Deteksi *Hate Speech* Pada Twitter Menggunakan Algoritma BERT

1st Adine Nayla
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
adinenayla@student.telkomuniversity.ac.id

2nd Casi Setianingsih
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
setiacasie@telkomuniversity.ac.id

3rd Burhanuddin Dirgantoro
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
burhanuddin@telkomuniversity.ac.id

Abstrak— *Hate speech* atau ujaran kebencian pada salah satu platform sosial media yaitu Twitter sudah tidak jarang ditemukan. Pada *platform* Twitter, pengguna bebas mendapatkan, bertukar informasi, serta mengungkapkan opini. Hal ini merupakan salah satu faktor utama seseorang dapat terkena ujaran kebencian pada Twitter. Korban yang terkena ujaran kebencian memiliki kemungkinan menderita gangguan kesehatan mental, dikarenakan sebagian besar korban ujaran kebencian diserang secara verbal atapun emosional. Minimnya penanggulangan deteksi ujaran kebencian pada platform sosial media Twitter masih jarang ditemukan. Pada penelitian ini, dilakukan proses simulasi menggunakan website beserta dengan pengujian dan analisis terhadap pendeteksian ujaran kebencian. Pengujian dilakukan dengan cara pengguna akan melakukan *input* kalimat pada *website hate speech*, lalu *website* akan melakukan *preprocessing* dan menganalisa kalimat tersebut menggunakan Algoritma BERT untuk mengklasifikasikan apakah kalimat tersebut termasuk *hate speech* atau tidak. Dari hasil pengujian diperoleh bahwa pendeteksian *hate speech* pada akun pengguna Twitter menggunakan Algoritma BERT mendapatkan akurasi sebesar 78.69%, presisi sebesar 78.90%, recall sebesar 78.69%, dan F1 score sebesar 78.77% terhadap pengklasifikasian golongan *hate speech*. Dengan demikian pengguna akan lebih mudah mendeteksi *hate speech* pada Twitter dengan menggunakan *website hate speech*.

Keywords-- algoritma *BERT*, aplikasi web, *hate speech*, twitter.

I. PENDAHULUAN

Pada zaman modern ini, teknologi merupakan hal yang tidak asing bagi suatu individu. Berbagai macam teknologi telah tercipta, salah satunya adalah teknologi informasi dimana pada teknologi informasi hadir sebuah istilah yang dinamakan dengan sosial media. Sosial media mempunyai arti sebagai media pertukaran informasi secara online dengan berbagai macam *platform* yang memiliki tujuan dan manfaat yang berbeda [1]. Pada sosial media, pengguna dapat bertukar ataupun mendapatkan informasi secara bebas. Perkembangan sosial media yang pesat dapat mempengaruhi sikap dan pola perilaku masyarakat.

Semenjak kemunculan sosial media, penggunaan *website* mengalami penurunan. Hal ini terjadi dikarenakan fitur pada sosial media lebih unggul daripada fitur *website*, salah satunya adalah pertukaran informasi yang selektif dan objektif dengan fitur pengaturan yang dapat diatur oleh pengguna secara bebas. Oleh karena itu, pengguna sosial media dapat dengan bebas mengakses informasi pribadi orang lain dan memanfaatkan informasi tersebut [2]. Dikarenakan kebebasan tersebut, peluang terjadinya *cyber bullying* meningkat.

Peningkatan terjadinya *cyber bullying* disebabkan oleh penyalahgunaan sosial media, salah satu contoh penyalahgunaan sosial media adalah ujaran kata kasar terhadap suatu kelompok ataupun individu. Sebuah individu atau kelompok yang mengalami *cyber bullying* memiliki risiko gejala gangguan kesehatan mental yang tinggi seperti depresi, kecemasan berlebihan, perubahan pola tidur dan pola makan, serta kehilangan minat untuk beraktivitas [2]. Berdasarkan hal tersebut, pendeteksi ujaran kebencian pada *platform* sosial media merupakan hal yang penting guna mencegah dan meminimalisir terjadinya *cyber bullying* pada sosial media.

