Jurnal Ilmu Siber dan Teknologi Digital (JISTED)
Vol 1, No 1, 2022, 1-28 https://doi.org/10.35912/jisted.v1i1.1509

Model Klasifikasi Berbasis *Multiclass Classification* dengan Kombinasi Indobert Embedding dan Long Short-Term Memory untuk *Tweet* Berbahasa Indonesia (*Classification Model Based on Multiclass Classification with a Combination of Indobert Embedding and Long Short-Term Memory for Indonesian-language Tweets*)

Thariq Iskandar Zulkarnain Maulana Putra[1]*, Suprapto[2], Arif Farhan Bukhori[3]
Program Studi Ilmu Komputer, Departemen Ilmu Komputer dan Elektronika, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Gadjah Mada, Yogyakarta[1,2,3]
thariqiskandar9@gmail.com[1,2,3]

Riwayat Artikel
Diterima pada 5 Oktober 2022
Revisi 1 pada 15 Oktober 2022
Revisi 2 pada 28 Oktober 2022
Disetujui pada 11 November 2022

Abstract
Purpose: This research aims to improve the performance of the text classification model from previous studies, by combining the IndoBERT pre-trained model with the Long Short-Term Memory (LSTM) architecture in classifying Indonesian-language tweets into several categories.
Method: The classification text based on multiclass classification was used in this research, combined with pre-trained IndoBERT namely Long Short-Term Memory (LTSM). The dataset was taken using crawling method from API Twitter. Then, it will be compared with Word2Vec-LTSM and fined-tuned IndoBERT.
Result: The IndoBERT-LSTM model with the best hyperparameter combination scenario (batch size of 16, learning rate of 2e-5, and using average pooling) managed to get an F1-score of 98.90% on the unmodified dataset (0.70% increase from the Word2Vec-LSTM model and 0.40% from the fine-tuned IndoBERT model) and 92.83% on the modified dataset (4.51% increase from the Word2Vec-LSTM model and 0.69% from the fine-tuned IndoBERT model). However, the improvement from the fine-tuned IndoBERT model is not very significant and the Word2Vec-LSTM model has a much faster total training time.
Keywords *Text Classification, Indonesian Tweets, IndoBERT, Long Short-Term Memory*
How to cite: Putra, T.I.Z.M., Suprapto, S., Bukhori, A.F. (2022). Model Klasifikasi Berbasis *Multiclass Classification* dengan Kombinasi Indobert Embedding dan Long Short-Term Memory untuk *Tweet* Berbahasa Indonesia. *Jurnal Ilmu Siber dan Teknologi Digital*, 1(1), 1-28.

1. Pendahuluan

Twitter merupakan salah satu situs media sosial yang sedang berkembang pesat dengan lebih dari 3,7 juta pengguna aktif mem-*posting* sekitar 10 juta *tweet* per hari (Ayo et al., 2020). Selain digunakan untuk *update* status, Twitter juga digunakan sebagai platform penyebaran informasi berbagai topik yang cukup akurat dan terpercaya. Pencarian *tweet* pada aplikasi Twitter dapat menggunakan fitur *search* dengan mengetikkan kata kunci atau *hashtag*. Akan tetapi, penggunaan kata kunci atau *hashtag* terkadang kurang akurat ketika menggunakan kata yang memiliki beberapa arti, sehingga, perlu dilakukan pemberian kategori terhadap *tweet* berdasarkan konteksnya untuk menghindari adanya bias pada kata yang memiliki beberapa arti serta meningkatkan *ranking* pada hasil pencarian Google.

Pemrosesan bahasa alami (*Natural Language Processing* - NLP) merupakan cabang dari kecerdasan buatan (*Artificial Intelligence*) yang memberikan kemampuan pada komputer untuk memahami teks dan kata-kata yang diucapkan dengan cara yang sama seperti yang dapat dilakukan manusia. Klasifikasi teks merupakan salah satu tugas NLP yang dapat memberikan kategori terhadap teks secara otomatis berdasarkan konteks dari teks tersebut dengan bantuan metode *machine learning* maupun *deep learning*. Klasifikasi teks menjadi area penelitian yang sering muncul dalam pemrosesan bahasa alami karena meningkatnya jumlah unggahan pengguna di berbagai jejaring sosial (Alwehaibi et al., 2021). Proses pengklasifikasian teks dapat terbagi menjadi dua jenis, yaitu *binary classification* dan *multiclass classification*.

Model *pre-trained word embedding* merupakan sebuah model *word embedding* yang telah dilatih pada *dataset* yang berukuran besar dan *general*, agar memiliki pemahaman semantik maupun sintaksis yang lebih baik. Pada tahun 2018, Devlin et al. mengusulkan sebuah model, yaitu *Bidirectional Encoder Representations from Transformer* (BERT) yang berhasil mendapatkan performa *state-of-the-art* pada banyak studi terkait NLP. BERT menggunakan *Transformer* yang merupakan mekanisme yang mempelajari hubungan kontekstual antara kata-kata dalam teks menggunakan *self-attention mechanism* (Vaswani et al., 2017). Khusus untuk bahasa Indonesia, Koto et al. (2020) berhasil mengembangkanmodel *pre-trained* BERT yang bernama IndoBERT.

Deep learning merupakan metode pembelajaran mesin yang terinspirasi oleh cara kerja sistem saraf otak manusia. Sistem ini dinamakan Jaringan Syaraf Tiruan (*Artificial Neural Network* - ANN). Pada model klasifikasi teks, vektor kata yang dihasilkan dari proses *word embedding* dapat dijadikan sebagai *input* pada lapisan *neural network* yang diklasifikasikan berdasarkan informasi yang dipelajari. *Long Short-Term Memory* (LSTM) merupakan modifikasi dari arsitektur *Recurrent Neural Network* (RNN) yang dapat mengatasi masalah *vanishing gradient* saat memproses data *sequential* yang panjang.

Telah ditemukan beberapa penelitian mengenai pengembangan model klasifikasi teks pada *dataset* berbahasa Indonesia. Koto et al. (2020) melakukan *fine-tuning* pada model yang dia kembangkan, yaitu IndoBERT untuk tugas analisis sentimen dan berhasil mendapatkan F1-*score* sebesar 84,13%. Muhammad et al. (2021) juga mencoba mengembangkan model analisis sentimen menggunakan Word2Vec dan *Long Short-Term Memory* (LSTM) dengan akurasi mencapai 85,96%. Di sisi lain, Hilmiaji et al. (2021) mencoba mengidentifikasi emosi dari *tweet* berbahasa Indonesia ke dalam 5 kelas menggunakan *word embedding* dari *library* Keras dengan arsitektur CNN dan berhasil mendapatkan F1-*score* sebesar 90,2%. Sedangkan Ramadhan (2021) mencoba mengklasifikasikan berita *online* Indonesia berdasarkan 4 topik yang sedang populer menggunakan Word2Vec dan *K-Nearest Neighbor* dengan akurasi 89,2%.

Beberapa penelitian yang telah disebutkan sebelumnya telah berhasil mengembangkan model klasifikasi teks baik yang berbasis *binary classification* maupun *multiclass classification*, khususnya pada *dataset* berbahasa Indonesia. Akan tetapi, model yang dikembangkan masih memiliki potensi untuk ditingkatkan akurasinya dengan menerapkan model *pre-trained word embedding* yang memiliki performa *state-of-the-art* dan mengombinasikannya dengan arsitektur *neural network*. Oleh karena itu, pada penelitian ini diusulkan model klasifikasi teks yang mengombinasikan model *pre-trained* IndoBERT dengan salah satu arsitektur *Recurrent Neural Network* (RNN), yaitu *Long Short-Term Memory* (LSTM), dalam mengklasifikasikan *tweet* berbahasa Indonesia ke beberapa kategori sesuai dengan konteksnya.

1.1 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah diuraikan sebelumnya bahwa sudah terdapat beberapa penelitian tentang pengembangan model klasifikasi teks pada *dataset* berbahasa Indonesia. Akan tetapi, model-model yang sudah dikembangkan masih mungkin untuk ditingkatkan akurasinya. Oleh karena itu, pada penelitian ini diusulkan sebuah model klasifikasi teks berbasis *multiclass classification* untuk *tweet* berbahasa Indonesia yang mengombinasikan model *pre-trained* IndoBERT

dengan salah satu arsitektur *Recurrent Neural Network* (RNN), yaitu *Long Short-Term Memory* (LSTM).

1.2 Tujuan Penelitian

Penelitian ini bertujuan untuk meningkatkan performa model klasifikasi teks yang telah dikembangkan pada penelitian sebelumnya, dengan menggunakan kombinasi dari model *pre-trained* IndoBERT dengan arsitektur *Long Short-Term Memory* (LSTM) dalam mengklasifikasikan *tweet* berbahasa Indonesia kebeberapa kategori sesuai dengan konteksnya.

Researchers in both advanced and developing economies have produced results when discussing liquidity ratio and profitability ratio in Nigeria. Duruechi et al (2016), Bassey and Moses (2015), and Edem (2017) all looked at liquidity management and performance from a macroeconomic viewpoint in Nigeria, with minimal attention paid to the pharmaceutical industry. Even in a few research that looked at other sectors, such as Kehinde (2013), and Idowu, et al, (2017), there were contradicting and varied results.

