BAB 3 ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi pemaparan analisis terhadap masalah yang ingin diatasi dengan pendekatan yang digunakan. Gambaran besar yang dimaksud meliputi analisis masalah, kerangka pemikiran, *flowchart* proses global, pengambilan *datasets*, dan analisis manual yang dilakukan.

3.1 Analisis Masalah

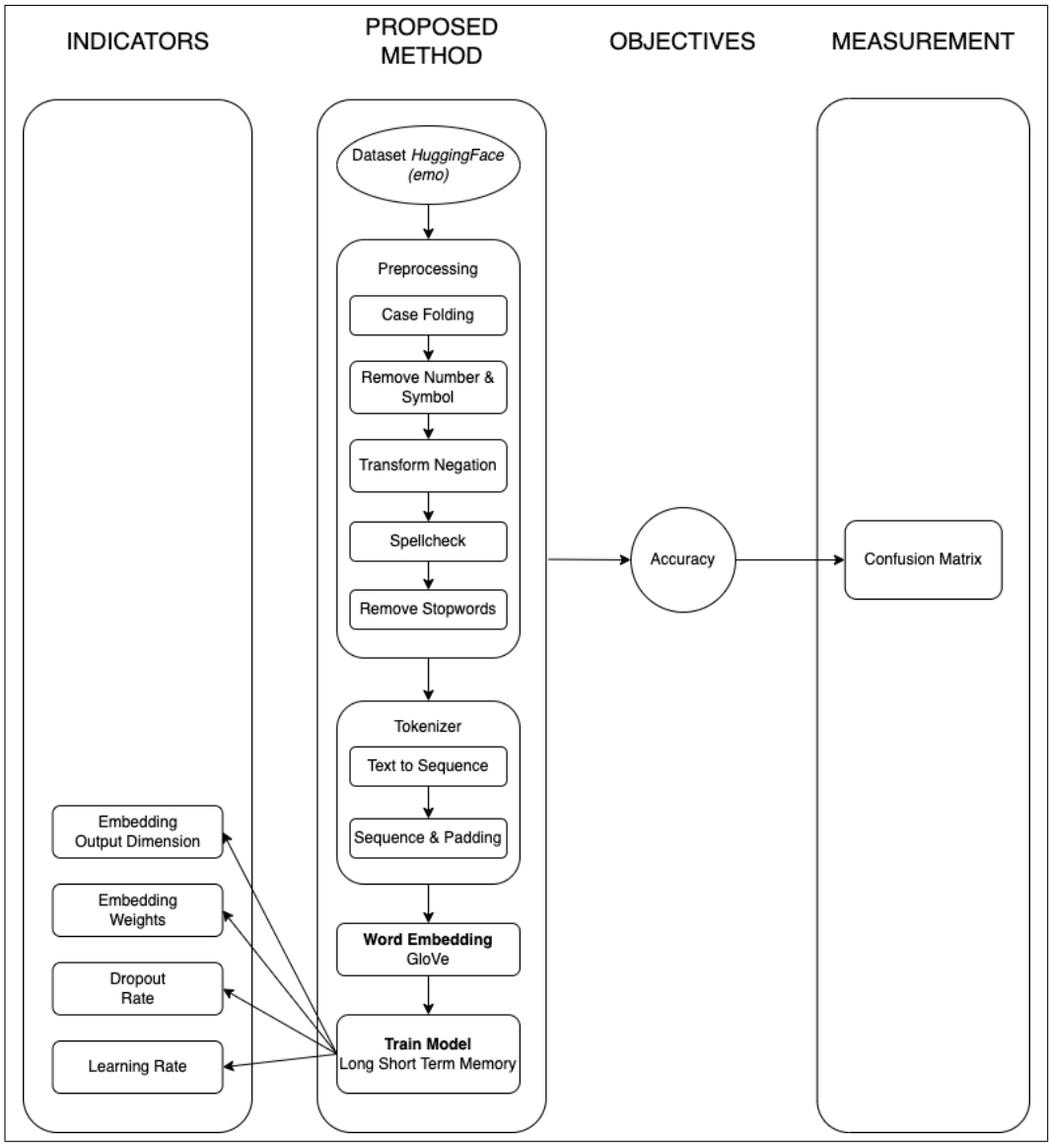
Pada bab 1 telah dijelaskan bahwa pada masa ini pendeteksian emosi diperlukan untuk meningkatkan kecerdasan sistem komputer karena emosi merupakan faktor penting dalam interaksi manusia komputer. Dengan pendeteksian emosi secara otomatis membantu sistem tutor cerdas, interaksi manusia komputer, analisis kepuasan pelanggan, sistem dialog, perilaku *blogger*, dan *enterprise computing*. Banyak penelitian saat ini menggunakan metode *machine learning* dan *deep learning* dalam melakukan pengenalan emosi. Penelitian menggunakan metode *deep learning* memiliki kesulitan dalam memperoleh data yang banyak, namun sejalan dengan pertumbuhan penggunaan internet menyebabkan jumlah data teks yang tersedia semakin banyak sehingga penelitian menggunakan metode *deep learning* diaplikasikan.

Pada penelitian ini, penulis membangun sebuah model LSTM untuk melakukan pengenalan emosi untuk data teks. Pada penelitian ini, penulis menilai akurasi dan mengidentifikasi parameter apa saja yang berpengaruh dalam penerapan algoritme LSTM pada pengenalan emosi untuk data teks. Pada penelitian ini, penulis juga menjawab beberapa masalah yang ada, seperti:

1. Bagaimana melakukan *preprocessing* data untuk data teks?
2. Bagaimana melakukan *tokenizer* pada data teks?
3. Bagaimana penggunaan *word embedding* terhadap model?
4. Bagaimana pengaruh parameter yang digunakan bersamaan ketika membuat model?
5. Bagaimana mendapatkan nilai akurasi terhadap model yang dibuat?

3.2 Kerangka Pemikiran

Berikut ini adalah kerangka pemikiran dari metode yang diusulkan untuk membangun model pengenalan emosi untuk data teks.



Gambar 3.1 Kerangka Pemikiran

Berikut adalah rangkaian penjelasan setiap bagian kerangka pemikiran pada gambar 3.1:

1. *Indicators* adalah parameter yang mempengaruhi hasil dari metode yang diajukan. Pada model yang dibuat terdapat sebuah indikator, yaitu *Embedding Output Dimension, Embedding Weights*, *Dropout Rate* dan *Learning Rate*.

   (a) *Embedding Output Dimension* merupakan penentuan dimensi vektor dari *embedding layer*. Indikator ini harus disesuaikan dengan jumlah vektor dalam *word embedding* agar proses pelatihan model mengeluarkan hasil yang tepat.