II. KAJIAN TEORI

A. *Natural Language Processing*

Natural Language Processing adalah sebuah ilmu terapan kecerdasan buatan mengenai pengolahan bahasa natural yang bertujuan untuk memproses atau menerjemahkan kata-kata yang disampaikan oleh manusia agar dapat dimengerti oleh komputer [3]. *Natural Language Processing* diciptakan sekitar tahun 1950-an sebagai titik potong antara *artificial intelligence* dan ilmu linguistik [4]. Pengertian dari bahasa natural itu sendiri adalah bahasa umum yang digunakan oleh manusia untuk berkomunikasi [3]. Bahasa ini tidak mudah untuk diaplikasikan pada komputer dikarenakan membutuhkan proses pemahaman yang lama untuk menerjemahkan bahasa tersebut.

B. *Neural Network*

Neural network atau jaringan saraf adalah suatu sistem komputasi dengan node yang saling berhubungan yang bekerja seperti neuron di otak manusia. *Neural network* bekerja menggunakan algoritma, dengan ini *neural network* dapat mengenali pola dan korelasi tersembunyi dalam data mentah, mengelompokkan dan mengklasifikasikannya secara waktu ke waktu dengan tujuan untuk terus belajar dan peningkatan. Terdapat empat jenis *neural network*, yaitu *Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Feedforward Neural Networks, Autoencoder Neural Networks* [5].

Pada *Convolutional Neural Networks (CNNs)* terdapat lima jenis lapisan: *input, convolution, pooling, fully connected* dan *output*. Setiap lapisan memiliki tujuan tertentu, seperti *summarizing*, *connecting*, atau *activating*. CNN biasa diterapkan pada *natural language processing* dan juga *forecasting*.

Sedangkan *Recurrent Neural Networks (RNNs)* menggunakan informasi sekuensial seperti data waktu dari perangkat sensor. Tipe *neural network* ini tidak seperti tipe *neural network* lainnya, RNN tidak *independent* satu sama lain dan *output* yang dihasilkan pada setiap elemen bergantung kepada perhitungan pada elemen sebelumnya. RNN biasa diterapkan pada *forecasting*, analisis sentimen dan aplikasi teks lainnya.

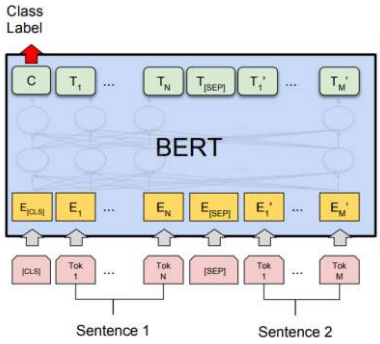
Berbeda dengan CNN dan RNN, *Feedforward neural networks* memliki perceptron yang terhubung ke stiap perceptron lapisan berikutnya. Informasi tersebut akan dikirimkan secara forward dari satu lapisan ke lapisan lainnya dan tidak ada *loop* umpan balik.

Terakhir yaitu *Autoencoder Neural Network*, pada tipe ini digunakan untuk membuat abstraksi yang disebut *encoder, encoder* terbentuk dari serangkaian input tertentu. Tipe ini hampir mirip dengan tipe *neural network* lainnya akan tetapi *autoencoder* berusaha untuk memodelkan *input* tersendiri [5].

C. Algoritma BERT

Bidirectional Encoder Representations from Transformers atau biasa disebut BERT adalah sebuah algoritma deep learning keluaran Google yang masih berkaitan dengan NLP (*Natural Language Processing*). Algoritma ini adalah invasi dari model Transformer dimana model tersebut memproses sebuah kata pada kalimat berdasarkan ada atau tidaknya kaitan antara kata tersebut dengan kalimat secara keseluruhan. Cara kerja algoritma BERT ini tidak seperti algoritma pemrosesan bahasa lainnya. BERT akan memproses sebuah kata dengan mempelajari konteks daripada kata tersebut berdasarkan kata-kata yang ada. Algoritma BERT memproses konteks penuh dengan cara melihat pola yang muncul pada sebelum atau sesudah kata [6].