2. Tinjauan Pustaka dan Pengembangan Hipotesis

2.1 Klasifikasi Teks

Klasifikasi teks menjadi area penelitian yang sering muncul dalam pemrosesan bahasa alami karena meningkatnya jumlah unggahan pengguna di berbagai jejaring sosial (Alwehaibi et al., 2021). Klasifikasi teks merupakan proses pemberian kategori ke teks berdasarkan isi / topik dari teks tersebut. Model klasifikasi teks dapat terbagi menjadi dua jenis, yaitu *binary classification* dan *multiclass classification*. Sebagian besar sistem klasifikasi teks dan kategorisasi dokumen dapat didekonstruksi menjadi empat fase, yaitu ekstraksi fitur, pengurangan dimensi, pemilihan pengklasifikasi, dan evaluasi.

2.2 Text Preprocessing

Text preprocessing merupakan salah satu tahapan penting pada pengembangan model klasifikasi teks, karena data teks yang diambil melalui proses *text mining* tidak selamanya dalam kondisi yang ideal dan terstruktur untuk diproses. Dalam banyak algoritma, terutama algoritma pembelajaran statistik dan probabilistik, *noise* dan fitur yang tidak perlu dapat memiliki efek buruk pada kinerja sistem. Sehingga, diperlukan sebuah proses yang dapat mengubah data teks menjadi lebih terstruktur melalui beberapa metode, meliputi *case folding*, *noise removal*, penggantian slang dan singkatan, *tokenizing, stemming*, dan *stopwords removal*.

2.2.1 Case Folding

Data teks tidak terlepas dari keberagaman kapitalisasi dalam membentuk sebuah kalimat. Keberagaman kapitalisasi ini dapat menjadi masalah besar saat mengklasifikasikan teks berukuran besar. Kapitalisasi yang tidak konsisten dapat diatasi dengan mengubah setiap huruf kapital menjadi huruf kecil. Teknik ini memproyeksikan semua kata dalam teks ke dalam ruang fitur yang sama (Gupta & Lehal Professor, 2009).

2.2.2 Noise Removal

Noise removal merupakan proses untuk menghilangkan tanda baca atau karakter selain teks. Tanda baca dan karakter khusus penting untuk pemahaman manusia tentang dokumen, tetapi dapat merusak algoritma klasifikasi (Pahwa et al., 2018).

2.2.3 Slang dan Singkatan

Slang dan singkatan merupakan salah satu anomali teks yang juga perlu ditangani pada tahapan *preprocessing*. Slang adalah jenis bahasa yang sering digunakan dalam percakapan informal namun memiliki sifat yang fleksibel (Sun et al., 2021). Proses untuk mengganti slang dan singkatan akan dibantu dengan kamus dari Taudata Analytics sebanyak 1682 kata (Sutanto, 2020).

2.2.4 Tokenizing

Tokenizing adalah prosedur pemecahan teks menjadi kata, frasa, atau bagian lain yang bermakna, yaitu token (Uysal & Gunal, 2014). Dengan kata lain, *tokenizing* adalah proses segmentasi teks, yang biasanya dilakukan dengan mempertimbangkan hanya karakter alfabet atau alfanumerik yang dibatasi oleh karakter non-alfanumerik (tanda baca dan spasi). Metode ini bertujuan sebagai penyelidikan kata-kata dalam sebuah kalimat. Proses *tokenizing* dapat dilakukan dengan menggunakan fungsi *word_tokenize* yang disediakan oleh *libary* NLTK. Proses *tokenizing* pada BERT dilakukan dengan menggunakan metode WordPiece, di mana setiap kalimat akan ditokenisasi menjadi per kata atau sub kata.

2.2.5 Stemming

Satu kata dapat muncul dalam berbagai bentuk, namun memiliki makna semantik yang sama. Pada kasus bahasa Indonesia, variasi bentuk kata muncul akibat adanya penambahan imbuhan (awalan dan akhiran) pada kata dasar, seperti kata "makan" ditambah dengan awalan "me-" akan menjadi "memakan" atau ditambah dengan akhiran "-an" akan menjadi "makanan". Sehingga diperlukan sebuah metode yang dapat menggabungkan berbagai bentuk kata ke dalam ruang fitur yang sama. Salah satu metode yang dapat digunakan adalah *stemming*, yang bertujuan untuk mendapatkan bentuk kata dasar dari variasi kata turunannya. Proses *stemming* untuk bahasa Indonesia dilakukan dengan menggunakan *library* Sastrawi (Robbani, 2018).

2.2.6 Stopwords Removal

Stopwords adalah kata-kata yang biasa ditemui dalam teks tanpa ketergantungan pada topik tertentu (konjungsi, preposisi, artikel, dll) (Uysal & Gunal, 2014). *Stopwords* memiliki frekuensi kemunculan yang tinggi. Oleh karena itu, keberadaan mereka dianggap tidak relevan dalam studi klasifikasi teks. Permasalahan ini dapat diatasi dengan menghilangkan *stopwords* dari teks tersebut. Namun, setiap bahasa memiliki daftar *stopwords* yang berbeda. Proses *stopwords removal* untuk bahasa Indonesia dapat menggunakan *library* Sastrawi (Robbani, 2018).

2.3 Word Embedding

Word embedding adalah salah satu poin paling penting untuk studi pemrosesan teks dan *input* paling penting untuk jaringan (Aydoğan & Karci, 2020). *Word embedding* adalah teknik pembelajaran fitur di mana setiap kata atau frasa dari kosakata dipetakan ke dalam vektor bilangan real berdimensi N. Fokus dari metode ini terletak pada penetapan vektor yang mirip dengan kata-kata yang memiliki arti yang serupa secara semantik (Goyal et al., 2021).

2.3.1 BERT

Bidirectional Encoder Representations from Transformer (BERT) pertama kali diperkenalkan pada tahun 2018 oleh Devlin et al. yang merupakan peneliti dari *Google AI Language*. Sesuai dengan namanya, BERT menggunakan *Transformer* yang merupakan mekanisme yang mempelajari hubungan kontekstual antara kata-kata dalam teks menggunakan *self-attention mechanism* (Vaswani et al., 2017). *Self-attention mechanism* memungkinkan *input* untuk berinteraksi satu sama lain (*self*) dan mencari tahu siapa yang harus diberi perhatian lebih (*attention*). Representasi urutan kata dari sebuah kalimat dihitung dengan menghubungkan kata-kata yang berbeda dalam urutan yang sama menggunakan mekanisme *encoder* dan *decoder*.

3.2.2 Word2Vec

Mikolov et al. (2013) mengusulkan model yang merepresentasikan "*word to vector*" sebagai arsitektur *word embedding* yang ditingkatkan dari model *Neural Network Language Model* (NNLM). Pendekatan Word2Vec menggunakan *shallow neural network* dengan dua *hidden layer*. Terdapat dua arsitektur yang berbeda pada Word2Vec, yaitu *Continuous* Bag-of-Words (CBOW), dan *Continuous* Skip-gram untuk membuat vektor berdimensi tinggi dari setiap kata.

***Continuous* Bag-of-Words**

Pada model *Continuous* Bag-of-Words, representasi terdistribusi dari konteks (kata-kata di sekitarnya) digabungkan untuk memprediksi kata di tengah. Misalnya, kata "uang" dan "nasabah" sebagai konteks untuk kata target "bank".

***Continuous* Skip-gram**

Arsitektur model lain yang sangat mirip dengan CBOW adalah model *Continuous* Skip-gram. Namun, alih-alih memprediksi kata saat ini berdasarkan konteksnya, ia mencoba memaksimalkan klasifikasi kata berdasarkan kata lain dalam kalimat yang sama. Skip-gram memiliki performa yang lebih baik dari CBOW pada sebagian besar evaluasi, tapi CBOW lebih cepat untuk dilatih (Mikolov et al., 2013).

2.4 Recurrent Neural Network (RNN)

Ide dasar dari *Recurrent Neural Network* (RNN) adalah membuat topologi jaringan yang mampu merepresentasikan data *sequential*. RNN berfokus pada sifat data di mana *instance* waktu sebelumnya (t–1) mempengaruhi *instance* pada waktu berikutnya (t). Secara lebih umum, diberikan sebuah urutan *input* $x = (x_1, x_2, \dots, x_t)$. Data x_t (i.e., vektor, gambar, teks, suara) dipengaruhi oleh data sebelum-sebelumnya (*history*) yang ditulis sebagai $P(x_t|x_1, x_2, \dots, x_{t-1})$.

2.4.1 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) pertama kali diperkenalkan oleh Hochreiter dan Schmidhuber (1997). LSTM merupakan modifikasi dari arsitektur RNN dengan menambahkan *memory cell* yang dapat menyimpan informasi untuk jangka waktu yang lama. Arsitetur ini diusulkan sebagai solusi dalam mengatasi masalah *vanishing gradient* pada RNN saat memproses data *sequential* yang panjang.

2.5 Hugging Face

Hugging Face merupakan *open-source library* untuk berbagai macam aplikasi NLP (Chaumond et al., 2016). Hugging Face menyediakan banyak model untuk kebutuhan NLP yang telah dikemas dan dapat secara langsung digunakan untuk pemodelan.

2.6 TensorFlow

TensorFlow (TF) merupakan *open source library* yang sangat populer untuk pengembangan *machine learning* berskala besar (Google Brain Team, 2015). TensorFlow mengemas model *machine learning* dan *deep learning* beserta algoritmanya yang dapat digunakan untuk berbagai kebutuhan. TensorFlow menggunakan Python sebagai *front-end* API-nya serta mengeksekusi aplikasinya menggunakan bahasa pemrograman C++.

2.7 Scikit-learn

Scikit-learn merupakan *machine learning library* untuk bahasa pemrograman Python yang dapat digunakan secara gratis (Cournapeau, 2007). Scikit-learn memiliki banyak fitur, seperti pemrosesan data, berbagai algoritma klasifikasi, regresi, dan *clustering*, serta evaluasi model. Scikit-learn didesain untuk dapat dioperasikan bersama *library* NumPy, *numerical* dan *scientific library* milik Python.