(b) *Embedding Weights* merupakan bobot yang dimasukan untuk representasi dari kata. Indikator ini disesuaikan dengan jumlah vektor dalam *word embedding* agar proses pelatihan model mengeluarkan hasil yang tepat.

(c) *Dropout Rate* merupakan penentuan *neuron* yang dipilih secara acak dan tidak dipakai / dibuang selama pelatihan. Indikator ini digunakan untuk mencegah terjadinya *overfitting* dan juga mempercepat proses pembelajaran.

(d) *Learning Rate* merupakan salah satu parameter *training* untuk menghitung nilai koreksi bobot pada waktu proses *training*. Nilai *learning rate* ini berada pada range nol (0) sampai satu (1). Semakin besar nilai *learning rate*, maka proses *training* berjalan semakin cepat.

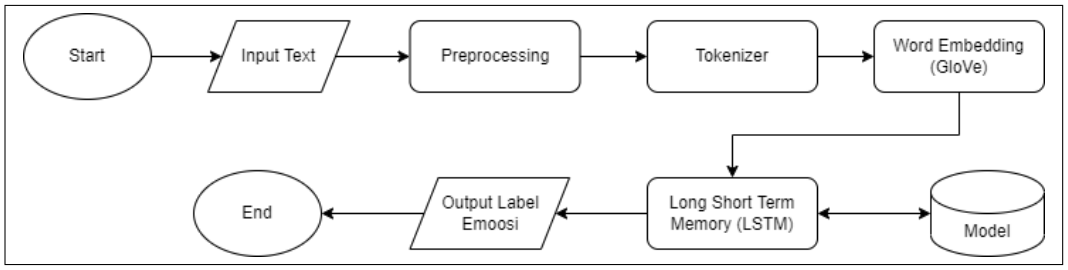
2. *Proposed Method* adalah bagian yang menjelaskan proses dari awal hingga akhir dalam penelitian ini. Proses pertama yang dilalui adalah tahap *preprocessing* yang didalamnya terdapat proses perubahan pada teks, dimana dilakukan proses untuk melakukan *case folding*, menghilangkan angka, menghilangkan simbol, mengubah kata negasi, melakukan *spellcheck*, dan menghilangkan *stopwords*. Kemudian data hasil *preprocessing* digunakan untuk melakukan *tokenizer* untuk melakukan tokenisasi, *text to sequence*, dan *sequence & padding*. Lalu dilakukan tahap *word embedding*, dimana dalam tahap ini *word embedding* dihasilkan menggunakan GloVe. Setelah itu hasil dari proses *tokenizer* dan *word embedding* digunakan untuk melatih model pengenalan emosi yang dibangun.

3. *Objectives* adalah bagian yang menjelaskan mengenai hal-hal yang menjadi acuan pengukuran dan pengukuran yang digunakan adalah akurasi model.

4. *Measurement* adalah satuan ukur pada bagian *Objectives*, seperti perhitungan *confusion matrix*.

3.3 Urutan Proses Global

Berikut ini adalah urutan proses global dari metode yang diusulkan untuk membangun pengenalan emosi untuk data teks.



Gambar 3.2 *Flowchart Emotion Detection in Text*

Berikut adalah rangkaian penjelasan urutan proses global pada gambar 3.2:

1. *Dataset* yang digunakan merupakan data teks dengan label emosinya yang diambil dari *website HuggingFace* dengan data bernama emo.

2. Setiap data teks melalui tahap *preprocessing*, dimana dalam tahap ini dilakukan proses *case folding*, menghilangkan angka, menghilangkan simbol, mengubah kata negasi, melakukan *spellcheck*, dan menghilangkan *stopwords* untuk mendapatkan data bersih.

3. Setelah data teks melalui tahap *preprocessing*, dilakukan tahap *tokenizer* untuk melakukan tokenisasi, *text to sequence*, dan *sequence & padding*. Hal ini dilakukan agar data teks beserta labelnya diproses oleh mesin.

4. *Word embedding* di *generate* untuk disiapkan dimasukkan ke dalam *embedding layer*. Digunakannya data *pretrained* GloVe untuk menghasilkan *word embedding*.

5. Setelah dilakukan proses *tokenizer* dan menghasilkan *word embedding*, dibuat sebuah model LSTM, dimana dalam model LSTM ini dibuat menggunakan 4 *layer*, yaitu:

   (a) *Embedding Layer* = Dalam *embedding layer* ini digunakan 5 parameter, yaitu: *input_dim, output_dim, input_length, weights*, dan *trainable*.

   (b) LSTM*Layer* = Dalam LSTM*layer* digunakan 1 parameter, yaitu: *units*.

   (c) *Dropout Layer* = Dalam *dropout layer* digunakan 1 parameter, yaitu: *rate*.

   (d) *Dense Layer* = Dalam *dense layer* digunakan 2 parameter, yaitu: *units* dan *activation*.

Dengan tambahan *categorical_crossentropy* sebagai *loss function* dan *adam* sebagai *optimizer* yang dibuat dalam model ketika melakukan *compile*, serta penggunaan *validation_split* dan *epoch* ketika melatih model.

6. Setelah model dibuat, maka dilakukan pengenalan emosi menggunakan model LSTM tersebut dan dikategorikan ke dalam 4 macam, yaitu:

   (a) *Others* (0)
   (b) *Happy* (1)
   (c) *Sad* (2)
   (d) *Angry* (3)

3.4 Analisis Manual

Dalam sub bab ini dijelaskan proses secara bertahap dengan menggunakan perhitungan manual yang ada dalam penelitian ini. Penjelasan dari proses-proses tersebut dijelaskan sebagai berikut.

3.4.1 Pengambilan *Dataset*

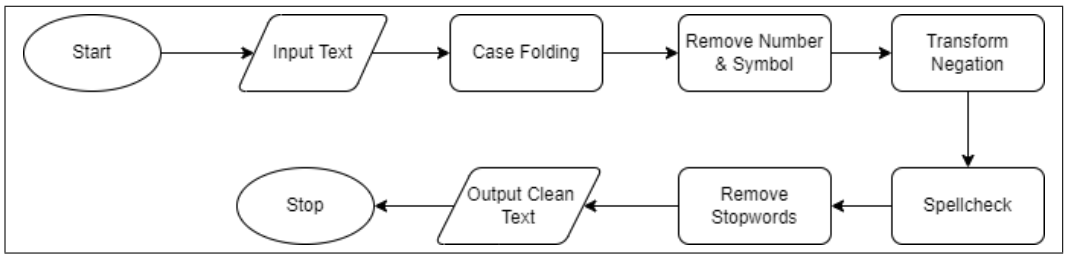
Dalam penelitian ini menggunakan satu dataset berupa teks berbahasa Inggris yang sudah diberi label emosi dalam bentuk angka yang diambil dari *website Huggingface*, yang dimana dalam *dataset* ini terdiri dari jumlah data *training* sebanyak 30160 dan jumlah data *testing* sebanyak 5509. Dataset ini didapatkan dengan menggunakan fungsi *load_dataset()* yang diimport dari sebuah *library* bernama *datasets*.