BERT dilatih menggunakan 2500 juta kata dari Wikipedia dan 800 juta kata dari buku. Pelatihan BERT ini tergolong menjadi dua metode *modeling* yaitu *Masked Language Model (MLM)* dan *Next Sentence Prediction (NSP)*. Pada MLM, sebuah kata pada *corpus* akan disembunyikan dan dilatih. Sementara pada NSP, BERT akan memprediksi kalimat selanjutnya yang memiliki konteks sesuai dengan kalimat sebelumnya.



GAMBAR 1
ARSITEKTUR BERT

Berdasarkan gambar 1, arsitektur BERT-Base mempunyai 110 juta parameter yaitu jumlah variabel yang tersedia dan dapat dipelajari oleh model. Pada arsitektur BERT-Base, terdapat 12 layer transformer dimana pada setiap blok transformer dapat merubah urutan representasi pada sebuah kata menjadi urutan kata atau kalimat yang sudah dikontekstualisasikan. Selain itu BERT mempunyai 768 *hidden size* atau 768 layer fungsi matematika yang terletak di antara input dan *output* dengan fungsi untuk memberikan bobot pada setiap kata. Tentunya BERT juga memiliki 12 *attention heads* dimana pengertian dari *attention heads* itu sendiri adalah ukuran blok dari sebuah transformer [7].

D. *Text Pre-processing*

Text preprocessing bertujuan untuk menyiapkan dan membersihkan kalimat sebelum memasuki tahap selnjutnya. Tahapan pada *text preprocessing* adalah *case folding, filtering, stemming, dan tokenizing*.

1. *Case Folding* (CF)
   Case folding adalah proses mengubah kalimat dari huruf kapital menjadi huruf kecil. Proses ini dilakukan karena arti kata yang menggunakan huruf kapital sama dengan arti kata dengan huruf kecil [8].
2. *Filtering* (F)
   Filtering adalah metode *preprocessing* yang digunakan untuk menghilangkan sufiks karena seringkali dianggap tidak penting dalam menentukan analisis sentimen teks.
3. *Stemming* (S)
   Stemming adalah metode *preprocessing* yang digunakan untuk mereduksi bentuk infleksional dan bentuk turunan dari sebuah kata menjadi bentuk dasar yang sama [8].
4. *Tokenizing* (T)
   Tokenizing adalah metode *preprocessing* yang digunakan untuk memisahkan setiap kata,

sehingga tidak lagi membentuk kalimat yang utuh.

Contoh *preprocessing* dapat dilihat pada Tabel 1.

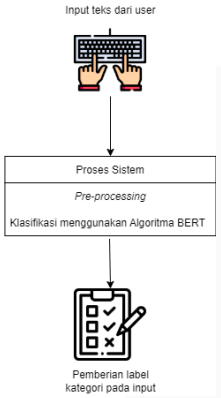
TABEL 1
TEXT PREPROCESSING

Text	CF	F	S	T
She's been MIA for 3 days	she's been mia for 3 days	she mia 3 days	She mia 3 days	'she', '#s', 'been', 'mia', 'for', '3', 'days'

III. METODE

A. Arsitektur Aplikasi

Aplikasi pendeteksi ujaran kebencian bekerja dengan mengklasifikasikan teks berbahasa Inggris sebagai input dan sistem akan menghasilkan keluaran sebagai label dan probabilitas akurasi terhadap klasifikasi ujaran kebencian. Ilustrasi aplikasi ditunjukkan pada Gambar 2 di bawah ini.