2.8 Hyperparameter

Hyperparameter adalah parameter dari algoritma pembelajaran yang tidak terpengaruh oleh algoritma pembelajaran itu sendiri (Géron, 2017). *Hyperparameter* harus diatur sebelum pelatihan dan tetap konstan selama pelatihan. Melakukan *hyperparameter tuning* merupakan tahapan penting dalam membangun model *machine learning* maupun *deep learning*. Hal ini dilakukan agar didapatkan model dengan performa optimal.

2.8.1 Epoch

Epoch merupakan *hyperparameter* yang menentukan berapa kali *neural network* melakukan proses pelatihan terhadap seluruh *dataset*. Satu *epoch* artinya ketika seluruh *dataset* sudah melalui proses

pelatihan pada *neural network* sampai dikembalikan lagi ke awal (Digmi, 2018). Proses pelatihan model tidak dapat dilakukan hanya dengan menggunakan satu *epoch*. Hal ini dikarenakan *dataset* yang digunakan terbatas dan untuk mengoptimalkan grafik *gradient descent* perlu adanya proses iteratif. Penentuan jumlah *epoch* bergantung pada keberagaman data pada *dataset* yang dimiliki.

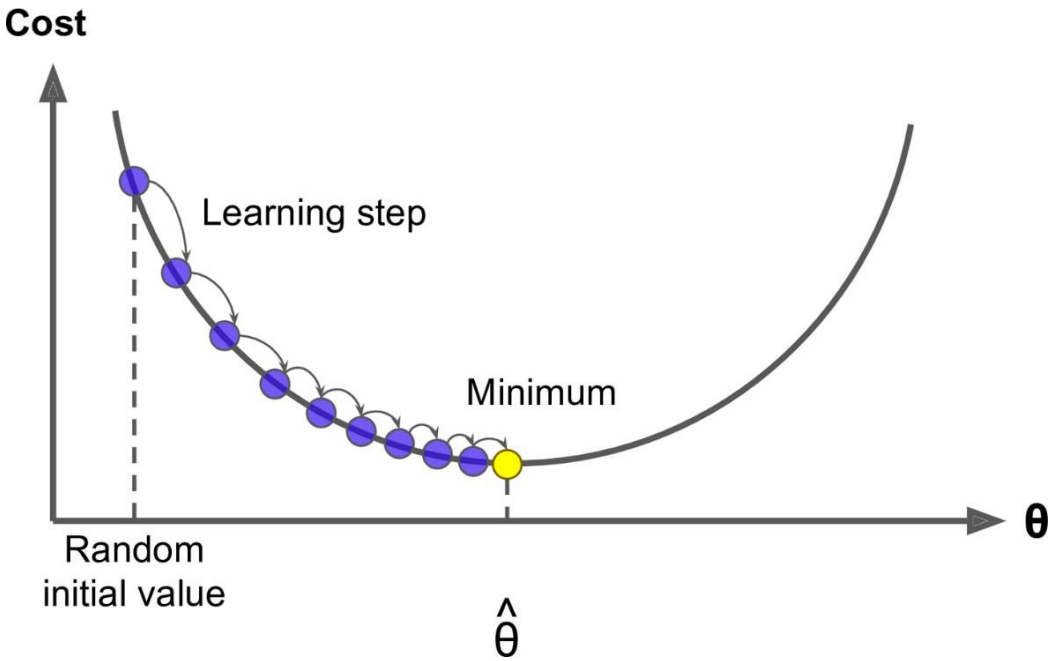
2.8.2 Batch Size

Batch size adalah jumlah sampel yang dimasukkan ke dalam *neural network* sebelum bobot disesuaikan. Pada akhir *batch*, prediksi dibandingkan dengan variabel *output* yang diharapkan untuk dihitung *error*-nya. Dari *error* ini, dilakukan pembaruan bobot untuk memperbaiki model dengan menggunakan algoritma *back-propagation* yang bergerak mundur dari *layer* terakhir menuju *layer* pertama. *Gradient descent* memiliki 3 variasi berbeda berdasarkan *batch size* yang digunakan untuk melakukan proses *update* bobot (Géron, 2017), yaitu:

1. *Batch Gradient Descent*
   Batch gradient descent menggunakan *batch size* yang sama dengan ukuran *dataset* pelatihan. Sehingga proses *update* bobot hanya dilakukan sekali setelah seluruh proses *forward-propagation* selesai.
2. *Stochastic Gradient Descent*
   Stochastic gradient descent adalah proses pembelajaran yang melakukan *update* untuk setiap 1 data.
3. *Mini-batch Gradient Descent*
   Mini-batch gradient descent menggunakan *batch size* sebesar 2 pangkat m. Dipilih faktor 2 jumlah data karena memori memiliki ukuran faktor 2, sehingga juga dapat mengoptimalkan memori yang terpakai.

2.8.3 Learning Rate

Learning rate digunakan untuk menentukan seberapa banyak bobot pada *neural network* yang akan diubah. Ukuran *learning rate* merupakan salah satu *hyperparameter* yang berpengaruh dalam tercapainya solusi optimal dari *gradient descent* seperti yang ditunjukkan oleh Gambar 3.8 (Géron, 2017).



Gambar 3.8 Gradient Descent dengan Solusi Optimal (Géron, 2017)

2.8.4 Probabilitas Dropout

Dropout merupakan proses menetapkan unit *input* di dalam jaringan menjadi 0 secara acak dengan probabilitas antara 0 dan 1 (Keras Team, 2015). Unit *input* yang tidak ditetapkan menjadi 0

ditingkatkan sebesar $1/(1 - probabilitas)$, sehingga jumlah dari semua *input* tidak berubah. Proses ini dapat mencegah terjadinya *overfitting* dan juga mempercepat proses pelatihan.

2.8.5 Metode Pooling

Pooling layer digunakan untuk mengurangi *input* dari perspektif spasial serta memfasilitasi pengurangan jumlah parameter jaringan, sehingga meningkatkan kecepatan komputasi dan juga mencegah *overfitting* (Muhammad et al., 2021).

2.8.6 Activation Function

Activation function merupakan fungsi dalam *neural network* yang mendefinisikan bagaimana *weighted sum* dari *input* diubah menjadi *output* dari *node* di setiap lapisan *neural network* (Brownlee, 2021). Beberapa *activation function* yang sering digunakan, yaitu:

1. Rectified Linear Activation (ReLU)

Rectified Linear Activation (ReLU) merupakan *activation function* yang sering digunakan pada *hidden layer* karena mudah diimplementasikan dan efektif dalam mengatasi keterbatasan dari *activatiom function* populer lainnya, seperti *Sigmoid* dan Tanh.

2. Logistic (*Sigmoid*)

Sigmoid activation function disebut juga *logistic function* yang digunakan dalam algoritma klasifikasi *logistic regression*. Fungsi ini mengambil sembarang nilai real sebagai nilai *input* dan *output* dalam rentang 0 hingga 1. Semakin besar nilai *input* (semakin positif), maka semakin dekat nilai *output* ke 1. Sedangkan semakin kecil nilai *input* (semakin negatif), maka semakin dekat nilai *output* menjadi 0. Fungsi *sigmoid* secara matematis dirumuskan oleh persamaan (3.10),

$$\sigma(\mathrm{x})\,\frac{1}{1+e^{-x}} \tag{3.10}$$

di mana e merupakan konstanta matematika, yang merupakan basis dari logaritma natural.

Pada LSTM *layer*, fungsi *sigmoid* digunakan sebagai *activation function* untuk *forget gate* (persamaan (3.3)), *input gate* (persamaan (3.4)), dan *output gate* (persamaan (3.7)). Sedangkan pada *output layer*, fungsi *sigmoid* dapat bekerja dengan baik pada tugas *binary classification* karena kelas target hanya akan memiliki nilai 0 atau 1.

3. Hyperbolic Tangent (Tanh)

Hyperbolic tangent activation function juga disebut sebagai fungsi tanh, yang sangat mirip dengan fungsi *sigmoid*. Bedanya, fungsi ini mengambil sembarang nilai real sebagai nilai *input* dan *output* dalam rentang -1 hingga 1. Semakin besar nilai *input* (semakin positif), maka semakin dekat nilai *output* menjadi 1. Sedangkan semakin kecil nilai *input* (semakin negatif), maka semakin dekat nilai *output* menjadi -1. Fungsi tanh secara matematis dirumuskan oleh persamaan (3.11),

$$\sigma(\mathrm{x})\,\frac{(e^{x}-e^{-x})}{(e^{x}+e^{-x})} \tag{3.11}$$

di mana e merupakan konstanta matematika, yang merupakan basis dari logaritma natural.

Pada LSTM *layer*, fungsi tanh digunakan sebagai *activation function* untuk *candidate cell state* (persamaan (3.5)) dan *final output gate* (persamaan (3.8)).

4. Linear

Linear activation function juga disebut "identitas" (dikalikan dengan 1) atau "*no activation*" karena fungsi ini tidak mengubah *weighted sum* dari *input* dengan cara apa pun dan mengembalikan nilai secara langsung.Fungsi *linear* secara matematis dirumuskan oleh persamaan (3.12).

$$L(x) = x \tag{3.12}$$

5. Softmax

Softmax activation function mengeluarkan nilai vektor berjumlah 1 yang dapat diinterpretasikan sebagai probabilitas keanggotaan kelas. Fungsi ini mirip dengan fungsi *argmax* yang menghasilkan 0 untuk semua kelas dan 1 untuk kelas yang dipilih. *Softmax* merupakan "*softer*" *version* dari fungsi *argmax* yang memungkinkan *output* dari setiap kelas memiliki nilai probabilitas yang apabila dijumlahkan akan berjumlah 1. Fungsi *softmax* secara matematis dirumuskan oleh persamaan (3.13),

$$s(x)_i = \frac{\exp x_i}{\sum_{j=1}^{n} \exp(x_j)} \quad (3.13)$$

di mana x merupakan vektor *input*, exp () merupakan fungsi eksponensial standar, x_i adalah vektor *input* pada elemen ke-i, x_j vektor *input* pada elemen ke-j yang akan dijumlahkan hasil perhitungan eksponensialnyahingga elemen ke-n, dan n adalah jumlah kelas.