Terlihat contoh input text berupa data dalam teks dan label emosi yang sudah dikategorikan menggunakan angka yang tampak pada gambar 2.6. Kategori tersebut dibedakan menjadi 4 macam, yaitu *others, sad, happy*, dan *angry*. Kemudian *input* teks tersebut diproses ke dalam tahap *preprocessing* untuk mendapatkan bentuk data yang diproses ketika melakukan *training* dan *testing* dalam model yang dibuat.

3.4.2 *Preprocessing*

Pada tahap ini dijelaskan proses pembersihan kata dalam teks. Setiap kata dalam teks dalam *input* teks dilakukan proses pembersihan sehingga digunakan dalam pembuatan model. Proses ini meliputi pembersihan kata dengan menggunakan *case folding, remove number, remove symbol, transform negation, spellcheck*, dan

remove stopwords. Berikut adalah langkah-langkah pada saat proses *preprocessing*.



Gambar 3.3 *Flowchart Preprocessing*

Berikut adalah contoh teks yang digunakan dalam analisa manual.

Tabel 3.1 Contoh Teks

Teks	**Label**
I don't like when U come here 2 days ago!	*Sad* (2)

3.4.2.1 *Case Folding*

Pada tahap ini dilakukan *case folding*, dimana *case folding* adalah salah satu bentuk *text preprocessing* yang paling sederhana dan efektif. Tujuan dilakukan *case folding* adalah untuk mengubah semua huruf dalam dokumen menjadi huruf kecil. Hanya huruf 'a' sampai 'z' yang diterima. Hal ini dilakukan untuk memudahkan memproses data teks di tahap *preprocessing* ini. Pada tabel 3.2, terlihat contoh dari perubahan perubahan data teks setelah dilakukan *case folding*.

Tabel 3.2 Contoh Proses *Case Folding* pada Teks

Sebelum	**Sesudah**
I don't like when U come here 2 days ago!	*i don't like when u come here 2 days ago!*

3.4.2.2 *Remove Number & Symbol*

Pada tahap ini angka dan simbol dihilangkan dalam teks. Tujuan penghapusan angka dan simbol ini dikarenakan tidak ada dampaknya terhadap pengenalan emosi yang dilakukan sehingga tahap ini digunakan untuk memudahkan memproses data teks saat melakukan pelatihan pada model. Pada tabel 3.3, terlihat contoh dari perubahan data teks setelah angka dan simbol dihilangkan.

Tabel 3.3 Contoh *Case Folding* pada Teks

Sebelum	**Sesudah**
i don't like when u come here 2 days ago!	*i don't like when u come here days ago*

3.4.2.3 *Transform Negation*

Pada tahap ini dilakukan *transform negation*, *negation* adalah mekanisme yang mengubah argumen positif menjadi penolakan terbalik. Secara khusus, dalam tugas analisis afektif, negasi memainkan peran penting karena kata-kata negasi mempengaruhi polaritas kata atau kalimat yang menyebabkan polaritas terbalik dalam banyak kasus. Sebelum dilakukan *transform negation* kata-kata seperti "*don't*", "*doesn't*", "*didn't*", dll. harus diubah menjadi "*do not*", "*does not*", "*did not*", dll. Setelah kata-kata negasi itu diubah, maka baru bisa dilakukan *transform negation*. Tujuan dilakukan *transform negation* adalah menggantikan kata yang diawali "*not*" menjadi kata antonimnya. Pada tabel 3.4, terlihat contoh dari penjabaran kata negasi dan pada tabel 3.5, terlihat contoh dari penggunaan *transform negation*.

Tabel 3.4 Contoh Penjabaran Kata Negasi pada Teks

Sebelum	**Sesudah**
i don't like when u come here days ago	*i do not like when u come here days ago*

Tabel 3.5 Contoh *Transform Negation* pada Teks

Sebelum	**Sesudah**
i do not like when u come here days ago	*i do unlike when u come here days ago*

3.4.2.4 *Spellcheck*

Pada tahap ini dilakukan *spellcheck*, dimana proses *spellcheck* adalah mengidentifikasi kata-kata yang mungkin salah eja dan memperbaikinya. Tujuan dilakukan *spellcheck* adalah untuk mengoreksi salah eja dan kesalahan ketik atau membiarkannya seperti dengan asumsi mereka mewakili teks bahasa alami dan kompleksitas yang terkait. Pada tabel 3.6, terlihat contoh dari penggunaan *spellcheck*.

Tabel 3.6 Contoh *Spellcheck* pada Teks

Sebelum	**Sesudah**
i do unlike when u come here days ago	*i do unlike when you come here days ago*

3.4.2.5 *Remove Stopwords*

Pada tahap ini dilakukan *remove stopwords*, dimana *stopwords* adalah menghasilkan kata-kata yang paling umum. Tujuan dilakukannya *remove stopwords* adalah untuk menghilangkan kata-kata umum sehingga ketika melakukan pelatihan pada model diproses lebih cepat. Pada tabel 3.7, terlihat contoh dari penggunaan *remove stopwords*.

Tabel 3.7 Contoh *Stopwords* pada Teks

Sebelum	**Sesudah**
i do unlike when you come here days ago	*unlike days ago*

3.4.3 *Tokenizer*

Pada tahap ini dijelaskan proses *tokenizer* dalam teks. Setiap teks yang sudah melakukan tahap pembersihan dilakukan *tokenizer* untuk membagi teks yang berupa kalimat, paragraf atau dokumen, menjadi token-token/bagian-bagian tertentu seperti pada tabel 3.8. Setelah dibuat menjadi token-token, setiap token dari teks dan label yang diberikan pada teks ini dilakukan proses ke tahap *text to sequence* dan *sequence & padding*.

Tabel 3.8 Contoh Tokenisasi pada Teks

Sebelum	**Sesudah**
unlike days ago	[*"unlike", "days", "ago"*]

3.4.3.1 *Text to Sequence*

Pada tahap ini, teks yang telah dilakukan tokenisasi diubah menjadi urutan kata yang dijadikan sebagai indeks kata. Kumpulan kata tersebut memiliki indeks yang unik, sehingga antar kata berbeda indeksnya. Indeks mewakili sebuah kata, sehingga teks berisikan kata yang diubah menjadi urutan indeks yang mewakili kata. Berikut adalah tabel 3.9 sebelum dan sesudah dilakukan *text to sequence*.

