# Pengenalan Emosi Dalam Teks Dengan Algoritme *Long Short Term Memory*

Daniel Christianto[#1], Herry Heryanto[#2]
[#]*Departemen Informatika, Institut Teknologi Harapan Bangsa*
*Jl. Dipati Ukur No. 80, Dago, Bandung, Indonesia*
[1]daniel.christianto3@gmail.com
[2]hery_heryanto@ithb.ac.id

***Abstract*—Recognition of emotions in text is a topic of research that has been widely carried out in recent years. Various studies have been carried out to solve the problem of recognizing emotions from texts. The increasing amount of text data that can be applied to various fields has made emotion recognition research develop rapidly, including in this study. This study tested the accuracy using the LSTM method and tested the effect of parameters on the accuracy of emotion detection in the text. There are 3 main indicators in this study, namely the number of emotion dimensions & embedding weights, learning rate, and dropout rate. The dataset used in this study is a dataset from the Computational Linguistics at Concordia (CLaC) Lab which has a total of 30160 data that has text and label features. In this study, 4 classes will be classified, namely others, happy, sad, and angry. Based on testing the number of emotion dimensions & embedding weights, learning rate, and dropout rate, it was found that the best accuracy value for each class in the emotion detection research in text was obtained. In the others class the highest accuracy value is 82%, in the happy class the highest accuracy value is 93%, in the sad class the highest accuracy value is 91%, and in the angry class the highest accuracy value is 94%.***

***Keywords*—*Deep Learning*, *Long Short Term Memory* (LSTM), *Natural Language Processing* (NLP), GloVe (*Global Vector*), *Emotion Detection*.**

***Abstrak*—Pengenalan emosi dalam teks merupakan topik penelitian yang banyak dilakukan beberapa tahun terakhir. Berbagai penelitian telah dilakukan untuk menyelesaikan masalah pengenalan emosi dari teks. Jumlah data teks yang semakin meningkat serta diaplikasikan pada berbagai bidang telah membuat penelitian pengenalan emosi berkembang pesat termasuk dalam penelitian ini. Penelitian ini menguji akurasi menggunakan metode LSTM dan menguji pengaruh parameter terhadap akurasi deteksi emosi dalam teks. Terdapat 3 indikator utama dalam penelitian ini, yaitu jumlah *emotion dimension & embedding weights*, *learning rate*, dan *dropout rate*. Dataset yang digunakan dalam penelitian ini, yaitu dataset dari *Computational Linguistics at Concordia* (CLaC) Lab yang memiliki jumlah data sebanyak 30.160 yang memiliki fitur *text* dan label. Pada penelitian ini dilakukan klasifikasi terhadap 4 kelas, yaitu *others*, *happy*, *sad*, dan *angry*. Berdasarkan pengujian jumlah *emotion dimension & embedding weights*, *learning rate*, dan *dropout rate*, didapatkan nilai akurasi terbaik untuk setiap kelas yang ada pada penelitian deteksi emosi dalam teks. Pada kelas *others* didapatkan nilai akurasi tertinggi sebesar 82%, pada kelas *happy* didapatkan nilai akurasi tertinggi sebesar 93%, pada kelas *sad* didapatkan nilai akurasi tertinggi sebesar 91%, dan pada kelas *angry* didapatkan nilai akurasi tertinggi sebesar 94%.***

***Kata Kunci*—*Deep Learning*, *Long Short Term Memory* (LSTM), *Natural Language Processing* (NLP), GloVe (*Global Vector*), *Emotion Detection*.**

## I. PENDAHULUAN

*Affective computing* merupakan bidang penelitian yang bertujuan untuk mengembangkan sistem komputer yang mampu mendeteksi, menanggapi, dan meniru emosi manusia. Pengenalan emosi merupakan kemampuan utama dan memiliki peran penting dalam *affective computing* [1]. Pengenalan emosi dilakukan berdasarkan ekspresi wajah, suara, gerak tubuh, sinyal electroencephalogram – EEG, dan teks [2].

Pengenalan emosi dalam teks merupakan topik penelitian yang banyak dilakukan beberapa tahun terakhir. Berbagai penelitian telah dilakukan untuk menyelesaikan masalah pengenalan emosi dari teks. Jumlah data teks yang semakin meningkat serta diaplikasikan pada berbagai bidang telah membuat penelitian pengenalan emosi berkembang pesat [4]. Seiring dengan banyaknya data teks yang tersedia menyebabkan pendekatan menggunakan *deep learning* diaplikasikan karena jumlah data yang besar [3].

Pendekatan *deep learning*, seperti *Recurrent Neural Network* (RNN) sangat cocok digunakan untuk memprediksi dengan data sekuensial karena dalam setiap pemrosesan yang dilakukan dalam RNN, *output* yang dihasilkan tidak hanya merupakan fungsi dari sampel itu saja, tapi juga berdasarkan *state internal* yang merupakan hasil dari pemrosesan sampel-sampel sebelumnya [5] [6]. Tetapi RNN memiliki kelemahan pada masalah *vanishing gradient* saat memproses data sekuensial yang panjang sehingga mengurangi akurasi dari suatu prediksi pada RNN [5] [6].