Fungsi softmax dapat bekerja dengan baik pada output layer untuk tugas multiclass classification, karena dapat menghasilkan vektor dengan panjang sesuai dengan jumlah kelas dan dinormalisasi agar memiliki jumlah probabilitas sama dengan 1. Vektor ini nantinya akan dibandingkan Loss Function

2.8.7 Loss Function

Loss function pada *neural network* berperan untuk menghitung *loss* atau *error* antara nilai prediksi yang dihasilkan oleh model *machine learning* pada *output layer* dengan nilai aktual / target (Chauhan, 2021). Dari *loss* tersebut diperoleh gradien yang digunakan untuk memperbarui bobot dari setiap *layer* pada proses *back-propagation*.

Cross-entropy merupakan *loss function* yang biasa digunakan untuk skenario tugas klasifikasi. *Cross-entropy loss* juga disebut sebagai *logarithmic loss*, *log loss*, atau *logistic loss*. Nilai probabilitas dari setiap kelas yang diprediksi dibandingkan dengan kelas aktual yang diinginkan, yaitu 0 atau 1 untuk dihitung skor / *loss* yang menghukum probabilitas berdasarkan seberapa jauh dari nilai yang sebenarnya. Hukumannya bersifat logaritmik yang menghasilkan skor besar untuk perbedaan besar yang mendekati 1 dan skor kecil untuk perbedaan kecil yang mendekati 0. *Cross-entropy* secara matematis dirumuskan dengan persamaan (3.14),

$$L = -\sum_{i=1}^{M} t_i \log(p_i) \quad (3.14)$$

di mana M merupakan jumlah kelas, t_i adalah nilai aktual dari kelas ke-i, dan p_i adalah nilai probabilitas hasil prediksi dari kelas ke-i.

Terdapat 2 metode perhitungan *cross-entropy* yang berbeda untuk masing- masing permasalahan *binary classification* dan *multiclass classification*, yaitu:

1. *Binary Cross-Entropy*

Binary cross-entropy adalah *loss function* yang digunakan dalam tugas *binary classification*. Fungsi ini menjawab pertanyaan dengan hanya dua pilihan (ya atau tidak, A atau B, 0 atau 1, kiri atau kanan, dsb). Apabila jumlah M = 2, maka *binary cross-entropy* secara matematis dirumuskan oleh persamaan (3.15),

$$L = -\sum_{i=1}^{2} t_i \log(p_i) \quad (3.15)$$
$$= -[\square \log(\square) + (1 - \square) \log(1 - \square)]$$

di mana t_i adalah nilai aktual dari kelas ke-i dan p_i adalah nilai probabilitas *sigmoid* hasil prediksi dari kelas ke-i.

Sigmoid adalah satu-satunya *activation function* yang kompatibel dengan *binary cross-entropy loss function*, karena *loss function* ini perlu menghitung logaritma dari p dan $(1 - p)$ yang hanya ada jika p bernilai antara 0 dan 1.

2. *Categorical Cross-Entropy*

Categorical cross entropy adalah *loss function* yang digunakan dalam tugas *multiclass classification*. Fungsi ini didesain untuk mengukur perbedaan antara 2 distribusi probabilitas. Jika M > 2 (*multiclass classification*), dihitung *loss* terpisah untuk setiap kelas yang diamati dan dijumlahkan hasilnya. *Categorical cross-entropy* secara matematis dirumuskan oleh persamaan (3.16),

$$L = -\sum_{i=1}^{M} t_i \log(p_i) \quad (3.16)$$

di mana M merupakan jumlah kelas, t_i adalah nilai aktual dari kelas ke-i, dan p_i adalah nilai probabilitas *softmax* hasil prediksi dari kelas ke-i.

Softmax adalah satu-satunya *activation function* yang disarankan untuk digunakan dengan *categorical cross-entropy loss function*.

2.9 Confusion Matrix

Confusion matrix merupakan sebuah pengukuran performa yang sering digunakan pada masalah klasifikasi di mana *output* dapat terdiri dari dua kelas atau lebih. Terdapat empat atribut yang merupakan kombinasi dari nilai yang diprediksi (*predicted*) dan nilai yang sebenarnya (*actual*), yaitu:

1. *True Positive*: Jumlah data yang bernilai positif baik pada kategori yang diprediksi maupun kategori yang sebenarnya.
2. *False Positive*: Jumlah data yang bernilai positif pada kategori yang diprediksi tetapi bernilai negatif pada kategori yang sebenarnya.
3. *True Negative*: Jumlah data yang bernilai negatif baik pada kategori yang diprediksi maupun kategori yang sebenarnya.
4. *False Negative*: Jumlah data yang bernilai negatif pada kategori yang diprediksi tetapi bernilai positif pada kategori yang sebenarnya.

Keempat atribut tersebut akan menjadi dasar perhitungan beberapa metrikevaluasi, yaitu:

1. Accuracy

Accuracy merupakan rasio prediksi benar (positif dan negatif) dengan keseluruhan data. Metrik ini paling umum digunakan karena mudah dihitung dan digunakan. Akan tetapi, metrik ini memiliki kekurangan yaitu kurang akurat untuk data yang tidak seimbang. Nilai *accuracy* dapat diperoleh dengan persamaan (3.17).

$$accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (3.17)$$

2. Precision

Precision merupakan rasio antara *True Positive* (TP) dengan keseluruhan data yang diprediksi positif. Sehingga, *precision* berusaha memperkecil terjadinya *False Positive* (FP). Nilai *precision* dapat diperoleh denganpersamaan (3.18).

$$precision = \frac{TP}{TP+FP} \quad (3.18)$$

3. Recall

Recall merupakan rasio antara *True Positive* (TP) dengan keseluruhan data yang kenyataannya bernilai positif. Sehingga, *recall* berusaha memperkecil terjadinya *False Negative* (FN). Nilai *recall* dapat diperoleh dengan persamaan (3.19).

$$recall = \frac{TP}{TP+FP} \quad (3.19)$$

4. F1-score

F1-score merupakan *harmonic mean* dari precision dan recall. Nilai F1- *score* dapat diperoleh dengan persamaan (3.20).

$$F1\ score = \frac{2 \times precission \times recall}{precission+recall} = \frac{2TP}{2TP+FP+FN} \quad (3.20)$$

3. Metode penelitian

3.1 Deskripsi Umum Model

Pada penelitian ini, diusulkan model klasifikasi teks berbasis *multiclass classification* pada *tweet* berbahasa Indonesia yang mengombinasikan model *pre-trained* IndoBERT dengan salah satu arsitektur *Recurrent Neural Network* (RNN), yaitu *Long Short-Term Memory* (LSTM). Proses pengembangan model terdiri dari beberapa langkah utama, yaitu pembuatan *dataset*, *text preprocessing*, pembuatan arsitektur model, pelatihan, dan evaluasi model. Adapun *dataset* yang digunakan merupakan kumpulan *tweet* berbahasa Indonesia yang diambil dengan metode *crawling* dari API Twitter. Model yang telah dikembangkan akan dibandingkan performanya dengan dua *baseline model*, yaitu Word2Vec-LSTM dan *fine-tuned* IndoBERT.

3.2 Pembuatan Dataset

Dataset yang digunakan pada penelitian ini berupa data *tweet* berbahasa Indonesia. Data tersebut didapatkan melalui proses *crawling* pada Twitter. Implementasi *crawling* pada Twitter memerlukan API *Key* yang terdaftar untuk dapat berinteraksi dengan Twitter. *Crawling* dilakukan menggunakan bahasa pemrograman Python dan *library* Tweepy, serta menggunakan metode API *Search*.

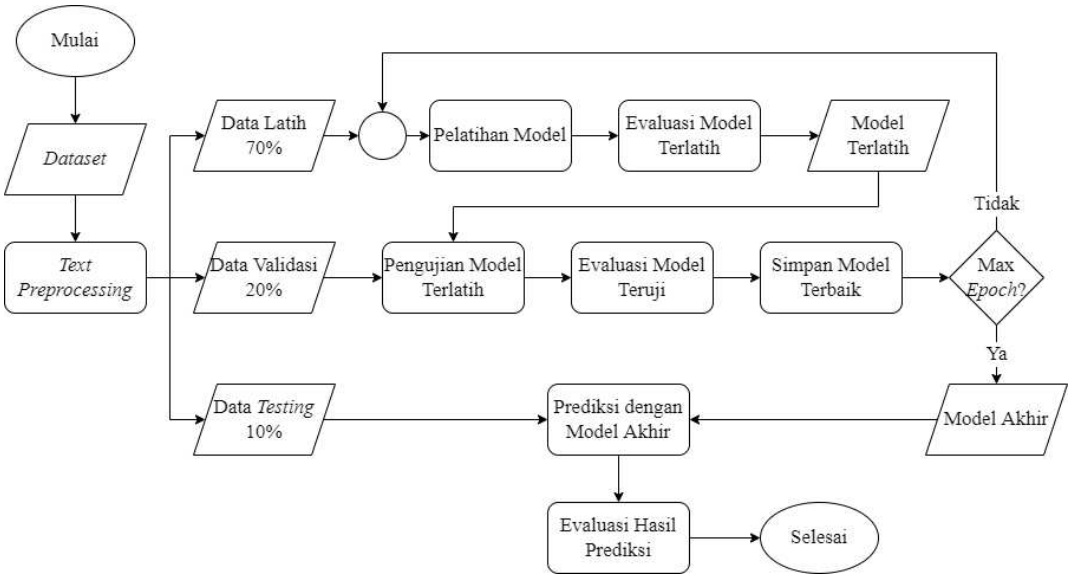
Data yang sudah diambil melalui proses *crawling* akan diberi label sesuai dengan topiknya. Data yang dikumpulkan terdapat sekitar 10.000 *tweet* yang akan terklasifikasi ke dalam 10 kelas, yaitu beasiswa, bulutangkis, demokrasi, film, investasi, kecantikan, konser, pajak, sepakbola, dan wisata. Proses pelabelan data akan dibantu oleh 3 sampai 5 teman agar mendapatkan kualitas data yang baik serta menghindari bias.