Tabel 3.9 Contoh *Text to Sequence* pada Teks

Sebelum	**Sesudah**
[*"unlike", "days", "ago"*]	[0, 1, 2]

3.4.3.2 *Padding & Seqeunce*

Pada tahap ini, dilakukan *padding* yaitu dengan menyesuaikan bentuk panjang *sequence* yang beragam menjadi panjang yang sama. Pada contoh ini, digunakannya panjang sequence sebesar 10 dan sisa *sequence* dijadikan 0. Berikut tabel 3.10 sebelum dan sesudah penerapan tahap padding.

Tabel 3.10 Contoh *Padding & Sequence*

Sebelum	**Sesudah**
[0, 1, 2]	[0, 0, 0, 0, 0, 0, 0, 0, 1, 2]

3.4.4 *Word Embedding* GloVe

Pada tahap ini, dilakukan pemetaan vektor pada setiap kata yang ada pada indeks kata. Indeks berisikan angka yang mewakili sebuah kata dan setiap indeks memiliki kata yang unik. Digunakannya *word embedding* GloVe untuk penelitian ini dan dengan dimensi 50 untuk setiap vektor katanya. Proses dilakukan dengan melihat indeks terlebih dahulu, kemudian melihat kata yang ada. Dari kata tersebut, dilihat pada *pretrained* GloVe, apakah kata tersebut ada dalam GloVe?. Jika ada, diambil vektor yang ada. Kemudian vektor tersebut disimpan dalam *array embedding* berukuran [total indeks kata x 50] untuk menyimpan vektor yang sesuai dengan yang ada pada indeks kumpulan kata. Berikut contoh tabel 3.11 representasi kata dan vektor yang diambil dari GloVe dari 3 indeks kata.

Tabel 3.11 Contoh *Word Embedding*

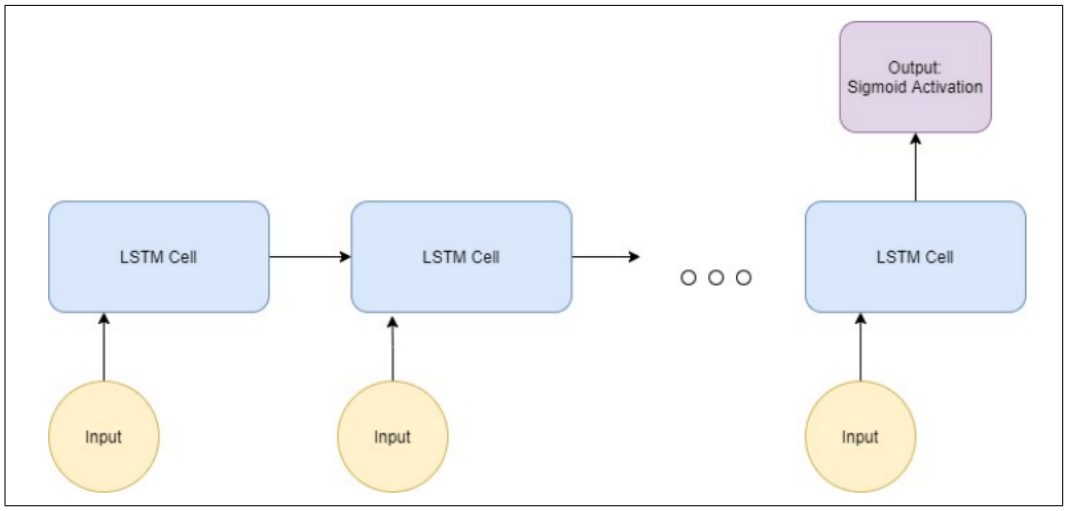
Indeks	**Kata**	**Vektor**
0	*unlike*	[0.40351, 0.25297, 0.058938, 0.016263, -0.16723, 0.49866, -0.46665, -0.39667, -0.42482, 0.22456, 0.17972, 0.31613, 0.17297, 0.25262, 0.21552, 0.4215, 0.092249, 0.26599, -0.079251, -0.50896, -0.30306, -0.42921, 0.1109, 0.4603, -0.072813, -1.1166, -0.6532, -0.16932, 0.0043999, -0.035547, 2.3802, -0.17461, 0.0065762, -0.57749, 0.17811, 0.080492, -0.087587, 0.072499, -0.75795, -0.15779, -0.15288, 0.24441, 0.076898, 0.52795, 0.16373, 0.37204, -0.21652, -0.3913, -0.56267, 0.10644]

Tabel 3.11 Contoh *Word Embedding* (Lanjutan)

Indeks	**Kata**	**Vektor**
1	*days*	[0.62236, 0.1973, 0.0023326, -0.44924, 0.084905, -0.33628, -0.90642, 0.62154, -0.20657, -0.20043, -0.6269, -0.64429, -0.31616, 0.23314, 1.2348, -0.080113, -0.7382, -0.3497, -0.97589, -0.046964, 0.29777, 0.68288, 0.87876, -0.28921, 0.1456, -1.411, -0.11301, 0.2132, 0.49548, 0.38685, 3.397, 0.40639, -0.54343, -0.00037414, 0.44268, -0.1015, 0.37693, 0.047623, 0.044414, 0.10038, -0.99794, 0.2082, 0.063226, 0.12436, -0.059387, 0.046682, -0.22335, -0.22082, -0.40414, -0.18502]
2	*ago*	[0.48384, 0.036723, 0.3554, -0.11934, 0.13925, 0.089866, -1.2636, -0.027009, -0.52288, -0.059505, -0.38398, -0.80808, -0.65736, -0.24665, 1.2932, 0.20113, -0.31522, -0.14376, -0.79002, 0.2567, -0.059354, 0.43529, 0.2418, -0.62907, -0.037233, -1.7944, -0.16567, 0.0054654, -0.056066, 0.19838, 2.9616, -0.11965, -0.16888, -0.035133, 0.024751, -0.41839, -0.21409, 0.41672, 0.16242, -0.2483, -0.90225, 0.15443, 0.17426, 0.21366, 0.10285, 0.096885, -0.56849, -0.21438, -0.68166, -0.3792]

3.4.5 *Long Short Term Memory* (LSTM)

Pada tahap pemodelan, digunakannya metode LSTM untuk membuat model belajar. Analisa secara manual dijelaskan dengan 2 *timestep*, yaitu *input* kata pertama sebagai *timestep* pertama dan kata kedua sebagai *timestep* kedua dari contoh kata. Kata yang digunakan yaitu ”*unlike*” dan ”*days*” dan digunakannya 3 nilai vektor pertama dari 50 vektor. Nilai secara acak dari distribusi normal dengan *mean* adalah 0 dan standar deviasi adalah 1 untuk inisialisasi weight, nilai 0 untuk inisialisasi bias dan jumlah *unit hidden layer* sebanyak 3. Berikut gambar 3.4 ilustrasi LSTM yang digunakan dalam penghitungan manual.