*Long Short Term Memory* (LSTM) merupakan salah satu jenis RNN. LSTM menyimpan informasi terhadap pola-pola pada data. LSTM mempelajari data mana saja yang disimpan dan data mana saja yang dibuang, karena pada setiap neuron LSTM memiliki beberapa *gates* yang mengatur memori pada setiap neuron itu sendiri [5] [6]. LSTM juga menambahkan *memory cell* yang menyimpan informasi untuk jangka waktu yang lama dan LSTM diusulkan sebagai solusi untuk mengatasi terjadinya *vanishing gradient* pada RNN saat memproses data sekuensial yang panjang sehingga metode LSTM ini memecahkan solusi *vanishing gradient* dalam RNN dan hasil prediksi dari model LSTM menjadi lebih akurat dengan nilai akurasi sebesar 85% [5] [6]. Selain itu dalam melakukan

pengenalan emosi, teks harus diubah menjadi bahasa yang dikenali oleh komputer. Oleh karena itu, teks seperti kata atau dokumen diubah menjadi vektor dalam pemrosesan bahasa alami yang disebut *word embedding* [7]. *Word embedding* membantu memberikan sebuah bobot pada suatu data yang menyebabkan data tersebut memiliki sebuah nilai dan dampak dari penggunaan *word embedding* adalah meningkatnya akurasi pada saat pelatihan model. Pada penelitian [6] dilihat model yang menggunakan *word embedding* mendapat nilai akurasi sebesar 95.17%. Penelitian ini memilih metode LSTM untuk diterapkan dalam pengenalan emosi untuk data teks dengan harapan memberikan hasil akurasi yang lebih tinggi dibandingkan beberapa metode sebelumnya.

## II. METODOLOGI

### A. Emotion Detection in Text

*Emotion Detection* (ED) adalah cabang dari analisis sentimen yang berhubungan dengan ekstraksi dan analisis emosi. Bercabang dari bidang analisis sentimen yang tujuan utamanya adalah menganalisis bahasa manusia dengan mengekstraksi pendapat, ide, dan pemikiran melalui penetapan polaritas baik negatif, positif, atau netral adalah cabang bidang dari ED, yang berupaya mengekstraksi lebih halus emosi seperti senang, sedih, marah, dan sebagainya, dari bahasa manusia daripada tugas polaritas kasar dan umum di analisis sentimen. Oleh karena itu ED adalah asosiasi sinergis dari emosi yang juga disebut afeksi dan teknologi serta memperoleh esensinya dari penerapan teknologi yang ditentukan emosi ke area yang berbeda untuk memberikan pengambilan keputusan yang cermat [4].

### B. Long Short Term Memory

LSTM merupakan varian dari *Recurrent Neural Network* (RNN). RNN adalah salah satu model kuat dari keluarga *deep learning* yang telah menunjukkan hasil luar biasa dalam lima tahun terakhir. Ini bertujuan untuk membuat prediksi pada data sekuensial dengan memanfaatkan arsitektur berbasis memori yang kuat. RNN (juga disebut *feedback*) menggunakan status memori tambahan [12]. RNN berisi siklus yang memberikan aktivasi jaringan dari langkah waktu sebelumnya sebagai input ke jaringan untuk mempengaruhi prediksi pada langkah waktu saat ini. Aktivasi ini disimpan dalam keadaan *internal* jaringan yang pada prinsipnya menyimpan informasi kontekstual temporal jangka panjang. Mekanisme ini memungkinkan RNN untuk mengeksploitasi jendela kontekstual yang berubah secara dinamis melalui riwayat urutan input [13].

Jaringan LSTM berbeda dengan *Multilayer Perceptrons* (MLP) klasik. Seperti MLP, jaringan terdiri dari lapisan neuron. Data input disebarkan melalui jaringan untuk membuat prediksi. Seperti RNN, LSTM memiliki koneksi berulang sehingga status dari aktivasi neuron dari langkah waktu sebelumnya digunakan sebagai konteks untuk merumuskan *output*. Tetapi tidak seperti RNN lainnya, LSTM memiliki formulasi unik yang memungkinkannya untuk menghindari masalah yang mencegah pelatihan dan penskalaan RNN lainnya [13]. LSTM mengatasi tantangan ini dengan desain unit komputasi jaringan LSTM yang disebut *memory cell*, *memory block*, atau singkatnya *cell*.