Setelah dilakukan proses pelabelan data, dilakukan pembersihan data secara manual untuk menghilangkan data *tweet* yang kurang relevan dengan kelasnya dan data *tweet* yang masih duplikat. Kemudian dilakukan *splitting* atau pemisahan data menggunakan *library* scikit-learn dengan perbandingan 70% untuk *train set*, 20% untuk *validation set*, dan 10% untuk *test set*. Sebagai persiapan pelatihan model, dilakukan proses *one-hot encoding* yang akan merepresentasikan data bertipe kategori sebagai vektor biner yang bernilai integer, 0 dan 1, di mana semua elemen akan bernilai 0 kecuali satu elemen yang bernilai 1, yaitu elemen yang memiliki nilai kategori tersebut.

Dataset akan dibuat menjadi 2 skenario. Skenario pertama merupakan *dataset* asli yang tidak dilakukan modifikasi. Sedangkan skenario kedua merupakan *dataset* yang akan dilakukan modifikasi dengan menghilangkan kata- kata yang memiliki nilai informasi yang terlalu tinggi, di mana kata-kata tersebut merupakan nama dari setiap kategori itu sendiri.

3.3 Rancangan Model Klasifikasi Teks

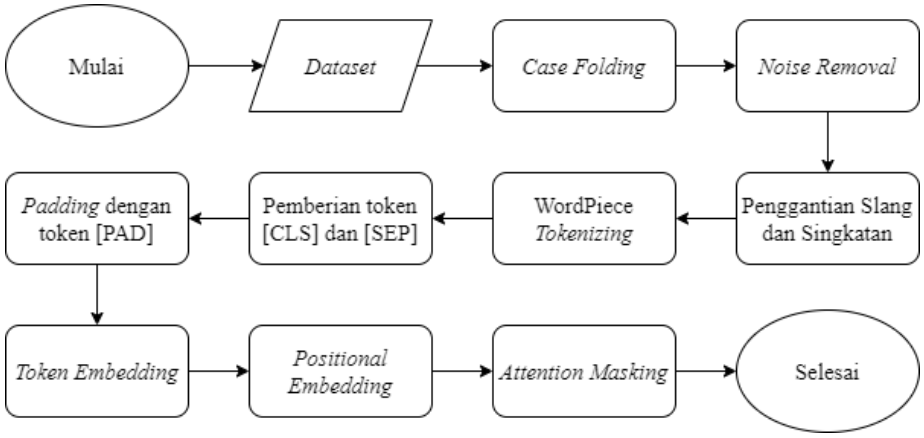
Model yang diusulkan pada penelitian ini adalah IndoBERT-LSTM. Model ini nantinya akan dibandingkan dengan dua *baseline model*, yaitu Word2Vec-LSTM dan *fine-tuned* IndoBERT. Secara keseluruhan, alur rancangan model klasifikasi teks akan terlihat seperti pada Gambar 4.1.



Gambar 4.1 Diagram Alur Pengembangan Model Klasifikasi Teks

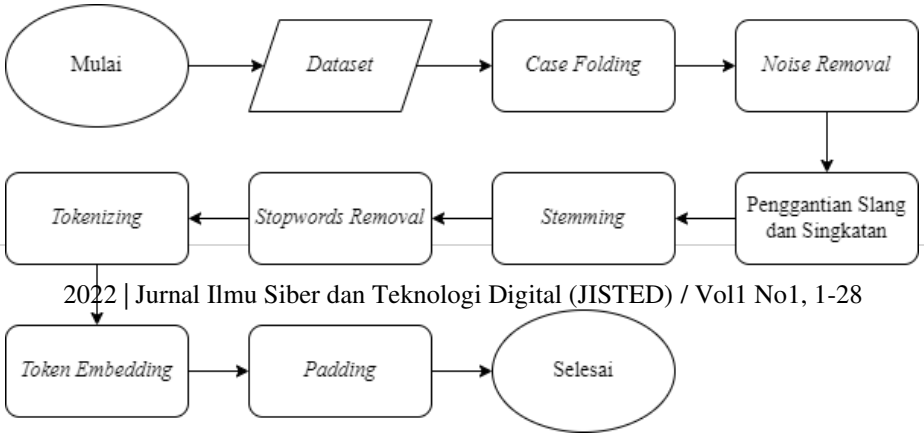
3.3.1 Rancangan Text Preprocessing

Setelah *dataset* didapatkan, selanjutnya dilakukan *text preprocessing*. *Text preprocessing* bertujuan untuk membersihkan *noise* dan fitur yang tidak diperlukan dari data teks agar menjadi lebih terstruktur dan dapat digunakan pada proses selanjutnya. Proses *text preprocessing* untuk model BERT cukup berbeda dengan model *word embedding* yang lain, karena menggunakan metode WordPiece *tokenizer* serta perlu membuat representasi *input* yang dapat diterima oleh model BERT. Sehingga, rancangan *text preprocessing* untuk model IndoBERT-LSTM dan *fine-tuned* IndoBERT akan terlihat seperti Gambar 4.2.



Gambar 4.2 *Text Preprocessing* IndoBERT-LSTM dan *fine-tuned* IndoBERT

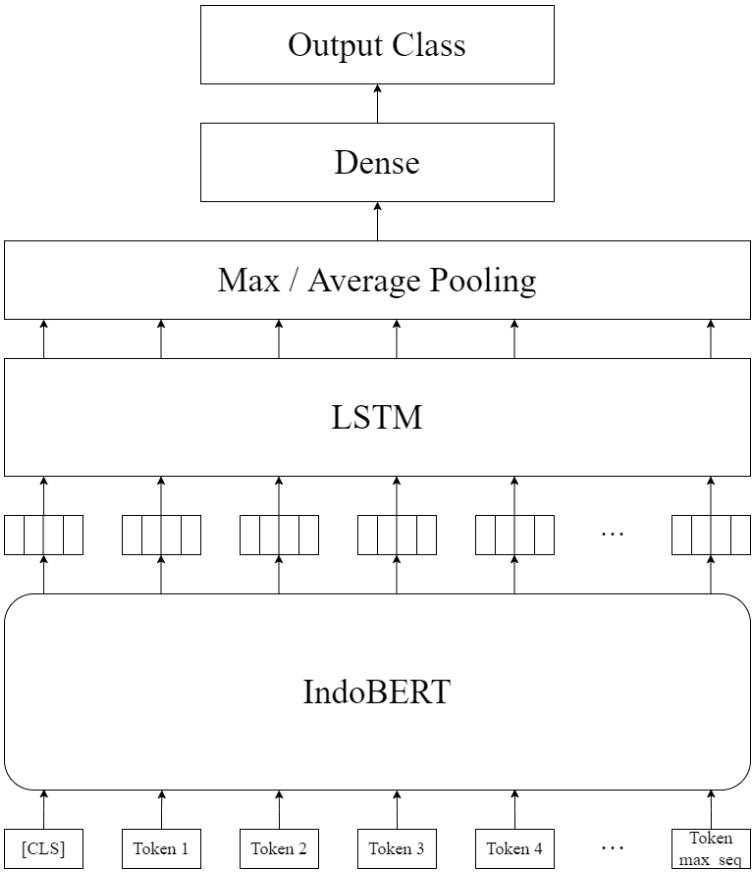
Sedangkan pada model Word2Vec-LSTM, rancangan *text preprocessing* akanterlihat seperti Gambar 4.3.



Gambar 4.3 *Text Preprocessing* Word2Vec-LSTM

3.3.2 Rancangan Model IndoBERT-LSTM

Representasi *input* yang telah dibuat akan diterima oleh model IndoBERT dan akan terus melalui tumpukan *encoder*. Setiap *encoder* mengaplikasikan *self- attention* dan memberikan *output* melalui *feed-forward network* yang kemudian dilanjutkan oleh *encoder* selanjutnya. Pada penelitian ini, digunakan model *pre- trained* IndoBERT dengan ukuran $\text{BERT}_{\text{BASE}}$, sehingga proses *embedding* akan berlanjut sebanyak 12 kali. Arsitektur model IndoBERT-LSTM diilustrasikan oleh Gambar 4.4.



Gambar 4.4 Arsitektur IndoBERT-LSTM

Penentuan *hyperparameter* merupakan langkah penting untuk mendapatkan model dengan performa terbaik. Terdapat dua kelompok *hyperparameter* yang akan digunakan. Kelompok pertama merupakan *hyperparameter* yang sudah ditetapkan dan tidak perlu dilakukan *tuning*. Sedangkan kelompok kedua merupakan *hyperparameter* yang masih harus dilakukan *tuning* untuk mendapatkan model dengan performa terbaik. Tabel 4.2 menunjukkan daftar *hyperparameter* yang akan digunakan pada model IndoBERT-LSTM.