Gambar 3.4 Ilustrasi LSTM Perhitungan Manual

3.4.6 *Timestep* Pertama

Pada *timestep* pertama dimulai dari kata *"unlike"* sebagai input dengan vektor yang dimilikinya adalah [0.40351, 0.25297, 0.058938] dan inisialisasi nilai *hidden state* sebelumnya dengan nilai 0 karena masih pada posisi perhitungan awal. Nilai vektor digunakan pada gerbang yang ada sebagai *input*.

3.4.6.1 *Forget Gate*

Penghitungan pertama dimulai dari *forget gate* menggunakan persamaan 2.1. Nilai W_f, U_f diinisialisasi dengan distribusi normal secara acak dan nilai b_f adalah 0. Berikut adalah penghitungannya untuk *forget gate timestep* pertama.

$$f_t = \sigma\left(\begin{bmatrix} -0.36658844 & -1.61106401 & 1.87203336 \\ -0.28343935 & 0.20231672 & -0.4251159 \\ 1.38593374 & -0.21863406 & 1.87334779 \end{bmatrix} \bullet \begin{bmatrix} 0.40351 \\ 0.25297 \\ 0.058938 \end{bmatrix} + \begin{bmatrix} 1.6002104 & -0.27069961 & -1.08463431 \\ 1.21455431 & 0.3184536 & -0.66009903 \\ 1.33881495 & 0.6323578 & -1.04369611 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right)$$

Dari penghitungan diatas, maka didapat matriks vektor hasil *forget gate* sebagai berikut.

$$f_t = \begin{bmatrix} 0.39051712 \\ 0.4779528 \\ 0.64893055 \end{bmatrix}$$

3.4.6.2 *Input Gate*

Setelah dilakukan penghitungan *forget gate*, penghitungan selanjutnya dilakukan untuk menghitung *input gate* dengan menggunakan persamaan 2.2. Nilai W_i, U_i diinisialisasi dengan distribusi normal secara acak dan nilai b_i adalah 0. Berikut adalah perhitungannya untuk *input gate timestep* pertama.

$$i_t = \sigma\left(\begin{bmatrix} 0.62298068 & 0.44012166 & 0.93798973 \\ 0.38306699 & -0.86468679 & -0.36847324 \\ -0.36003432 & -0.05322914 & -0.38808374 \end{bmatrix} \bullet \begin{bmatrix} 0.40351 \\ 0.25297 \\ 0.058938 \end{bmatrix} + \begin{bmatrix} 0.18626383 & -0.37952795 & -0.92415426 \\ 0.21628096 & -0.45263405 & -0.7327017 \\ 0.56757731 & 0.29866377 & -0.64779853 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *input gate* sebagai berikut.

$$i_t = \begin{bmatrix} 0.60300451 \\ 0.47854181 \\ 0.45472046 \end{bmatrix}$$

Kemudian dilanjutkan dengan perhitungan untuk *candidate state*. Hal yang membedakan pada penghitugan untuk *candidate state* adalah digunakannya *hyperbolic tangent* sebagai fungsi aktivasi. Persamaan 2.3 digunakan dalam penghitungan *candidate state*. Berikut perhitungan untuk *candidate state*.

$$\tilde{C}_t = \sigma\left(\begin{bmatrix} 1.36162997 & -0.98190943 & -0.85895997 \\ 0.30978415 & -0.14687127 & -3.30831318 \\ 1.23276443 & 0.04983586 & 1.30341836 \end{bmatrix} \bullet \begin{bmatrix} 0.40351 \\ 0.25297 \\ 0.058938 \end{bmatrix} + \begin{bmatrix} 0.23412451 & -1.16006662 & 1.7361476 \\ -3.3169362 & -1.62590328 & -1.96597113 \\ -1.25348511 & -0.93574191 & 0.10017843 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *candidate state* sebagai berikut.

$$\tilde{C}_t = \begin{bmatrix} 0.24530619 \\ -0.10673032 \\ 0.52763398 \end{bmatrix}$$

3.4.6.3 *Output Gate*

Perhitungan selanjutnya yaitu *output gate*. *Output gate* menentukan nilai untuk *hidden state* selanjutnya dengan menggunakan persamaan 2.4. Nilai W_o, U_o diinisialisasi dengan distribusi normal secara acak dan nilai b_o adalah 0. Berikut adalah perhitungannya untuk *output gate*.

$$o_t = \sigma\left(\begin{bmatrix} -0.06649506 & -1.3413727 & -1.10056287 \\ 0.89552448 & 0.72447281 & -2.15472494 \\ 1.59390351 & -0.37055401 & 0.38455522 \end{bmatrix} \bullet \begin{bmatrix} 0.40351 \\ 0.25297 \\ 0.058938 \end{bmatrix} + \begin{bmatrix} -1.86619525 & -0.49338088 & -0.49403357 \\ -1.03749413 & 1.62050985 & 1.13333811 \\ -0.76364195 & -0.44071865 & 0.73512484 \end{bmatrix} \bullet \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *output gate* sebagai

berikut.

$$o_t = \begin{bmatrix} 0.39388197 \\ 0.60291546 \\ 0.63924342 \end{bmatrix}$$

3.4.6.4 *Cell State*

Perhitungan selanjutnya adalah memperbaharui nilai *memory* yang ada pada *cell state*. Pada *timestep* pertama, nilai *cell state* sebelumnya tidak ada karena masih penghitungan awal, maka diinisialisasikan dengan nilai 0. Persamaan 2.5 digunakan untuk perhitungan *cell state*. Berikut perhitungan *cell state*.

$$C_t = \begin{bmatrix} 0.39051712 \\ 0.4779528 \\ 0.64893055 \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0.60300451 \\ 0.47854181 \\ 0.45472046 \end{bmatrix} \odot \begin{bmatrix} 0.24530619 \\ -0.10673032 \\ 0.52763398 \end{bmatrix}$$

Dari perhitungan diatas, maka didapat nilai *cell state* yang telah diperbaharui, nilai ini diteruskan untuk *cell* selanjutnya dan untuk menentukan nilai *hidden state*. Berikut hasil penghitungan dari *cell state*

$$C_t = \begin{bmatrix} 0.14792074 \\ -0.05107492 \\ 0.23992597 \end{bmatrix}$$

3.4.6.5 *Hidden State*

Perhitungan terakhir adalah menentukan nilai *hidden state*. Nilai *hidden state* digunakan untuk penghitungan pada *cell* selanjutnya. Dengan menggunakan persamaan 2.6, didapat nilai *hidden state*. Berikut perhitungan untuk *hidden state*.