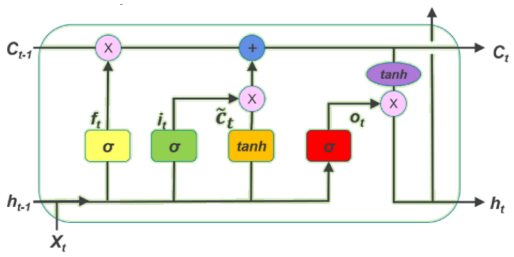


Fig. 1. Ilustrasi Arsitektur LSTM [6]

Setiap unit LSTM, dilihat pada gambar 1 memiliki sel memori, dan keadaan pada waktu $t$ direpresentasikan sebagai $C_t$. Membaca dan memodifikasi dikendalikan oleh gerbang sigmoid dan mempengaruhi *input gate* $i_t$, *forget gate* $f_t$ dan *output gate* $o_t$. LSTM dihitung sebagai berikut: Pada saat ini, model menerima *input* dari dua sumber eksternal $h_{t-1}$ dan $x_t$. Status tersembunyi $h_t$ dihitung dengan *input vector* $x_t$ yang diterima jaringan pada waktu $t$ dan status tersembunyi sebelumnya $h_{t-1}$. Saat menghitung status node lapisan tersembunyi, *input gate*, *output gate*, *forget gate* dan $x_t$ secara bersamaan mempengaruhi status node [6].

*1) Forget Gate*

*Forget gate* adalah gerbang yang memiliki tugas untuk menentukan informasi apa yang harus disimpan dan dibuang pada *cell* [13]. Informasi yang ingin dihitung, diambil *input* dari *hidden state* sebelumnya. Kalkulasi dimulai dari *weight* dikalikan dengan nilai *hidden state* sebelumnya dan nilai *weight* dikalikan dengan *input* yang kemudian ditambahkan dengan nilai bias. Nilai tersebut selanjutnya dihitung menggunakan *sigmoid activation* yang menghasilkan nilai dari *forget gate* dengan hasil apakah disimpan atau dibuang. Berikut adalah persamaan 1 untuk *forget gate*.

$$forget\ gate : f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \quad (1)$$

| | |
|---|---|
| Keterangan | : |
| $f_t$ | : Nilai dari *forget gate*. |
| $\sigma_g$ | : Fungsi aktivasi sigmoid. |
| $W_f$, $U_f$ | : Nilai *weight*. |
| $x_t$ | : Nilai *input* berupa vektor. |
| $h_{t-1}$ | : Nilai dari *hidden state* sebelumnya. |
| $b_f$ | : Nilai *bias*. |

*2) Input Gate*

*Input gate* adalah *gate* yang menentukan nilai *input* baru yang dimana nilai tersebut digunakan untuk *update cell* [13]. Kalkulasi dimulai dari $W_i$ dikalikan dengan nilai dari $x_t$ sebelumnya dan $U_i$ dikalikan dengan nilai *input* $h_{t-1}$. Kemudian hasil tersebut dijumlah dan ditambah dengan nilai $b_i$. Setelah itu nilai tersebut dihitung menggunakan *sigmoid activation* dan

menghasilkan nilai untuk *input gate*. Berikut adalah persamaan 2 untuk penghitungan *input gate*.

$$input\ gate : i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \quad (2)$$

Kalkulasi selanjutnya dilakukan dengan penghitungan yang sama seperti *input gate* dan sebagai pembedanya adalah penghitungan akhir yang menggunakan *tanh activation* untuk menghasilkan nilai *candidate state*. Berikut persamaan 3 untuk penghitungan *candidate state*.

$$candidate\ state : \tilde{C}_t = \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \quad (3)$$

| Keterangan | : |
|---|---|
| $i_t$ | : Nilai dari *input gate*. |
| $\tilde{C}_t$ | : Nilai dari *candidate gate*. |
| $\sigma_g$ | : Fungsi aktivasi sigmoid. |
| $\sigma_h$ | : Fungsi aktivasi *tanh*. |
| $W_i, U_i, W_c, U_c$ | : Nilai *weight*. |
| $x_t$ | : Nilai *input* berupa vektor. |
| $h_{t-1}$ | : Nilai dari *hidden state* sebelumnya. |
| $b_i, b_c$ | : Nilai *bias*. |

*3) Output Gate*

*Output gate* adalah *gate* yang memiliki tugas untuk menentukan nilai yang digunakan untuk penghitungan *hidden state* baru berdasarkan pada *input* dan memori pada *cell* [13]. Penghitungan pertama dilakukan oleh *output gate* yaitu dengan $W_o$ dikalikan dengan nilai $x_t$ dan $U_o$ dikalikan dengan nilai $h_{t-1}$. Didapat dua nilai hasil perkalian tersebut yang kemudian dijumlahkan. Hasil penjumlahan tersebut ditambah dengan nilai $b_o$. Kemudian hasil tersebut dilakukan penghitungan menggunakan *sigmoid activation* dan dihasilkannya nilai untuk *output gate*. Berikut persamaan 4 untuk *output gate*.

$$output\ gate : o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \quad (4)$$

| Keterangan | : |
|---|---|
| $o_t$ | : Nilai dari *output gate*. |
| $\sigma_g$ | : Fungsi aktivasi sigmoid. |
| $W_o$, $U_o$ | : Nilai *weight*. |
| $x_t$ | : Nilai *input* berupa vektor. |
| $h_{t-1}$ | : Nilai dari *hidden state* sebelumnya. |
| $b_o$ | : Nilai *bias*. |