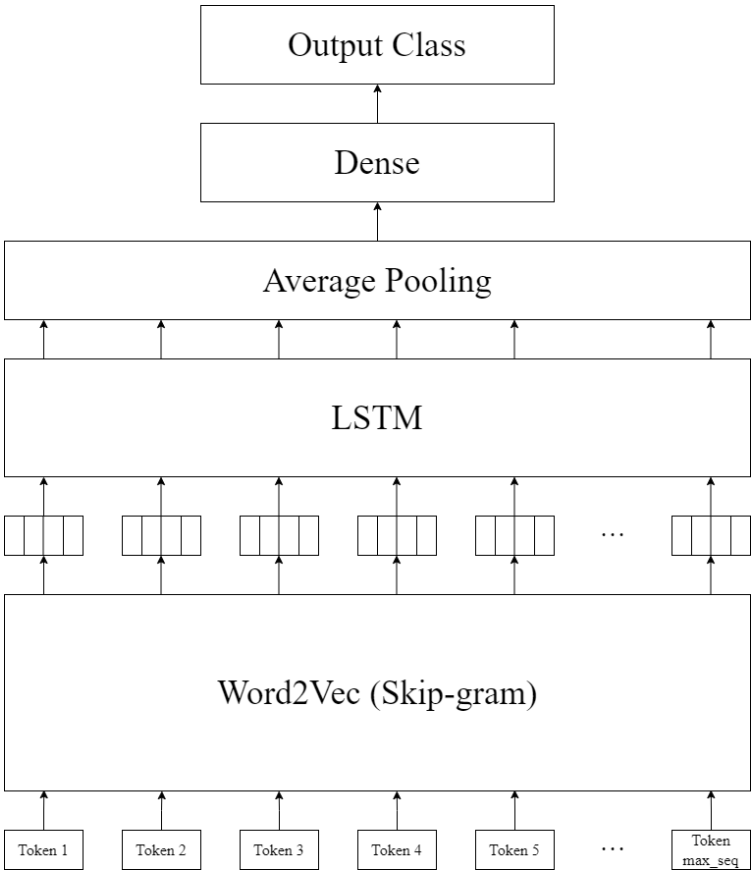
Tabel 4.2 Daftar *Hyperparameter* pada Model IndoBERT-LSTM

Kelompok	***Hyperparameter***	***Value***
Tidak perlu t*uning*	*Epoch*	20
	Max sequence	128
	Probabilitas d*ropout*	20%
	Activation function	*Softmax*
	Loss function	*Categorical cross-entropy*
Perlu t*uning*	*Batch fize*	16 atau 32
	Learning rate	2e-5 atau 5e-5

	Metode p*ooling*	*Average pooling* atau m*ax pooling*

3.3.3 Rancangan Model Baseline Word2Vec-LSTM

Model Word2Vec-LSTM akan berperan sebagai *baseline model* untuk perbandingan penggunaan model *pre-trained word embedding* yang berbeda. Arsitektur model Word2Vec-LSTM akan terlihat seperti pada Gambar 4.5 Arsitektur Word2Vec-LSTM.



Gambar 4.5 Arsitektur Word2Vec-LSTM

Hyperparameter yang digunakan sebagian besar mengacu pada penelitian yang telah dilakukan oleh Muhammad et al., (2021). Tabel 4.3 menunjukkan daftar *hyperparameter* yang akan digunakan pada model Word2Vec-LSTM.

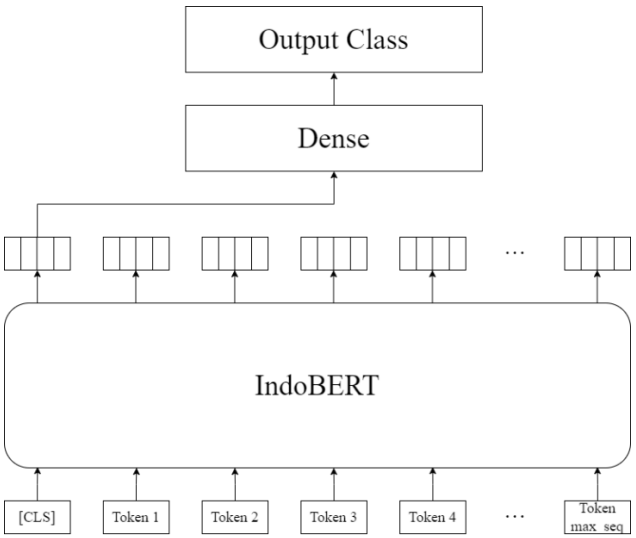
Tabel 4.3 Daftar *Hyperparameter* pada Model Word2Vec-LSTM

Hyperparameter	**Value**
Epoch	20
Batch size	64
Max sequence	128
Vector dimension	300
Learning rate	0,001
Probabilitas *dropout*	20%

Metode pooling	*Average pooling*
Activation function	*Softmax*
Loss function	*Categorical cross-entropy*

3.3.4 Rancangan Model Baseline Fine-tuned IndoBERT

Model *fine-tuned* IndoBERT akan berperan sebagai *baseline model* untuk perbandingan model yang menggunakan arsitektur LSTM dan yang tidak. Token yang digunakan pada proses *fine-tuning* untuk tugas klasifikasi teks hanyalah token [CLS] yang akan merepresentasikan keseluruhan kalimat. Vektor *output* dari token [CLS] akan dikirimkan melalui *feed forward neural network* agar dapat dilakukan klasifikasi teks berbasis *multiclass classification*. Arsitektur model *fine-tuned* IndoBERT akan terlihat seperti pada Gambar 4.6.



Gambar 4.6 Arsitektur *Fine-tuned* IndoBERT

Ukuran model IndoBERT yang digunakan adalah BERT$_{BASE}$. Tabel 4.4 menunjukkan daftar *hyperparameter* yang akan digunakan pada model *fine-tuned* IndoBERT.

Tabel 4.4 Daftar *Hyperparameter* pada Model *Fine-tuned* IndoBERT

Hyperparameter	**Value**
Epoch	20
Batch size	32
Max sequence	128
Learning rate	5e-5
Probabilitas *dropout*	20%
Activation function	*Softmax*
Loss function	*Categorical cross-entropy*

3.3.5 Perhitungan Loss

Model klasifikasi yang dikembangkan pada penelitian ini berbasis *multiclass classification* dengan jumlah kelas sebanyak 10. Sehingga, perhitungan *loss* yang cocok digunakan adalah *categorical cross-entropy*, karena dapat mengukur perbedaan antara 2 distribusi probabilitas. Vektor hasil prediksi terlebih dahulu dinormalisasi menggunakan fungsi aktivasi *softmax* agar memiliki probabilitas berjumlah 1. Selanjutnya, perhitungan *loss* dilakukan pada *output layer* dengan membandingkan vektor hasil prediksi yang telah dinormalisasi dengan vektor biner dari kelas sesungguhnya yang didapatkan melalui proses *one-hot encoding*.

3.3.6 Rancangan Evaluasi Model

Pengukuran performa model klasifikasi teks dilakukan berdasarkan metrik-metrik yang dihitung dari *confusion matrix*. Terdapat 10 kelas yang digunakan, sehingga akan terdapat 10 kolom dan 10 baris untuk *confusion matrix*. Tabel 4.5 mengilustrasikan *confusion matrix* untuk 10 kelas.

Tabel 4.5 Rancangan *Confusion Matrix* untuk 10 Kelas

Confusion Matrix		*True Class*									
		A	B	C	D	E	F	G	H	I	J
Predicted Class	A	TA	FA	FA	FA	FA	FA	FA	FA	FA	FA
	B	FB	TB	FB	FB	FB	FB	FB	FB	FB	FB
	C	FC	FC	TC	FC	FC	FC	FC	FC	FC	FC
	D	FD	FD	FD	TD	FD	FD	FD	FD	FD	FD
	E	FE	FE	FE	FE	TE	FE	FE	FE	FE	FE
	F	FF	FF	FF	FF	FF	TF	FF	FF	FF	FF
	G	FG	FG	FG	FG	FG	FG	TG	FG	FG	FG
	H	FH	FH	FH	FH	FH	FH	FH	TH	FH	FH
	I	FI	FI	FI	FI	FI	FI	FI	FI	TI	FI
	J	FJ	FJ	FJ	FJ	FJ	FJ	FJ	FJ	FJ	TJ

Confusion matrix tersebut akan digunakan sebagai dasar perhitungan metrik dalam mengevaluasi performa model. Metrik yang akan digunakan dalam membantu mengevaluasi pelatihan model IndoBERT-LSTM adalah *validation accuracy*. Sedangkan metrik yang digunakan pada proses pengujian untuk ketiga model klasifikasi teks adalah *precision*, *recall*, dan F1-*score*, karena metrik ini dapat menghindari bias pada perhitungan dengan data yang kurang seimbang.