GAMBAR 2
ARSITEKTUR WEBSITE HATE SPEECH

Gambar 2 menjelaskan gambaran umum dari aplikasi pendeteksi ujaran kebencian yang dimulai ketika pengguna telah membuka aplikasi web dengan memasukkan kalimat yang ingin dideteksi. Ketika pengguna memiliki teks input, sistem akan melakukan pra-proses data. Jika sistem telah selesai pra-pemrosesan, hasilnya akan diklasifikasikan menggunakan algoritma BERT dan data akan diberi label dan tingkat akurasi akan dikeluarkan sebagai output.

B. Perangkat Lunak

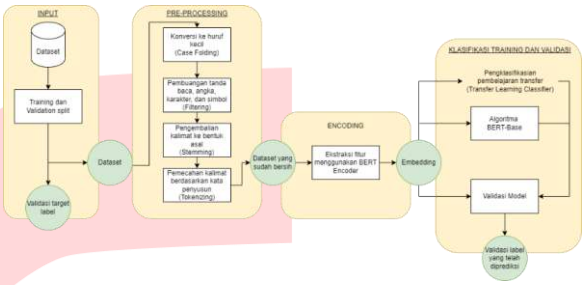
Perangkat lunak yang digunakan dalam perancangan model, pembuatan *website*, dan simulasi untuk sistem menggunakan perangkat lunak dengan spesifikasi sebagai berikut:

1. Bahasa pemrograman python
2. Google Colab
3. Flask module
4. Keras module
5. SKLearn module
6. Tensorflow module

C. Perancangan Sistem

Perancangan sistem adalah gambaran dari sistem deteksi *hate speech* yang akan dirancang. Dalam perancangan sistem ini terdapat gambaran diagram alir sistem dan Algoritma BERT.

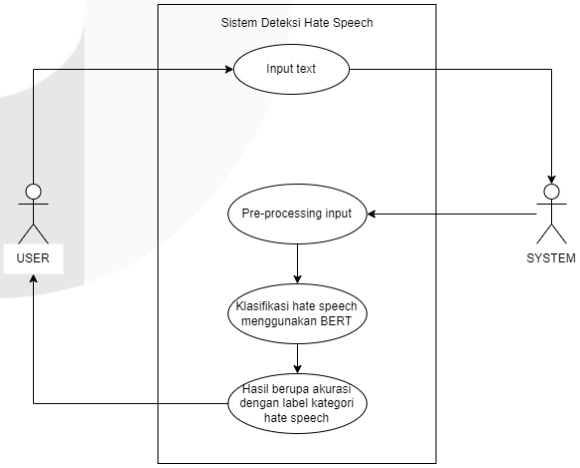
1. Diagram Alir Sistem



GAMBAR 3
ALUR SISTEM

Pada tahap ini, sistem akan bekerja dengan menerima sebuah input data. Data tersebut kemudian akan dilakukan *pre-processing* oleh sistem dengan tujuan untuk mempersiapkan data agar bisa memasuki tahap selanjutnya dengan optimal. *Output* sistem adalah probabilitas input terhadap 3 golongan hate speech.

2. *Use Case Diagram*

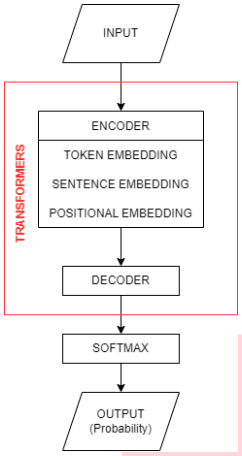


GAMBAR 4
USE CASE DIAGRAM

Berdasarkan Gambar 4, terdapat dua *actor* pada diagram yang terdiri dari *user* dan sistem. *User* akan berinteraksi secara langsung dengan halaman *website* untuk menginput kan sebuah kalimat yang akan dilakukan proses klasifikasi, setelah itu sistem akan

mengirimkan output berupa akurasi dan pelabelan kalimat.