*4) Cell State*

*Cell state* adalah bagian tambahan dari RNN yang memiliki tugas sebagai memori. Tahapan ini menentukan memori dari sebelumnya apa yang bisa disimpan atau dilupakan. Sebelum penghitungan, dipastikan dimulai dari memiliki hasil nilai-nilai dari *forget gate* dan *input gate*. Penghitungan dimulai dari nilai *forget gate* dikalikan dengan nilai *cell state* sebelumnya. Kemudian nilai *input gate* dikalikan dengan *candidate state*. Dari kedua nilai tersebut, dijumlahkan hasilnya yang menghasilkan nilai *cell state* yang terbaru. Berikut persamaan 5 untuk penghitungan terbaru nilai *cell state*.

$$cell\ state : C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \quad (5)$$

| Keterangan | : |
|---|---|
| $C_t$ | : Nilai dari *cell state*. |
| $f_t$ | : Nilai dari *forget gate*. |
| $C_{t-1}$ | : Nilai dari *cell state* sebelumnya. |
| $i_t$ | : Nilai dari *input gate*. |
| $\tilde{C}_t$ | : Nilai dari *candidate state*. |

*5) Hidden State*

Setelah nilai dari *output gate* didapat, perhitungan selanjutnya dilakukan untuk menemukan nilai *hidden state* baru. Perhitungan dilakukan dengan nilai *cell state* dihitung dengan *tanh activation* dan kemudian hasil tersebut dikalikan dengan nilai *output gate*. Dari perhitungan tersebut didapat nilai *hidden state* baru yang diteruskan untuk penghitungan selanjutnya. Berikut persamaan 6 untuk penghitungan menentukan nilai *hidden state*.

$$hidden\ state : h_t = o_t \odot \sigma_h(C_t) \quad (6)$$

| Keterangan | : |
|---|---|
| $h_t$ | : Nilai dari *hidden state*. |
| $o_t$ | : Nilai dari *output gate*. |
| $\sigma_h$ | : Fungsi aktivasi *tanh*. |
| $C_t$ | : Nilai dari *candidate state*. |

## III. Analisis dan Perancangan Sistem

### A. Kerangka Pemikiran

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

Berikut adalah rangkaian penjelasan setiap bagian kerangka pemikiran pada gambar 2:

- *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*.
  1) *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.

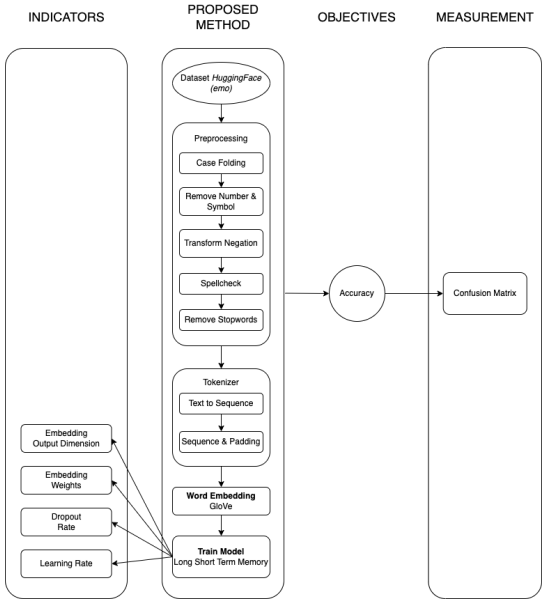


Fig. 2. Kerangka Pemikiran

2) *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.
3) *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.
4) *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.

- *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.
- *Objectives* adalah bagian yang menjelaskan mengenai hal-hal yang menjadi acuan pengukuran dan pengukuran yang digunakan adalah akurasi model.
- *Measurement* adalah satuan ukur pada bagian *Objectives*, seperti perhitungan *confusion matrix*.

### *B. Urutan Proses Global*

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

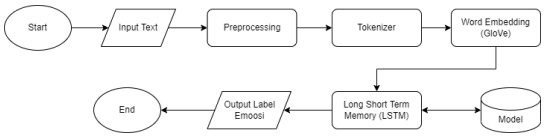


Fig. 3. Urutan Proses Global

Berikut adalah rangkaian penjelasan urutan proses global pada gambar 3:

- *Dataset* yang digunakan merupakan data teks dengan label emosinya yang diambil dari *website HuggingFace* dengan data bernama emo.
- 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.
- 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.
- *Word embedding* di *generate* untuk disiapkan dimasukkan ke dalam *embedding layer*. Digunakannya data *pretrained* GloVe untuk menghasilkan *word embedding*.
- Setelah dilakukan proses *tokenizer* dan menghasilkan *word embedding*, dibuat sebuah model LSTM, dimana dalam model LSTM ini dibuat menggunakan 4 *layer*, yaitu:
  1) *Embedding Layer* = Dalam *embedding layer* ini digunakan 5 parameter, yaitu: *input_dim, output_dim, input_length, weights*, dan *trainable*.
  2) LSTM*Layer* = Dalam LSTM*layer* digunakan 1 parameter, yaitu: *units*.
  3) *Dropout Layer* = Dalam *dropout layer* digunakan 1 parameter, yaitu: *rate*.
  4) *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.
- Setelah model dibuat, maka dilakukan pengenalan emosi menggunakan model LSTM tersebut dan dikategorikan ke dalam 4 macam, yaitu:
  1) *Others* (0)
  2) *Happy* (1)
  3) *Sad* (2)
  4) *Angry* (3)

## IV. Implementasi Dan Pengujian

### A. Implementasi Arsitektur

Penelitian ini mengimplementasikan sebuah arsitektur yang diuji dengan sebuah dataset untuk mengetahui pengaruh dari penambahan *layer* terhadap model yang dibuat. Arsitektur ini digunakan dari [5] dengan ditambahkannya lapisan *Dropout* sebagai regularisasi model agar model tidak menjadi *overfit* ketika model sedang melakukan pembelajaran. Gambar 4 menunjukan rancangan arsitektur yang digunakan dalam penelitian ini beserta dengan dimensi *input* dan *output*:

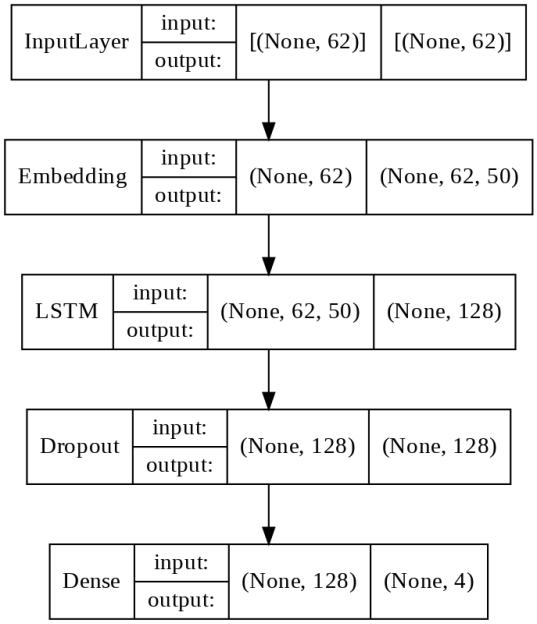


Fig. 4. Rancangan Arsitektur dalam Penelitian

Gambar 4 memperlihatkan bahwa ada 4 lapisan yang digunakan ketika membuat model ini, keempat lapisan itu adalah:

- *Input Layer* menerima *input* dengan panjang vektor sebanyak 62.
- *Embedding Layer* dengan parameter input_dim yang merupakan jumlah *input dimension*, output_dim yang merupakan jumlah *output dimension*, input_length yang merupakan panjang dari *sequence*, *weights* yang merupakan matriks *embedding* dari GloVe, dan *trainable* diberikan *input False* untuk menjaga nilai vektor tidak berubah. *Embedding layer* menerima *input* dengan panjang vektor 62 dan menghasilkan *output* dengan panjang vektor 62 x 50.
- *Long Short Term Memory Layer* menerima input dengan ukuran panjang *sequence* X dimensi GloVe, yaitu 62 x 50 dan menghasilkan *output* dengan panjang vektor 128.
- *Dropout Layer* dengan parameter rate yang merupakan fraksi unit *input* untuk dijatuhkan. *dropout layer* menerima *input* dengan panjang vektor 128 dan menghasilkan *output* dengan panjang vektor 128.
- *Dense Layer* dengan fungsi aktivasi *softmax* yang digunakan sebagai *output layer* untuk deteksi. *Dense layer* menerima input dengan ukuran panjang vektor 128 dan menghasilkan *output* dengan panjang vektor 4.

### B. Hasil Pengujian

#### 1) Kelas Others

Berdasarkan hasil pengujian pada kelas *others* dilihat bahwa setiap model yang ada di atas memiliki perbedaan nilai *precision*, *recall*, *f1 score*, dan akurasi yang sangat tipis. Meskipun dengan nilai tertinggi ada pada model dengan parameter dimensi 300, *learning rate* 0.0001, dan *dropout* 0.3 yang dimana pada model ini memiliki kompleksitas yang tinggi hanya memiliki perbedaan nilai yang sangat kecil dibanding model lainnya.

