac{\partial E_t}{\partial U} \quad \textbf{(2.13)}$$

$$\frac{\partial E_t}{\partial U} = \left(\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial U}\right) + \left(\frac{\partial E_t}{\partial h_{(t-1)}}\frac{\partial h_{(t-1)}}{\partial U}\right) + \left(\frac{\partial E_t}{\partial h_{(t-2)}}\frac{\partial h_{(t-2)}}{\partial U}\right) + \cdots + \left(\frac{\partial E_t}{\partial h_1}\frac{\partial h_1}{\partial U}\right)$$

$$\frac{\partial E_t}{\partial h_t}\frac{\partial h_t}{\partial U} = \delta_t \otimes x_t \quad \textbf{(2.14)}$$

dimana :

$\frac{\partial E_t}{\partial U}$ = Turunan dari nilai error ke t terhadap bobot U

E_t = Nilai Error ke t

U = Bobot dari input layer ke *hidden layer*

x_t = Input ke t

t = *timestep* atau urutan token

2.4.5 *Minibatch Stochastic Gradient Descent*

Algoritma *machine learning* atau *neural network* dapat dikatakan bagus jika dalam memprediksi memberikan sedikit kesalahan, untuk mengukur nilai kesalahan digunakan *loss function*, seperti *cross entropy*. Jika nilai loss function kecil kemungkinan dalam memprediksi mendapatkan performa yang baik. Untuk mendapatkan nilai *loss function* yang kecil tersebut dipengaruhi nilai bobot dimana dalam tahap pelatihan bobot akan terus diperbaharui untuk mendapatkan bobot yang optimal. Salah satu algoritma pelatihan adalah *Gradient Descent* (GD).

Gradient Descent adalah salah satu metode pelatihan dengan meminimalkan nilai error untuk mengukur keakuratan *neural network* dalam melakukan tugas yang diberikan . GD bertujuan untuk mengatasi nilai error (*loss*) yang tinggi dengan cara memperbarui parameter. Adapun rumus dari GD ditunjukan persamaan (2.15).

$$\theta_{(baru)} = \theta_{(lama)} - a \times \frac{\partial E(error)}{\partial \theta(bobot)} \quad \textbf{(2.15)}$$

dimana :

$\theta_{(baru)}$ = Nilai bobot yang baru

$\theta_{(lama)}$ = Nilai bobot yang lama

a = *learning rate* (laju pembelajaran)

$\frac{\partial E(error)}{\partial \theta(bobot)}$ = Nilai turunan antara nilai error dan bobot

Setelah melakukan pelatihan biasanya bobot yang optial didapatkan, dalam pelatihannya GD menggunakan seluruh data untuk mendapatkan bobot baru sedangkan *Stochastic* GD menggunakan satu buah data. Jika yang digunakan dalam pelatihan data yang digunakan sebagian dikenal juga dengan *Minibatch* SGD.

2.5 Permodelan Sistem

Permodelan sistem merupakan metode yang memungkinkan untuk memodelkan sistem pada aplikasi yang akan dibangun. Adapun permodelan yang digunakan pada penelitian tugas akhir ini.

2.5.1 Diagram Konteks

Diagram Konteks adalah sebuah diagram sederhana yang menggambarkan hubungan antara *entity* luar, masukan dan keluaran dari sistem [8]. Diagram konteks dimulai dengan penggambaran terminator, aliran data, aliran kontrol penyimpanan, dan proses tunggal yang menunjukan keseluruhan sistem. Diagram konteks merupakan level 0 atau level tertinggi dari DFD dimana diagram konteks akan menggambarkan seluruh input, proses dan output pada sistem.

2.5.2 *Data Flow Diagram*

DFD merupakan tampilan input, proses, dan output dari suatu sistem dimana objek data akan mengalir ke dalam sistem melalui input sistem dan ditransformasikan oleh elemen pemrosesan sistem kemudian data yang dihasilkan akan keluar dari sistem [8]. Objek data diwakili oleh panah berlabel dan tranformasi diwaliki oleh lingkaran. DFD digambarkan secara hierarki dimana model aliran data pertama (kadang disebut DFD level 0) mewakili sistem secara keseluruhan. Model aliran data selanjutnya memberikan masing-masing detail yang semakin meningkat. *Data Flow Diagram* (DFD) menggambarkan model logika dan mengekspresikan transformasi data dalam suatu sistem. Ini termasuk mekanisme untuk memodelkan aliran data dan mendukung dekomposisi untuk mengilustrasikan detail dari aliran data dan fungsi. *Data Flow Diagram* tidak dapat menyajikan informasi tentang urutan operasi. Oleh karena itu, ini bukan metode pemodelan proses atau prosedur.

DFD mencakup karakteristik berikut.