$$h_t = \begin{bmatrix} 0.39388197 \\ 0.60291546 \\ 0.63924342 \end{bmatrix} \odot \sigma_h \left(\begin{bmatrix} 0.14792074 \\ -0.05107492 \\ 0.23992597 \end{bmatrix} \right)$$

Dari perhitungan terakhir diatas, maka didapat nilai *hidden state* dari sebuah *cell*. Berikut hasil dari perhitungan untuk nilai *hidden state*.

$$h_t = \begin{bmatrix} 0.05784205 \\ -0.03076711 \\ 0.15049441 \end{bmatrix}$$

3.4.7 *Timestep* Kedua

Setelah didapatkannya nilai *hidden state* pada *timestep* pertama, maka dilanjutkan dengan *input* menggunakan kata kedua. Contoh yang digunakan adalah kata *"days"* yang memiliki vektor [0.62236, 0.1973, 0.0023326]. Urutan penghitungan sama dengan *timestep* sebelumnya, dengan menggunakan nilai *hidden state* dan *cell state* yang telah diperbaharui sebelumnya.

3.4.7.1 *Forget Gate*

Perhitungan dimulai dengan menggunakan persamaan 2.1 untuk mendapatkan nilai *forget gate*. *Input* yang dimasukan adalah vektor kata dan nilai *hidden state* pada *timestep* pertama. Berikut penghitungan untuk *forget gate* pada *timestep* kedua.

$$f_t = \sigma \left(\begin{bmatrix} -0.04913193 & -0.23109881 & -0.60465429 \\ 1.23493577 & 2.17908225 & 0.44119263 \\ -0.68026136 & 0.15091 & -0.22769326 \end{bmatrix} \bullet \begin{bmatrix} 0.62236 \\ 0.1973 \\ 0.0023326 \end{bmatrix} + \begin{bmatrix} -0.16674495 & -0.7063736 & -0.47867191 \\ 1.26108418 & 0.28889196 & 0.04094385 \\ 0.41756866 & 1.79028567 & 0.93253053 \end{bmatrix} \bullet \begin{bmatrix} 0.05784205 \\ -0.03076711 \\ 0.15049441 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *forget gate* sebagai berikut.

$$f_t = \begin{bmatrix} 0.46567079 \\ 0.7807006 \\ 0.42929887 \end{bmatrix}$$

3.4.7.2 *Input Gate*

Kemudian dilakukannya perhitungan untuk *input gate* dengan menggunakan persamaan 2.2. Berikut perhitungan untuk *input gate* pada *timestep* kedua.

$$i_t = \sigma\left(\begin{bmatrix} 1.46662098 & -1.18835177 & -1.1066311 \\ 0.97978437 & -0.67817222 & -1.67490876 \\ 0.73719701 & -0.37583658 & 0.58606196 \end{bmatrix} \bullet \begin{bmatrix} 0.62236 \\ 0.1973 \\ 0.0023326 \end{bmatrix} + \begin{bmatrix} -0.75156608 & -0.24255579 & 0.9542411 \\ -0.51737153 & 1.13295774 & -1.10393987 \\ -0.43572849 & 0.37843361 & 1.54677234 \end{bmatrix} \bullet \begin{bmatrix} 0.05784205 \\ -0.03076711 \\ 0.15049441 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *input gate* sebagai berikut.

$$i_t = \begin{bmatrix} 0.68639557 \\ 0.55999651 \\ 0.64151604 \end{bmatrix}$$

Perhitungan selanjutnya adalah menemukan nilai *candidate state* dengan menggunakan persamaan 2.3. Berikut penghitungan untuk *candidate state* pada *timestep* kedua.

$$\tilde{C}_t = \sigma\left(\begin{bmatrix} -0.00480373 & 0.65676634 & 0.95335957 \\ 1.77298325 & -1.00736161 & 1.30444274 \\ 0.39506239 & -0.08948894 & 1.28120212 \end{bmatrix} \bullet \begin{bmatrix} 0.62236 \\ 0.1973 \\ 0.0023326 \end{bmatrix} + \begin{bmatrix} 1.003646 & -1.16106992 & -0.59164454 \\ 1.96547564 & 1.10985581 & 1.87020428 \\ 0.38294438 & -0.2390663 & 1.00793169 \end{bmatrix} \bullet \begin{bmatrix} 0.05784205 \\ -0.03076711 \\ 0.15049441 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *candidate state* sebagai berikut.

$$\tilde{C}_t = \begin{bmatrix} 0.13276231 \\ 0.85345023 \\ 0.39050617 \end{bmatrix}$$

3.4.7.3 *Output Gate*

Perhitungan selanjutnya adalah *output gate* yang menggunakan persamaan 2.4 untuk menentukan nilai *hidden state* selanjutnya. Berikut perhitungan untuk *output gate*.

$$o_t = \sigma\left(\begin{bmatrix} 1.00549259 & 1.76712501 & -1.27702349 \\ -0.96922206 & -0.33626641 & 1.14818098 \\ 0.78386791 & -1.88373285 & -0.53568864 \end{bmatrix} \bullet \begin{bmatrix} 0.62236 \\ 0.1973 \\ 0.0023326 \end{bmatrix} + \begin{bmatrix} 0.0702497 & -0.52573271 & -0.2914909 \\ 0.81404842 & -0.22791995 & 0.924764 \\ 1.05096884 & 1.40161185 & -0.45755878 \end{bmatrix} \bullet \begin{bmatrix} 0.05784205 \\ -0.03076711 \\ 0.15049441 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}\right)$$

Dari perhitungan diatas, maka didapat matriks vektor hasil *output gate* sebagai berikut.

$$o_t = \begin{bmatrix} 0.72067741 \\ 0.38376413 \\ 0.51593076 \end{bmatrix}$$

3.4.7.4 *Cell State*

Perhitungan selanjutnya adalah memberikan pembaharuan untuk nilai *memory cell* sebelumnya dengan menggunakan nilai *forget gate*, *input gate* dan *candidate gate* pada *timestep* kedua. Persamaan 2.5 digunakan untuk memperbaharui nilai *cell state*. Berikut penghitungan untuk *cell state* pada *timestep* kedua.

$$C_t = \begin{bmatrix} 0.46567079 \\ 0.7807006 \\ 0.42929887 \end{bmatrix} \odot \begin{bmatrix} 0.14792074 \\ -0.05107492 \\ 0.23992597 \end{bmatrix} + \begin{bmatrix} 0.68639557 \\ 0.55999651 \\ 0.64151604 \end{bmatrix} \odot \begin{bmatrix} 0.13276231 \\ 0.85345023 \\ 0.39050617 \end{bmatrix}$$