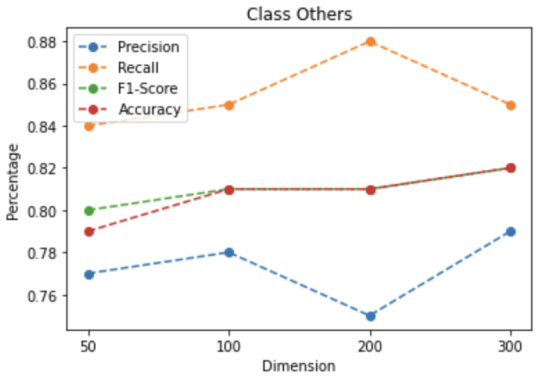


Fig. 5. Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* Terbaik Untuk Setiap Dimensi Pada Kelas *Others*

Berdasarkan gambar 5 didapatkan hasil pengujian terbaik dari setiap dimensi untuk kelas *others*. Dari hasil tersebut disimpulkan bahwa model terbaik untuk kelas *others* terdapat pada parameter dengan nilai dimensi sebesar 300, *dropout rate* sebesar 0.3, dan *learning rate* sebesar 0.0001. Dari model tersebut didapat nilai *precision* sebesar 79%, nilai *recall* sebesar 85%, nilai *f1 score* sebesar 82%, dan nilai akurasi sebesar 82%.

#### 2) Kelas Happy

Berdasarkan hasil pengujian pada kelas *happy* dilihat bahwa 3 model (model 2, model 3, dan model 4) di atas memiliki nilai akurasi yang sama, yaitu 93%. Namun hal yang unik ada pada model 2 dan model 3 yang memiliki nilai *learning rate* dan *dropout* yang sama, dimana yang berbeda dari kedua model tersebut adalah nilai dimensi yang digunakan. Hal ini mengartikan bahwa model yang memiliki dimensi yang tinggi belum tentu mendapatkan nilai yang lebih bagus dibandingkan model yang memiliki dimensi yang lebih rendah.

Berdasarkan gambar 6 didapatkan hasil pengujian terbaik dari setiap dimensi untuk kelas *happy*. Dari hasil tersebut disimpulkan bahwa model terbaik untuk kelas *happy* terdapat pada parameter dengan nilai dimensi sebesar 300, *dropout rate* sebesar 0.1, dan *learning rate* sebesar 0.0001. Dari model tersebut didapat nilai *precision* sebesar 79%, nilai *recall* sebesar 61%, nilai *f1 score* sebesar 69%, dan nilai akurasi sebesar 93%.

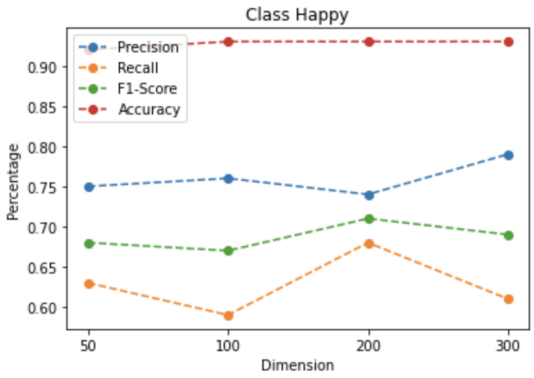


Fig. 6. Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* Terbaik Untuk Setiap Dimensi Pada Kelas *Happy*

*3) Kelas Sad*

Berdasarkan hasil pengujian pada kelas *sad* dilihat bahwa semakin kecil nilai *learning rate* meningkatkan nilai akruasi pada model. Hal ini disebabkan nilai *learning rate* menentukan besarnya perubahan yang diaplikasikan pada nilai bobot dan bias sehingga semakin besar nilai *learning rate* membuat komputasi pada saat pembelajaran menjadi lebih cepat, namun memiliki kekurangan yang menyebabkan model mendapatkan akurasi yang kurang maksimal begitu juga sebaliknya.

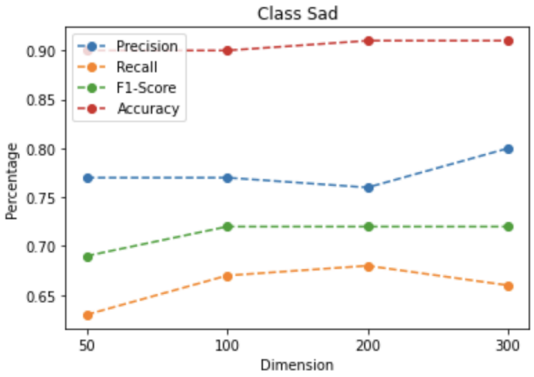


Fig. 7. Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* Terbaik Untuk Setiap Dimensi Pada Kelas *Sad*

Berdasarkan gambar 7 didapatkan hasil pengujian terbaik dari setiap dimensi untuk kelas *sad*. Dari hasil tersebut disimpulkan bahwa model terbaik untuk kelas *sad* terdapat pada parameter dengan nilai dimensi sebesar 300, *dropout rate* sebesar 0.4, dan *learning rate* sebesar 0.001. Dari model tersebut didapat nilai *precision* sebesar 80%, nilai *recall* sebesar 66%, nilai *f1 score* sebesar 72%, dan nilai akurasi sebesar 91%.