4. Hasil dan Pembahasan

4.1 Pembuatan Dataset

Pembuatan *dataset* dilakukan melalui proses *crawling* dari aplikasi Twitter dengan bantuan *library* Tweepy. Proses *crawling* dilakukan secara berulang untuk setiap kategori, yaitu beasiswa, bulutangkis, demokrasi, film, investasi, kecantikan, konser, pajak, sepakbola, dan wisata. Cuplikan hasil *dataset* yang telah dibuat dapat dilihat pada Gambar 6.1.

text	class
b'-dips! Saran kado ulangtahun buat cewek dong? \xf0\x9f\xa5\xb2 ku bingung karna belum terlalu kenal banget dia, nanya temen bikin buket makeup yg basic aja'	kecantikan
b'ADA KABAR GEMBIRA BUAT NEOTIZEN YANG SUKA INVESTASI EMAS!\n\nSekarang kamu bisa top saldo Lakuemas di neobank! Setelah berhasil top up pakai Pembayaran VA Bank Neo, kamu bisa dapat cashback hingga Rp200 ribu & kupon diskon Lakuemas 50%! Jadi lebih hemat & bisa buatmu makin cuan! \xf0\x9f\x98\xbd https://t.co/ofa1UFfoB0'	investasi
b'5 Tempat Ngabuburit di Jaksel, Ada Kampung Wisata sampai Taman Piknik https://t.co/W9IJidyIai'	wisata
b'Wisata Alam - Gunung Papandayan di Garut, jabar https://t.co/dGxkWkaFZi'	wisata
b'SchApp! Hai aku mau tanya, ada ga beasiswa kuliah full? Kalo ada dapatnya gimana ya?\n\n\xe2\x9c\x84\xe3\x83\xbb\xe3\x83\xbb\xe3\x83\xbb\n\xf0\x9f\xa7\x90 Trivia: penumpukan asam laktat dalam tubuh terjadi ketika otot kita kekurangan oksigen, yang menyebabkan rasa pegal.'	beasiswa
b'investasi tuh di PAHALA bukan CRYPTO'	investasi
b'Rian suruh belajar servis lagi deh, eror mulu...servis hal dasar dalam permainan bulutangkis padahal'	bulutangkis
b'KENAROK SALATIGA KONSER HIBUR TNI / POLRI... GOYANG SEMUA LURR... " MAU... https://t.co/XV9KP0vqyh lewat @YouTube. @frans_indonesia @YouTube @LiveNation @AppleMusic @BN_Music @ECHO_Musikpreis @konsermusikindo'	konser
b'Sobat Pajak sekarang bisa melakukan pembayaran pajak PBB-P2 di aplikasi Mobile BCA dan OVO lho!\n\nIni merupakan salah satu cara agar wajib pajak membayar pajak dengan mudah dan cepat, dan bisa membayar pajak dimanapun dan kapanpun.\n\nJangan lupa bayar pajak tepat waktu ya Sobat! https://t.co/y6KOXvESJb'	pajak
b"Tandanya orang munafik itu banyak. Dan diantaranya adalah tdk suka pada hukum syariat Allah SWT (QS 4:61). Suka pada syari'at Allah SWT itu tidak harus jadi pemimpin. Suka pada syari'at Allah SWT itu rela hidup tanpa berpangkat disistem thogut konstitusi ideologi demokrasi pncsl."	demokrasi
b'Terlalu enteng untuk Kevin. Jadi seneng bisa cepet tidur. \xf0\x9f\x98\x8d\n\n#ThomasCup2022 #ThomasUberCup2022 #ThomasCup #ThomasUberCup #badminton #BadmintonIndonesia #bulutangkis'	bulutangkis
b'Gara" penasaran sama dr.strange 2 harus bgt nih marathon series marvel. Yaudahlah enjoyy'	film
b'Ga nonton thomas cup karena emang ga sempet kalo udh masa2 masuk kuliah tapi kalo malem menjelang tidur always liatin updatean + komen2 lucu BL indo'	bulutangkis
b'Korlap demonya mana nih..?? \nNggak tanggung jawab...!! \n\nDemonya protes pemerintah\nNuntut ini itu ke pemerintah, rusak fasilitas umm yg dibangun pemerintah\nHabis demo nggak bisa pulang, pemerintah yg nganterin\n\nSampai sini masih percaya mereka mewakili suara rakyat.?? \nMikiir..!? https://t.co/D8CCCWwc3U'	demokrasi
b'Inget pas konser h@rry, ada yg ngedrag oknum tks gara2 dia bilang jungk00k pengang tangannya t@e, org yg ngedrag ini g percaya gara2 fotonya blur dan.... beberapa menit kemudian bang h0bi ngepost video dan disitu j3ka pegang tangan t4e beneran\xf0\x9f\x98\x82 mata tks itu tajem'	konser
b'Bilqis Prasista Mengejutkan Dunia Bulutangkis https://t.co/OglbxADuwu'	bulutangkis
b'Foto-Foto Masjid Perahu, Destinasi Wisata Religi Tersembunyi di Jakarta\n\n#CelebritiesDotID #InfoCelebTerkini \nhttps://t.co/MJcQfwR0ar'	wisata
b'Langganan netflix buat nonton film yg lgi rame \xe2\x9d\x8c\nLangganan netflix buat nonton barbie\xe2\x9c\x85'	film
b'IHSG turun, porto merah tp masih oke karena udh take profit sebelum lebaran. Tapi jangan tanya crypto. Dahlah bye bye dulu aja sampe nunggu akhir tahun \xf0\x9f\x98\x8c'	investasi
b'Sportiipstore juga menerima pembuatan jersey buat tim futsal dan sepakbola kalian ya #sportiipstore #murah #berkualitas'	sepakbola

Gambar 6.1 Sampel data dalam *Dataset*

Proses pelatihan model dilakukan dalam 2 skenario *dataset*. Skenario pertama merupakan *dataset* asli yang tidak termodifikasi. Sedangkan skenario kedua merupakan *dataset* yang termodifikasi dengan menghilangkan kata-kata yang memiliki nilai informasi yang terlalu tinggi dalam melakukan klasifikasiteks, di mana kata-kata tersebut merupakan nama dari setiap kategori itu sendiri.

4.2 Hasil Model IndoBERT-LSTM

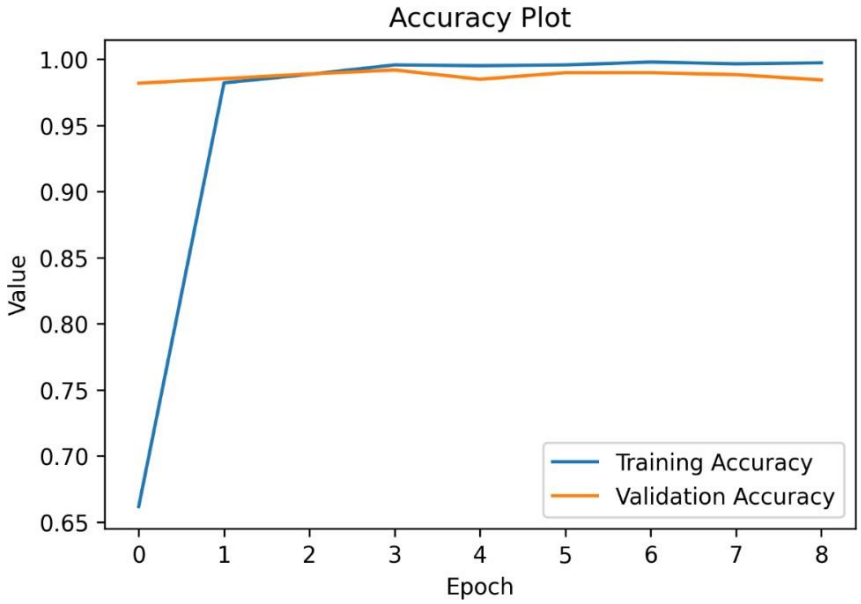
4.2.1 Hasil pada Dataset tidak Termodifikasi

Proses pelatihan model IndoBERT-LSTM dilakukan sesuai dengan skenario pelatihan yang telah dirancang sebelumnya. Terdapat 8 skenario pelatihan yang merupakan kombinasi dari 3 *hyperparameter* yang di-*tuning*, yaitu *batch size* sebesar 16 dan 32, *learning rate* sebesar 2e-5 dan 5e-5, serta *pooling layer* menggunakan *average pooling* dan *max pooling*. Hal ini dilakukan untuk menguji pengaruh dari setiap *hyperparameter* tersebut dalam menghasilkan model IndoBERT-LSTM terbaik.

Tabel 6.1 Hasil Evaluasi pada Pelatihan Model IndoBERT-LSTM

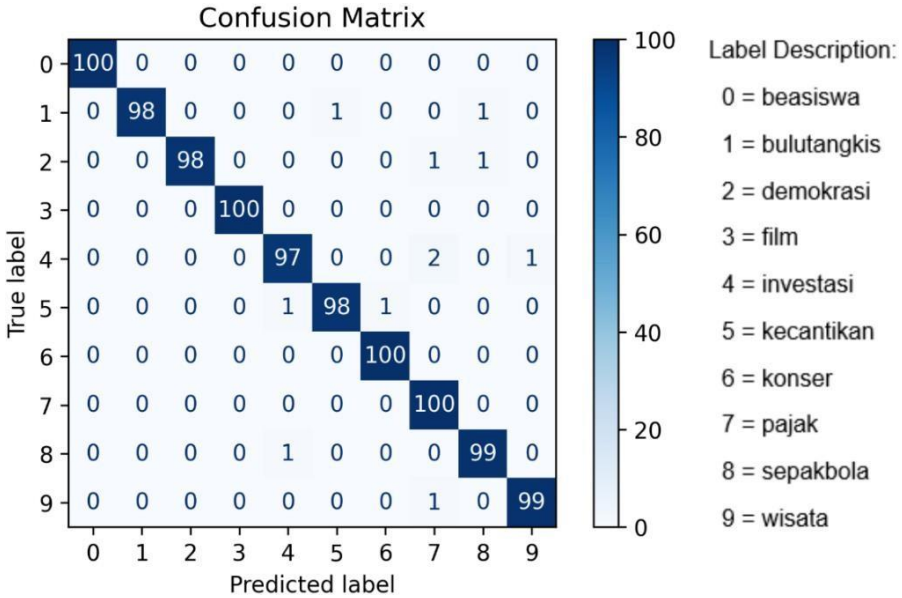
No.	***Batch Size***	***Learning Rate***	***Pooling***	***Validation Accuracy***
1.	16	2e-5	*Average*	99,20%
2.	16	2e-5	*Max*	99,00%
3.	16	5e-5	*Average*	99,10%
4.	16	5e-5	*Max*	98,95%
5.	32	2e-5	*Average*	99,15%
6.	32	2e-5	*Max*	99,00%
7.	32	5e-5	*Average*	99,10%
8.	32	5e-5	*Max*	99,00%