3. Algoritma BERT



GAMBAR 5
FLOWCHART ALGORITMA BERT

Pada Gambar 5 merupakan alur kerja dari sistem Algoritma BERT, jika terdapat input maka BERT akan mengirimkan input tersebut kepada *transformer encode*r dimana input akan melakukan proses *token embedding, sentence embedding, dan positional embedding*. Setelah input telah selesai pada bagian *encoder, encoder* akan mengirimkan hasil output kepada *decoder*. *Decoder* akan mengirimkan hasil output kepada *softmax* agar hasil matriks dapat diubah menjadi probabilitas. Probabilitas ini akan menjadi hasil akhir pada sistem.

D. Desain Sistem

Tahap awal agar sistem dapat bekerja adalah menerima masukan yang diperoleh dari pengguna. Setelah mendapatkan input, data akan diproses terlebih dahulu sehingga dataset siap untuk digunakan pada tahap selanjutnya.

1. Text Preprocessing

Proses ini akan mengubah kalimat mentah sebuah kata menjadi kalimat terstruktur sesuai kebutuhan tahap selanjutnya. Proses ini terdiri dari *case folding, tokenizing, filtering, dan stemming.*

sebuah.

a. Case Folding

Proses ini digunakan untuk mengubah huruf besar menjadi huruf kecil. Contohnya dapat dilihat pada Tabel 2 di bawah ini.

TABEL 2
CASE FOLDING

Input	Output
That bitch has been MIA for 3 days & no words from her at all.	that bitch has been mia for 3 days & no words from her at all.

b. Filtering

Proses ini dilakukan untuk menghilangkan angka, kalimat penyambung, dan simbol-simbol yang tidak diperlukan. Contoh dari *filtering* dapat dilihat pada tabel 3.

TABEL 3
FILTERING

Input	Output
that bitch has been mia for 3 days & no words from her at all.	that bitch has been mia for days no words from her at all

c. Stemming

Proses ini dilakukan untuk mengembalikan arti konteks pada kalimat kepada inti dari konteks aslinya. Contoh dari stemming dapat dilihat pada tabel 4.

TABEL 4
STEMMING

Input	Output
that bitch has been mia for days no words from her at all	bitch mia days no words her all

d. Tokenizing

Proses ini dilakukan untuk memisahkan setiap kata pada kalimat sehingga tidak lagi membentuk suatu kalimat. Contoh dari *tokenizing* dapat dilihat pada tabel 5.

TABEL 5
TOKENIZING

Input	Output
bitch mia days no words her all	‘bitch’, ‘mia’, ‘days’, ‘no’, ‘words’, ‘her’, ‘all’

2. Klasifikasi menggunakan algoritma BERT

Proses ini akan mengubah kalimat menjadi label dan probabilitas akurasi berdasarkan *hate speech* label.

a. Token embedding

Langkah pertama adalah menempatkan ID kosakata pada sebuah kalimat. Contoh kalimatnya adalah "I Love NLP". Kemudian token tersebut ditambahkan sehingga kalimatnya menjadi [CLS] I [MASK] NLP [SEP]. Token [CLS] akan memberi tahu algoritma bahwa itu adalah awal kalimat, sedangkan token [MASK] akan memberi tahu algoritma bahwa token itu digunakan untuk memprediksi kata yang cocok dan berhubungan dengan kata lain dalam sebuah kalimat, dan Token [SEP] akan memberi tahu algoritme bahwa token itu sendiri digunakan untuk menandai akhir kalimat.