1. Mendukung analisis dan persyaratan tahap desain sistem.
2. Teknik diagram dengan anotasi.
3. Menggambarkan jaringan kegiatan / proses sistem target.

4. Memungkinkan untuk perilaku paralel dan asinkron.
5. Penyempurnaan bertahap melalui dekomposisi hirarki proses.

Ada empat simbol dasar yang digunakan untuk mewakili *Data Flow Diagram*.

1. Proses

   Suatu proses menerima input data dan menghasilkan output dengan konten atau formulir yang berbeda. Proses dapat sesederhana mengumpulkan data input dan menyimpan dalam database, atau bisa rumit seperti menghasilkan laporan yang berisi penjualan bulanan semua toko yang banyak.

2. *Data Flow*

   Data flow adalah jalur bagi data untuk berpindah dari satu bagian sistem informasi ke bagian lainnya. Aliran data dapat mewakili elemen data tunggal seperti ID Pelanggan atau dapat mewakili sekumpulan elemen data (atau struktur data).

3. *Data Store*

   Data store atau *data repository* digunakan dalam diagram aliran data untuk mewakili situasi ketika sistem harus menyimpan data karena satu atau lebih proses perlu menggunakan data yang disimpan di lain waktu.

4. *External Entity*

   Entitas eksternal adalah orang, departemen, organisasi luar, atau sistem informasi lain yang menyediakan data ke sistem atau menerima output dari sistem. Entitas eksternal adalah komponen di luar batas sistem informasi. Mereka mewakili bagaimana sistem informasi berinteraksi dengan dunia luar.

2.6 Bahasa Pemrograman

Bahasa pemrograman atau sering diistilahkan juga dengan bahasa komputer, adalah instruksi standar untuk memerintah komputer. Bahasa pemrograman ini merupakan suatu himpunan dari aturan sintaks dan semantik yang dipakai untuk mendefinisikan program komputer.

2.6.1 *Python*

Python adalah bahasa pemrograman interpretatif multiguna dengan filosofi perancangan yang berfokus pada tingkat keterbacaan kode. *Python* diklaim sebagai bahasa yang menggabungkan kapabilitas, kemampuan, dengan sintaksis kode yang sangat jelas, dan dilengkapi dengan fungsionalitas pustaka standar yang besar serta komprehensif. *Python* sering digunakan oleh banyak aplikasi pada *data science* karena *python* memiliki pustaka (*library*) untuk *data loading*, *visualization, statistics*, *natural language procesing*, *image processing* dan lainnya [19]. Pada penelitian ini *python* digunakan sebagai bahasa pemrograman dalam membangun sistem ABSA.

2.7 Pengujian Performa

Untuk mengukur seberapa bagus model agoritma yang telah dibuat maka dibutuhkan metode untuk mengukur performa algoritma . Berikut adalah penjelasan dari metode untuk mengukur performa algoritma dalam penelitian ini.

2.7.1 Akurasi

Akurasi salah satu metode yang digunakan untuk mengukur performa algoritma. Metode ini memberikan informasi mengenai jumlah persentase keberhasilan sebuah algoritma melakukan prediksi secara benar [20]. Adapun persamaan dari akurasi dapat dilihat pada persamaan (2.16)

$$Akurasi = \frac{Jumlah\ Label\ Benar}{Total\ Label} \quad \textbf{(2.16)}$$

2.7.2 *F1 Score*

Pada kasus klasifikasi *f1 score* bertujuan untuk mengetahui keseimbangan kinerja pada kelas minoritas dan mayoritas, oleh karena itu *f1 score* cocok untuk mengukur performa algoritma dalam hal data yang tidak seimbang [21]. Selain *f1 score* ada juga *precision* dan *recall*, precision adalah tingkat ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh

sistem sedangkan *recall* adalah tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi. Pada penelitian ini aspek yang berlabel bukan sentimen lebih banyak dari label positif dan label negatif. Adapun persamaan *f1 score* dapat dilihat pada persamaan (2.19).

$$recall = \frac{Jumlah\ Prediksi\ Benar\ Kelas\ X}{Total\ Kelas\ X} \quad \textbf{(2.17)}$$

$$precision = \frac{Jumlah\ Prediksi\ Benar\ Kelas\ X}{Total\ Prediksi\ terhadap\ Kelas\ X} \quad \textbf{(2.18)}$$

$$F_1\ score(X) = \left(2 \times \frac{(precision\ \times recall)}{(precision + recall)}\right) \quad \textbf{(2.19)}$$

$$weighted\ F_1\ score = \frac{1}{\sum_i |X_i|} \times \sum_i |X|_{\ 1}\ score(X_i) \quad \textbf{(2.20)}$$

dimana:

$|X_i|$: jumlah anggota kelas X_i