*4) Kelas Angry*

Berdasarkan hasil pengujian pada kelas *angry* dilihat bahwa model 3 memiliki akurasi yang lebih rendah dibanding model 4 meskipun nilai *learning rate* yang digunakan pada model 3 lebih kecil dibandingkan pada model 4. Hal ini mengartikan bahwa nilai yang didapatkan pada model tidak selalu tergantung pada *learning rate* dan nilai *dropout* memberikan dampak juga terhadap model yang memiliki kompleksitas yang tinggi.

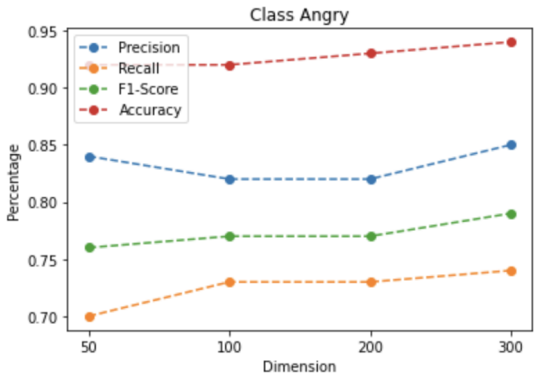


Fig. 8. Visualisasi Plot *Precision*, *Recall*, *F1 Score*, dan *Accuracy* Terbaik Untuk Setiap Dimensi Pada Kelas *Angry*

Berdasarkan gambar 8 didapatkan hasil pengujian terbaik dari setiap dimensi untuk kelas *angry*. Dari hasil tersebut disimpulkan bahwa model terbaik untuk kelas *angry* terdapat pada parameter dengan nilai dimensi sebesar 300, *dropout rate* sebesar 0.4, dan *learning rate* sebesar 0.0001. Dari model tersebut didapat nilai *precision* sebesar 85%, nilai *recall* sebesar 74%, nilai *f1 score* sebesar 79%, dan nilai akurasi sebesar 94%.

## V. KESIMPULAN

Kesimpulan dari pembuatan sistem deteksi emosi dalam teks dengan menggunakan metode *Long Short Term Memory* melalui pengujian yang telah dilakukan. Berikut adalah rincian dari kesimpulan setelah dilakukannya pengujian:

- Pada pengujian ini didapatkan nilai akurasi terbaik untuk setiap kelas yang ada pada penelitian deteksi emosi dalam teks. Pada kelas *others* didapatkan nilai akurasi tertinggi sebesar 82%, pada kelas *happy* didapatkan nilai akurasi tertinggi sebesar 93%, pada kelas *sad* didapatkan nilai akurasi tertinggi sebesar 91%, dan pada kelas *angry* didapatkan nilai akurasi tertinggi sebesar 94%. Disimpulkan bahwa metode LSTM di menghasilkan performa akurasi yang sangat baik.
- Pengujian nilai *Embedding Dimension & Embedding Weights* terhadap deteksi emosi dalam teks berpengaruh terhadap akurasi. Dengan menggunakan nilai *Embedding Dimension & Embedding Weights* yang kecil menghasilkan nilai akurasi yang baik tetapi tidak optimal dibandingkan dengan nilai *Embedding Dimension & Embedding Weights* yang besar dengan hasil akurasi yang optimal.
- Nilai *learning rate* mempengaruhi hasil akurasi pada deteksi emosi dalam teks. Nilai *learning rate* yang besar mempercepat hasil belajar tetapi menghasilkan model yang *overfit* dan menghasilkan akurasi yang tidak optimal dibandingkan dengan nilai *learning rate* kecil yang memperlambat model belajar, namun menghasilkan nilai akurasi yang optimal.

- Pengaruh nilai *dropout rate* terhadap akurasi tergantung pada nilai parameter lainnya yang dimana semakin tinggi kompleksitas (nilai *learning rate* yang kecil dan nilai *Embedding Dimension & Embedding Weights* yang besar) suatu model ketika pelatihan membutuhkan nilai *dropout rate* yang semakin besar agar model tidak menjadi *overfit* dan mengimbangi model yang memiliki kompleksitas tinggi sedangkan semakin kecil kompleksitas model ketika pelatihan membutuhkan nilai *dropout rate* yang tidak terlalu besar.

## VI. Saran

Saran untuk pengembangan yang dilakukan untuk sistem deteksi emosi dalam teks adalah sebagai berikut.

- Dataset yang digunakan disarankan memiliki jumlah kata-kata yang beragam di setiap datanya dan memiliki jumlah data yang banyak dikarenakan jumlah data yang beragam ini membuat model bisa memprediksi berbagai macam panjang kata dengan lebih akurat.
- Dataset yang digunakan disarankan untuk diproses lebih jauh terutama dalam mengkoreksi kata - kata yang menjadi *noise*, seperti kata - kata yang salah penulisan, kata - kata yang disingkat, dan kata - kata yang tergabung karena pada penelitian ini hanya dilakukan pengkoreksian kata yang paling banyak muncul saja bila di gambar plotnya menggunakan *word cloud*.
- Mengembangkan dan mengoptimisasi metode dengan penerapan *Bidirectional* LSTM untuk penelitan selanjutnya.