TABEL 6
CONTOH TOKEN EMBEDDING

I Love NLP				
$E_{[CLS]}$	E_I	$E_{[MASK]}$	E_{NLP}	$E_{[SEP]}$

b. Sentence embedding

Sentence embedding akan menempatkan kelas numerik untuk membedakan antara kalimat A dan kalimat B. Sehingga kalimat tersebut menjadi:

TABEL 7
CONTOH SENTENCE EMBEDDING

I Love NLP				
E_A	E_A	E_A	E_A	E_A

c. Transformer positional encoding

Transformer positional encoding akan menempatkan lokasi setiap kata pada sebuah kalimat. Kalimat tersebut akan menjadi:

TABEL 8
CONTOH TRANSFORMER POSITIONAL

I Love NLP				
E_0	E_1	E_2	E_3	E_4

d. Transformer encoder

Misalkan nilai random embedding adalah sebagai berikut:

I Love NLP Positional Encoder

$$\begin{bmatrix}0.1 & -0.5\\ 0.6 & 0.9\\ 0.8 & 0.6\end{bmatrix} \quad \begin{bmatrix}0.0 & 0.1\\ 0.84 & 0.54\\ 0.12 & -0.41\end{bmatrix}$$

Maka untuk menghasilkan *output embedding* maka input dan *positional encoder* akan dijumlahkan sebagai berikut:

$$\begin{bmatrix}0.1 & -0.5\\ 0.6 & 0.9\\ 0.8 & 0.6\end{bmatrix} + \begin{bmatrix}0.0 & 0.1\\ 0.84 & 0.54\\ 0.12 & -0.41\end{bmatrix} = \begin{bmatrix}0.1 & -0.4\\ 1.44 & 1.44\\ 0.12 & 0.19\end{bmatrix}$$

Hasil diatas merupakan Misalkan *Queries (Q), Keys (K), Values (V)* dan nilai d adalah sebagai berikut:

Q K V d

$$\begin{bmatrix}0.71 & 0.30\\ 0.38 & 0.22\\ 0.34 & 0.36\end{bmatrix} \quad \begin{bmatrix}0.71 & 0.69\\ 0.59 & 0.74\\ 0.71 & 0.52\end{bmatrix} \quad \begin{bmatrix}0.58 & 0.53\\ 0.76 & 0.83\\ 0.89 & 0.78\end{bmatrix} \quad 2$$

Berdasarkan persamaan 1 maka ketiga nilai tersebut akan dilakukan perhitungan dan nilai K akan di *transpose* menggunakan formula berikut untuk mendapatkan nilai *softmax*:

$$softmax = \left(\frac{QK^T}{\sqrt{d}}\right)V \qquad (1)$$

Keterangan:

Q : queries
K : keys
d : embedding
V : values

$$K^T = \begin{bmatrix}0.71 & 0.59 & 0.71\\ 0.69 & 0.74 & 0.52\end{bmatrix}$$

$$QK^T = \begin{bmatrix}0.71 & 0.64 & 0.66\\ 0.42 & 0.38 & 0.38\\ 0.48 & 0.46 & 0.42\end{bmatrix}$$

$$\frac{QK^T}{\sqrt{d}} = \begin{bmatrix}0.50 & 0.45 & 0.47\\ 0.3 & 0.27 & 0.27\\ 0.34 & 0.32 & 0.3\end{bmatrix}$$

$$\frac{QK^T}{\sqrt{d}} \text{ x } V = \begin{bmatrix}1 & 1\\ 0.6 & 0.5\\ 0.7 & 0.6\end{bmatrix}$$

Maka hasil akhir dari perhitungan atas adalah:

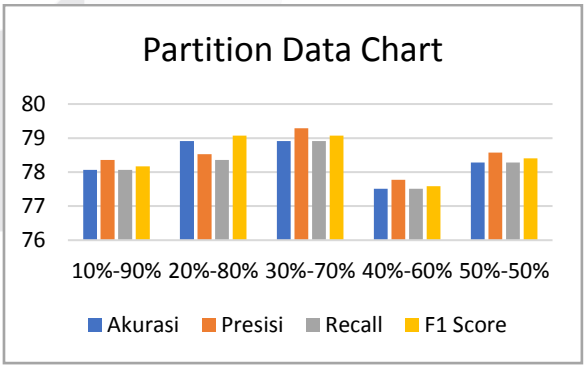
$$\begin{bmatrix}1 & 1\\ 0.6 & 0.5\\ 0.7 & 0.6\end{bmatrix}$$

Hasil tersebut merupakan nilai *output* dalam bentuk probabilitas. Perhitungan ini akan kembali berulang pada setiap *head* pada *multi head*.