Dari perhitungan diatas, maka didapat nilai *cell state* yang telah diperbaharui, nilai ini diteruskan untuk *cell* selanjutnya dan untuk menentukan nilai *hidden state*. Berikut hasil penghitungan dari *cell state*

$$C_t = \begin{bmatrix} 0.16000983 \\ 0.43805493 \\ 0.35351592 \end{bmatrix}$$

3.4.7.5 *Hidden State*

Perhitungan selanjutnya adalah menentukan nilai *hidden state*. Nilai *hidden state* didapatkan dengan menggunakan persamaan 2.6.Berikut penghitungan untuk *hidden state*.

$$h_t = \begin{bmatrix} 0.72067741 \\ 0.38376413 \\ 0.51593076 \end{bmatrix} \odot \sigma_h \left(\begin{bmatrix} 0.16000983 \\ 0.43805493 \\ 0.35351592 \end{bmatrix} \right)$$

Dari perhitungan terakhir diatas, maka didapat nilai *hidden state* dari sebuah cell. Berikut hasil dari penghitungan untuk nilai *hidden state*.

$$h_t = \begin{bmatrix} 0.1143413 \\ 0.15812267 \\ 0.1751533 \end{bmatrix}$$

Proses untuk *timestep* ketiga hingga nilai yang ditentukan memiliki penghitungan yang sama. Setelah penghitungan selesai, nilai *hidden state* yang dihasilkan oleh LSTM dilanjutkan perhitungannya dengan menggunakan *neural network*. Setelah dilakukan perhitungan LSTM, dilakukan perhitungan pada *dropout layer*.

3.4.8 *Dropout Layer*

Hasil yang sudah dihitung melalui *layer* LSTM, nilai *hidden neuron* dikeluarkan pada *cell* terakhir pada LSTM, dan nilai tersebut dilakukan perhitungan pada *Dropout layer*, dengan menggunakan persamaan 2.7. Hal ini dilakukan untuk menonaktifkan beberapa nilai fitur agar *neural network* yang dibangun tidak terlalu banyak belajar. Parameter probabilitas yang digunakan adalah 0.5, sehingga kemungkinan menghapus setiap fitur adalah 0.5. Berikut adalah contoh perhitungan proses *dropout*.

$$\begin{aligned} O_i &= O_i \times \frac{1}{p} \\ &= 0.1143413 \times \frac{1}{0.5} \\ &= 0.2286826 \end{aligned}$$

Dari perhitungan diatas, maka didapat nilai hasil perhitungan *dropout*. Berikut

hasil dari penghitungan untuk nilai *dropout*.

$$Output\ dropout = \begin{bmatrix} 0.2286826 \\ 0.31624534 \\ 0.3503066 \end{bmatrix}$$

3.4.9 *Dense Layer*

Pada lapisan *dense*, dilakukan perhitungan sesuai dengan persamaan 2.8. Jumlah kelas pada contoh perhitungan manual ini adalah 4 sehingga menggunakan *dense layer* sebanyak 4 unit. Hasil dari *dropout layer* digunakan untuk menghitung *dense layer*. Berikut adalah contoh perhitungan *dense layer* yang berdasarkan persamaan 2.8.

$$y = f\left(\begin{bmatrix} -0.095457 \\ 0.26964079 \\ 0.62887384 \end{bmatrix} \bullet \begin{bmatrix} 0.2286826 \\ 0.31624534 \\ 0.3503066 \end{bmatrix} + \begin{bmatrix} -0.12877315 \\ -0.29305055 \\ 0.51698025 \\ -0.77540555 \end{bmatrix}\right)$$

Dari perhitungan diatas, maka didapat nilai hasil perhitungan *dense*. Berikut hasil dari penghitungan untuk nilai *dense*.

$$y = f\begin{bmatrix} 0.15496879 \\ -0.00930861 \\ 0.80072219 \\ 0.49166361 \end{bmatrix}$$

3.4.10 *Output Layer* dengan Aktivasi *Softmax*

Hasil yang sudah dihitung melalui *dense layer*, dilanjutkan dengan perhitungan selanjutnya dengan menggunakan fungsi aktivasi *softmax*. Perhitungan ini dilakukan dengan mencari probabilitas setiap kelas yang telah didefinisikan sebelumnya. Berikut adalah contoh perhitungan aktivasi *softmax* yang berdasarkan persamaan 2.9.

$$\begin{aligned}\phi_i &= \frac{\exp(z_i)}{\sum_j \exp(z_j)} \\ &= \frac{\exp(0.15496879)}{\exp(0.15496879) + \exp(-0.00930861) + ... + \exp(0.49166361)} \\ &= 0.23365922\end{aligned}$$

Dari perhitungan terakhir diatas, maka didapat nilai hasil perhitungan *softmax* yang dimana hasil ini merepresentasikan *output* yang diinginkan apakah teks ini masuk dalam kategori emosi *others*, *happy*, *sad*, dan *angry*. Berikut hasil dari penghitungan untuk nilai *softmax*.

$$\phi_i = \begin{bmatrix} 0.23365922 & 0.19826139 & 0.4456871 & 0.12239228 \end{bmatrix}$$

Dari hasil nilai yang diatas, menunjukkan hasil prediksi yang didapat. Nilai-nilai tersebut mewakili probabilitas dari jumlah kelas yang ada sesuai dengan indexnya masing-masing dimana indeks 0 mewakili emosi *others*, indeks 1 mewakili *happy*, indeks 2 mewakili *sad*, dan index 3 mewakili *angry*. Indeks yang nilainya paling tinggi mengartikan bahwa teks tersebut termasuk ke dalam kategori emosi sesuai indeksnya dan dari contoh ini nilai paling tinggi ada di indeks 2 yang berarti teks ini masuk ke dalam kategori *sad*.

3.4.11 *Categorical Cross-Entropy*

Setelah menghitung nilai *softmax*, proses selanjutnya adalah dengan menghitung nilai *loss* / *error* menggunakan *categorical cross-entropy*. *Loss function* merupakan fungsi pengukuran *error* dari sebuah model *neural network* dalam melakukan klasifikasi atau prediksi. *Categorical cross-entropy* digunakan karena klasifikasi yang digunakan sebanyak lebih dari 2 kelas dengan menggunakan persamaan 2.10. Contoh perhitungannya adalah sebagai berikut.

$$\begin{aligned} Loss &= -\log(\phi_i) \\ &= -\log(0.23365922) \\ &= 0.6314170774 \end{aligned}$$

3.4.12 *Confusion Matrix*

Confusion matrix digunakan untuk mengukur performa dalam klasifikasi. Pada penelitian ini, terdapat 4 buah *class* dalam mendeteksi emosi dan ke-4 buah *class* tersebut diwakili oleh sebuah nilai berbentuk vektor 4 x 1 yang dimana di setiap vektor tersebut mewakili seberapa besar probabilitas kecenderungan sebuah emosi. Vektor pertama mewakili emosi *"others"* dalam teks, vektor kedua mewakili emosi *"happy"* dalam teks, vektor ketiga mewakili emosi *"sad"* dalam teks, dan vektor keempat mewakili emosi *"angry"* dalam teks. *Confusion matrix* tercipta dari hasil pengujian dibandingkan dengan hasil yang sebenarnya, sehingga ditentukan apakah pengujian memiliki hasil yang baik atau tidak.