## References


[1] J. Marín-Morales, C. Llinares, J. Guixeres, and M. Alcañiz, "Emotion Recognition in Immersive Virtual Reality: From Statistics to Affective Computing", *Sensors*, vol. 20, no. 18, p. 5163, 2020.

[2] M.-H. Su, C.-H. Wu, K.-Y. Huang, and Q.-B. Hong, "LSTM-based Text Emotion Recognition Using Semantic and Emotional Word Vectors", *2018 First Asian Conference on Affective Computing and Intelligent Interaction (ACII Asia)*, May 2018.

[3] M. Krommyda, A. Rigos, K. Bouklas, and A. Amditis, "An Experimental Analysis of Data Annotation Methodologies for Emotion Detection in Short Text Posted on Social Media", *Informatics*, vol. 8, no. 1, p. 19, 2021.

[4] F. A. Acheampong, C. Wenyu, and H. Nunoo-Mensah, "Text-based Emotion Detection: Advances, Challenges, and Opportunities", *Engineering Reports*, vol. 2, no. 7, 2020.

[5] N. Murthy, S. Rao Allu, B. Andhavarapu, M. Bagadi, and M. Belusonti, "Text Based Sentiment Analysis Using LSTM", *International Journal of Engineering Research and Technology*, vol. 9, no. 05, May 2020.

[6] W. K. Sari, D. P. Rini, and R. F. Malik, "Text Classification Using Long Short Term Memory with GloVe Features", *Scientific Journal of Computer Electrical Engineering and Informatics*, vol. 5, no. 2, p. 85, 2020.

[7] H. Park and K. Kim, "Impact of Word Embedding Methods on Performance of Sentiment Analysis with Machine Learning Techniques," *Journal of The Korea Society of Computer and Information*, vol. 25, no. 8, pp. 181–188, Aug. 2020.

[8] Pramod Singh, *Machine Learning with PySpark With Natural Language Processing and Recommmender Systems,* 1st ed. Reading, MA: Apress, 2019. [E-book] Available: z-library.

[9] B. Bengfort, *Applied Text Analysis with Python: Enabling Language Aware Data Products with Machine Learning,* O'Reilly Media, Incorporated, 2018. [E-book] Available: z-library.

[10] M. T. Pilehvar and J. Comacho-Collados, *Embeddings in Natural Language Processing: Theory and Advances in Vector Representations of Meaning*, Morgan and Claypool, 2020. [E-book] Available: z-library.

[11] Pennington, J., Socher, R. and Manning, C.D., "Glove: Global Vectors for Word Representation", *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (EMNLP), pp. 1532-1543, 2014.

[12] S. Kostadinov, *Recurrent Neural Networks with Python Quick Start Guide: Sequential Learning and Language Modeling with Tensorflow,* PACKT Publishing Limited, 2018. [E-book] Available: z-library.

[13] J. Brownlee, *Long Short-Term Memory Networks with Python: Develop Sequence Prediction Models with Deep Learning,* Machine Learning Mastery, 2017. [E-book] Available: z-library.

[14] K. Sailunaz, M. Dhaliwal, J. Rokne, and R. Alhajj, "Emotion Detection From Text and Speech: A Survey", *Social Network Analysis and Mining*, vol. 8, no. 1, 2018.

[15] N. Srivastava, G. Hinton, A. Krizhevsky, and I. Sutskever, "Dropout: A Simple Way to Prevent Neural Networks From Overfitting", *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.

[16] Kingma, Diederik P., and Jimmy Ba., "Adam: A Method for Stochastic Optimization.", *International Conference for Learning Representations*, San Diego, 2015.

[17] J. Heaton, *Artificial Intelligence for humans, volume 3: Deep Learning and Neural Networks*, vol. 3. s.l.: Heaton Research, Inc., 2015. [E-book] Available: z-library.

[18] Y. Ho and S. Wookey, "The Real World Weight Cross-Entropy Loss Function: Modeling the costs of Mislabeling", *IEEE Access*, vol. 8, pp. 4806–4813, 2020.

[19] N. Babanejad, A. Agrawal, A. An, and M. Papagelis, "A Comprehensive Analysis of Preprocessing for Word Representation Learning in Affective Tasks", *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[20] A. Chatterjee, K. N. Narahari, M. Joshi, and P. Agrawal, "Semeval-2019 task 3: Emocontext Contextual Emotion Detection in Text", *Proceedings of the 13th International Workshop on Semantic Evaluation*, 2019.



**Daniel Christianto**, menerima gelar Sarjana Komputer tahun 2022 dari Institut Teknologi Harapan Bangsa.

**Hery Heryanto**, menerima gelar Sarjana Komputer tahun 2004 dan Magister Sistem Informasi tahun 2010 dari STMIK LIKMI. Tahun 2016 menerima gelar Doktor dari STEI ITB. Saat ini aktif sebagai dosen di Prodi Informatika ITHB. Minat penelitian pada bidang ilmu *speech processing* dan basis data.