IV. HASIL DAN PEMBAHASAN

A. Data Partition Test

Pada skenario ini dilakukan distribusi data *testing* dan data *training* dengan rasio yang berbeda. Skenario *default* dalam pengujian ini adalah epoch 15, batch size 32, dan 0,00003. Skenario partisi data akan diuji menjadi lima skenario, yaitu *train* 90% dengan *test* 10%, *train* 80% dengan *test* 20%, *train* 70% dengan *test* 30%, *train* 60% dengan *test* 40%, dan *train* 50% dengan *test*. uji 50%. Gambar 6 menunjukkan hasil pengujian partisi data.

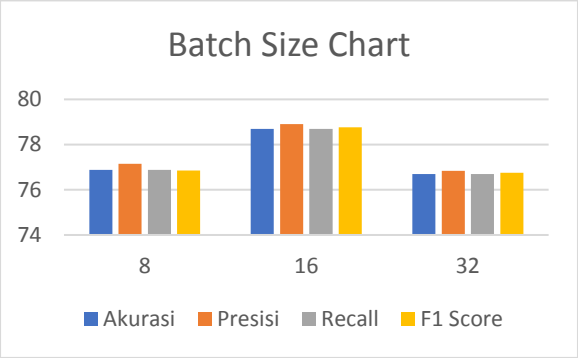


GAMBAR 6
GRAFIK PARTISI DATA

B. Batch Size Test

Setelah mempartisi data dan mendapatkan hasil yang optimal, langkah selanjutnya adalah skenario pengujian ukuran *batch* dengan mengubah nilai ukuran batch. Skenario akan diuji menjadi tiga skenario, yaitu 8 *batch size*, 16 *batch size*, dan 32 *batch*

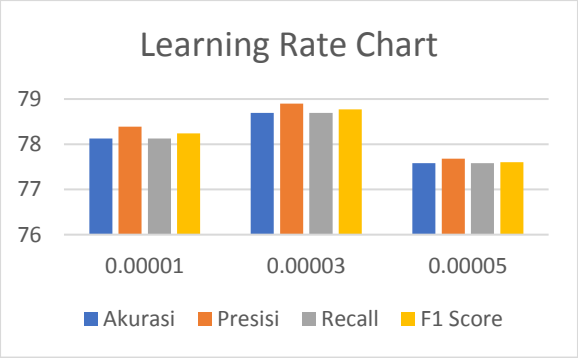
size. Gambar 7 menunjukkan 4 hasil pengujian ukuran batch.



GAMBAR 7
GRAFIK BATCH SIZE

C. Learning Rate Testing

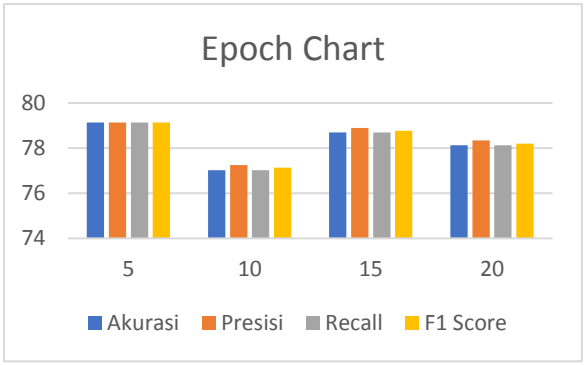
Skenario pengujian dilakukan dengan mengubah nilai *learning rate* pada pelatihan. Skenario ini didasarkan pada nilai partisi data dan ukuran batch terbaik. Skenario *learning rate* akan diuji dalam tiga skenario yaitu, 0,00001, 0,00003, dan 0,00005. Gambar 8 menunjukkan hasil tes learning rate.