Diambilnya 20 teks dari sebuah dataset yang digunakan untuk pengujian sebagai contoh. Dari 20 teks tersebut, dilakukan *text preprocessing* dan *tokenizer*. Kemudian teks tersebut dilakukan prediksi dengan model yang sudah dilatih. Proses pengujian dilakukan dengan menggunakan metode LSTM. Output dari pengujian adalah nilai dari probabilitas terhadap masing-masing *class*. Berikut tabel 3.12 untuk contoh *output* dari hasil pengujian beserta *class* aslinya.

Tabel 3.12 Contoh *Output Testing* Dibandingkan dengan *Class* Aslinya

No	***Output***	***Predicted***	***Actual***
1	[0.23365922, 0.19826139, 0.4456871, 0.12239228]	2	2
2	[0.13365922, 0.5456871, 0.29826139, 0.02239228]	1	1
3	[0.57214479, 0.15294985, 0.02469444, 0.25021092]	0	0
4	[0.11000595, 0.17890865, 0.1726846, 0.5384008]	3	3
5	[0.146054, 0.56719807, 0.21479072, 0.0719572]	1	1
6	[0.14455366, 0.22312725, 0.56534016, 0.06697893]	2	1
7	[0.27490751, 0.2156633, 0.07040159, 0.4390276]	3	3
8	[0.3348164, 0.16867405, 0.05259408, 0.44391548]	3	3
9	[0.14306156, 0.1052263, 0.51716088, 0.23455126]	2	2
10	[0.38866931, 0.33897824, 0.19337403, 0.07897841]	0	1

Tabel 3.12 Contoh *Output Testing* Dibandingkan dengan *Class* Aslinya (Lanjutan)

No	***Output***	***Predicted***	***Actual***
11	[0.08950106, 0.09899153, 0.73787686, 0.07363056]	2	2
12	[0.26694779, 0.18266468, 0.20047083, 0.34991671]	3	3
13	[0.30817482, 0.20555882, 0.14596202, 0.34030434]	3	0
14	[0.42657658, 0.20337053, 0.20509333, 0.16495956]	0	1
15	[0.06900305, 0.00994723, 0.89049897, 0.03055075]	2	2
16	[0.10074773, 0.85664745, 0.01296517, 0.02963965]	1	1
17	[0.83261179, 0.01383494, 0.06257871, 0.09097456]	0	0
18	[0.19484142, 0.10872772, 0.40056537, 0.2958655]	2	3
19	[0.36294168, 0.09103902, 0.28149261, 0.26452669]	0	2
20	[0.14287557, 0.15106955, 0.25815884, 0.44789605]	3	3

Confusion matrix terbentuk dari tabel 3.12 dengan pembagian hasil *predicted* dibandingkan dengan hasil *actual* tercipta tabel *confusion matrix* untuk *multiclass*. Berikut tabel 3.13 *confusion matrix* yang tercipta.

Tabel 3.13 Contoh *Confusion Matrix* 4*x*5 dengan 4 *Class*

		Predicted			
		Others	*Happy*	*Sad*	*Angry*
Actual	*Others*	2	0	0	1
	Happy	2	3	1	0
	Sad	1	0	4	0
	Angry	0	0	1	5

Dari tabel 3.13 bisa didapatkan bahwa untuk kelas *Others* ditemukan nilai *True Positive* sebanyak 2, nilai *True Negative* sebanyak 14, nilai *False Positive* sebanyak 3, dan nilai *False Negative* sebanyak 1. Kelas *Happy* ditemukan nilai *True Positive* sebanyak 3, nilai *True Negative* sebanyak 16, nilai *False Positive* sebanyak 0, dan nilai *False Negative* sebanyak 3. Kelas *Sad* ditemukan nilai *True Positive* sebanyak 4, nilai *True Negative* sebanyak 13, nilai *False Positive* sebanyak 2, dan nilai *False Negative* sebanyak 1. Terakhir kelas *Angry* ditemukan nilai *True Positive* sebanyak 5, nilai *True Negative* sebanyak 13, nilai *False Positive* sebanyak 1, dan nilai *False Negative* sebanyak 1. Dari hasil tersebut, maka ditemukan nilai akurasi dari

pengujian. Berikut perhitungannya dengan menggunakan persamaan 2.11 untuk mendapatkan nilai akurasi.

$$Accuracy = \frac{2+3+4+5}{20} = 0.7$$

Maka ditemukan juga nilai *Precision, Recall,* dan *F1 Score* dari hasil tabel *confusion matrix* tersebut. Berikut penghitungan *recall* menggunakan rumus dari 2.12.

$$Recall = \frac{(2/(2+1)) + (3/(3+3)) + (3/(4+1)) + (5/(5+1))}{4} = 0.65$$

Kemudian perhitungan untuk *precision* dilakukan dengan menggunakan rumus dari 2.13. Berikut perhitungan untuk *precision*.

$$Precision = \frac{(2/(2+3)) + (3/(3+0)) + (3/(4+2)) + (5/(5+1))}{4} = 0.683$$

Pada perhitungan terakhir, dihitungnya nilai F1 *Score* yang didapat dari penghitungan *precision* dan *recall* di atas menggunakan rumus 2.14. Berikut perhitungan untuk *f1 score*.

$$F1Score = \frac{2 \times 0.683 \times 0.65}{0.683 + 0.65} = 0.666$$

Dari tabel 3.13 bisa didapatkan hasil *accuracy*, *recall*, *precision*, dan *F1Score* untuk setiap kelasnya. Berikut hasil dari perhitungan *accuracy*, *recall*, *precision*, dan *F1Score* yang dilihat dalam tabel 3.14.

Tabel 3.14 *Classification Report* dengan 4 *Class*

	Accuracy	*Precision*	*Recall*	*F1Score*
Others	0.80	0.40	0.67	0.50
Happy	0.85	1	0.50	0.67
Sad	0.85	0.67	0.80	0.73
Angry	0.90	0.83	0.83	0.83