GAMBAR 8
GRAFIK LEARNING RATE

D. Epoch Testing

Setelah dilakukan pengujian *learning rate* yang optimal, selanjutnya adalah skenario pengujian *epoch* dengan mengubah jumlah *epoch*. Pengujian *epoch* akan diuji ke dalam empat skenario yaitu, 5 *epoch*, 10 *epoch*, 15 *epoch*, dan 20 *epoch*. Gambar 9 menunjukkan hasil *epoch test*.



GAMBAR 9
GRAFIK EPOCH

E. Kesimpulan Testing

Hasil performansi paling optimal terletak pada model dengan hasil 30% *testing data partition* dan 70% *training data*, *batch size* 16, *learning rate* 0,00003, dan *epoch* 15. Hasil akhir pada dataset bahasa Inggris akurasi 78,69%, 78,90% presisi, recall sebesar 78,69%, dan skor F1 sebesar 78,77% terhadap klasifikasi kelompok ujaran kebencian.

V. KESIMPULAN

Berdasarkan hasil pengujian dan analisis yang telah dilakukan, maka dapat disimpulkan sebagai berikut:

1. Model BERT mempelajari kategori ujaran kebencian berdasarkan pelabelan pada kumpulan data yang divalidasi oleh penerjemah bahasa Inggris.

2. Hasil performansi paling optimal terdapat pada model dengan hasil partisi data *testing* 30% dan data *training* 70%, *batch size* 16, *learning rate* 0,00003, dan *epoch* 15. Hasil akhir pada dataset bahasa Inggris adalah akurasi 78,69%, Presisi 78,90%, recall 78,69%, dan skor F1 78,77% terhadap klasifikasi kelompok ujaran kebencian. Selain itu, semakin banyak data pelatihan, model memiliki wawasan yang lebih dalam tentang data yang dipelajari dan penggunaan transformator dan softmax dalam menghasilkan probabilitas dalam model.

REFERENSI

[1] T. A. Cahyanto, V. Wahanggara and D. Ramdana, "Analisis dan Deteksi Malware Menggunakan Metode Malware Analisis Dinamis dan Malware Analisis Statis," *Jurnal Sistem & Teknologi Informasi Indonesia,* vol. 2, p. 19, 2017.

[2] R. Garett, L. R. Lord and S. D. Young, "Associations Between Social Media and Cyberbullying : A Review of The Literature," vol. 2, 2016.

[3] IBM Cloud Education, "IBM Cloud Learn Hub," Natural Language Processing (NLP), 2 Juli 2020. [Online]. Available: https://www.ibm.com/cloud/learn/natural-language-processing. [Accessed 20 Juli 2022].

[4] P. M. Nadkarni, L. Ohno-Machado and W. W. Chapman, "Natural Language Processing : an introduction," pp. 544-551, 2011.

[5] SAS Institute Inc., "Neural Networks : What they are & why they matter," [Online]. Available: https://www.sas.com/en_id/insights/analytics/neural-networks.html. [Accessed 12 September 2022].

[6] Pandu Nayak, "Understanding searches better than ever before," Google, 25 Oktober 2019. [Online]. Available: https://blog.google/products/search/search-language-understanding-bert/. [Accessed 30 November 2021].

[7] B. Muller, "huggingface bert 101," BERT 101 STATE OF THE ART NLP MODEL EXPLAINED, 2 Maret 2022. [Online]. Available: https://huggingface.co/blog/bert-101. [Accessed 28 Agustus 2022].

[8] D. Gunawan, C. A. Sembiring and M. A. Budiman, "The Implementation of Cosine Similarity to Calculate Text Relevance between Two Documents," *Journal of Physics : Conference Series,* pp. 1-6, 2017.