Tabel 6.1 menunjukkan hasil *validation accuracy* dari setiap skenario pelatihan model IndoBERT-LSTM. Terlihat model pada skenario pertama dengan *batch size* sebesar 16, *learning rate* sebesar 2e-5, dan metode *pooling* menggunakan *average pooling* berhasil mendapatkan *validation accuracy* tertinggi yang mencapai 99,20%. Apabila melihat hasil pelatihan model pada skenario ke-1 dan ke-5, model dengan jumlah *batch size* 16 memiliki *validation accuracy* yang lebih tinggi daripada jumlah *batch size* 32 meskipun perbedaannya tidak terlalu signifikan. Hal ini dikarenakan *batch size* yang lebih kecil lebih menimbulkan *noise* dan menawarkan efek regularisasi, sehingga menghasilkan *generalization error* yang lebih rendah. Perbedaan *learning rate* juga mempengaruhi hasil pelatihan model, di mana *learning rate* 2e-5 memiliki *validation accuracy* lebih tinggi daripada *learning rate* 5e-5 seperti yang terlihat pada perbandingan skenario ke-1 dan ke-3. *Learning rate* yang lebih besar menyebabkan perubahan *gradient descent* yang lebih besar pula, sehingga menyebabkan kurang tercapainya solusi optimal yang diinginkan. Sedangkan pada skenario penggunaan metode *pooling*, metode *average pooling* menghasilkan *validation accuracy* yang lebih baik daripada metode *max pooling*. Hal ini disebabkan vektor *output* yang dihasilkan oleh metode *average pooling* lebih merepresentasikan keseluruhan rangkaian vektor yang dihasilkan oleh *layer* sebelumnya dengan cara mengambil rata-ratanya, di mana pada metode *max pooling* hanya diambil vektor tertinggi yang belum tentu dapat merepresentasikan keseluruhan rangkaian vektor.



Gambar 6.2 Plot Akurasi Pelatihan Model IndoBERT-LSTM (1)

Grafik perkembangan akurasi hasil pelatihan model IndoBERT-LSTM dengan skenario terbaik pada *dataset* yang tidak termodifikasi dapat dilihat pada Gambar 6.2. Terlihat pada *epoch* pertama model ini telah mencapai *validation accuracy* di atas 95% dan puncaknya berada pada *epoch* ke-4 dengan *validation accuracy* sebesar 99,20%. Pada *epoch* selanjutnya, model tidak lagi mengalami peningkatan akurasi, sehingga pelatihan model berhenti pada *epoch* ke-9. Total waktu yang dibutuhkan untuk melatih model IndoBERT-LSTM pada *dataset* yang tidak termodifikasi sekitar 30 menit. Model juga terlihat tidak mengalami *overfit*.



Gambar 6.3 *Confusion Matrix* Pengujian Model IndoBERT-LSTM (1)

Model IndoBERT-LSTM skenario terbaik dengan *validation accuracy* sebesar 99,20% disimpan agar dapat diuji dengan melakukan prediksi pada data *test* yang belum pernah ditemui sebelumnya. Dari hasil prediksi tersebut, dibuat *confusion matrix* seperti yang terlihat pada Gambar 6.3. Dapat dilihat bahwa model dapat mengklasifikasikan data *test* ke setiap kelas dengan sangat baik. Terdapat 4 kelas

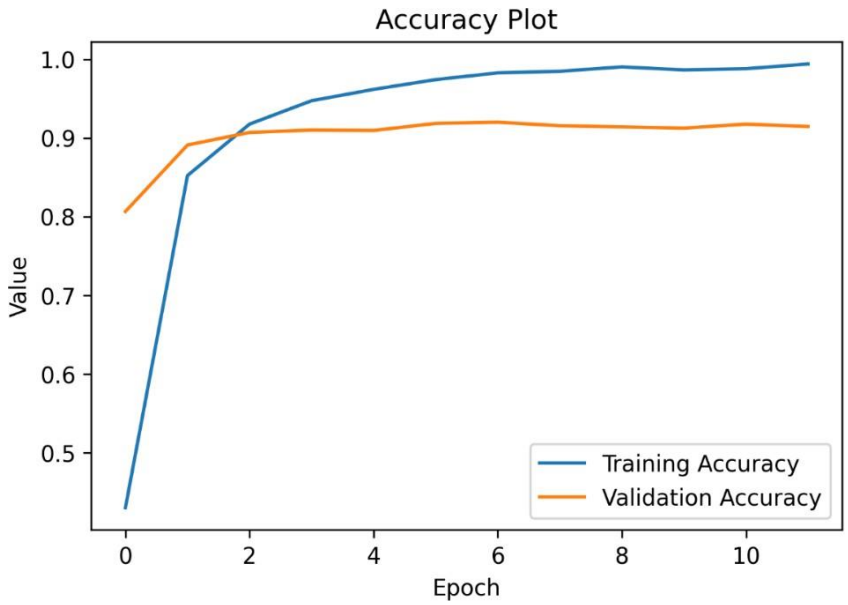
yang diprediksi dengan benar 100% dan kelas lain hanya mengalami sedikit kesalahan. Kelas dengan kesalahan terbanyak adalah kelasinvestasi yang diprediksi sebagai kelas pajak sebanyak 2 kali.

Kesalahan-kesalahan prediksi tersebut bisa terjadi karena adanya kata-kata yang secara kontekstual memiliki arti yang sama namun digunakan pada beberapa kelas, atau karena tidak adanya kata-kata dengan nilai informasi yang tinggi pada *tweet* tersebut yang dapat dijadikan tumpuan oleh model dalam melakukan klasifikasi.

Dari *confusion matrix* tersebut dapat dihitung nilai *macro-average* dari *precision* sebesar 98,92%, *recall* sebesar 98,90%, dan F1-*score* sebesar 98,90%. Terlihat model IndoBERT-LSTM menunjukkan performa yang sangat baik dalam mengklasifikasikan *dataset* yang tidak termodifikasi.

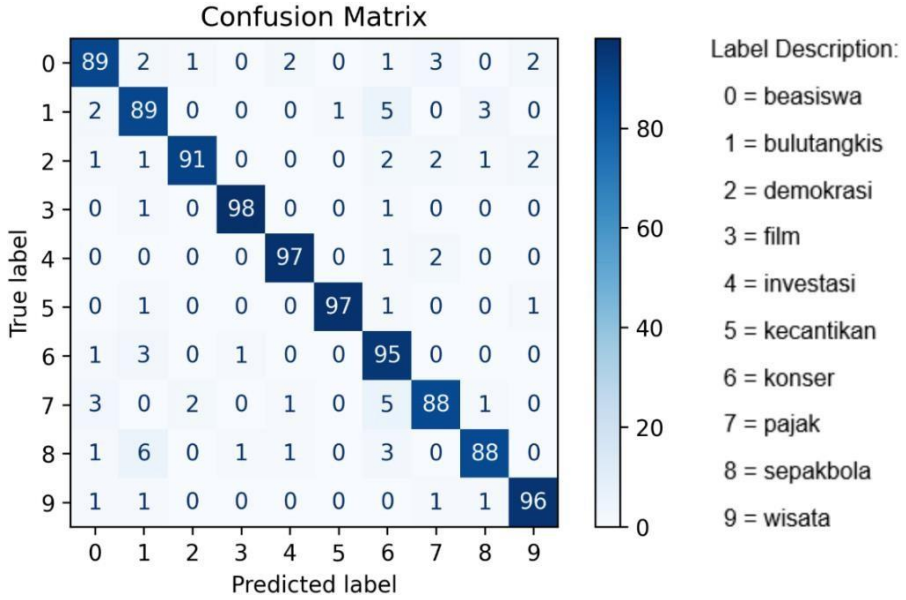
4.2.2 Hasil pada Dataset Termodifikasi

Pada percobaan pelatihan menggunakan *dataset* tidak termodifikasi, Model IndoBERT-LSTM dapat dengan sangat mudah melakukan klasifikasi. Hal ini dikarenakan hampir di setiap data *tweet* terdapat kata-kata yang memiliki nilai informasi yang tinggi, yaitu kata-kata yang merupakan nama dari setiap kategori itu sendiri yang memudahkan model dalam melakukan klasifikasi. Oleh karena itu, dilakukan juga percobaan untuk memodifikasi *dataset* dengan cara menghilangkan kata-kata penting tersebut yang bertujuan untuk melihat bagaimana performa model pada *dataset* yang lebih sulit dan saling berdekatan.



Gambar 6.4 Plot Akurasi Pelatihan Model IndoBERT-LSTM (2)

Pelatihan model IndoBERT-LSTM pada *dataset* yang telah termodifikasi dilakukan dengan menggunakan skenario kombinasi *hyperparameter* terbaik yang telah didapatkan pada pelatihan model dengan *dataset* yang belum termodifikasi. Grafik perkembangan akurasi hasil pelatihan model pada *dataset* yang telah termodifikasi dapat dilihat pada Gambar 6.4. Terlihat adanya perkembangan di mana pada *epoch* pertama model ini mendapatkan *validation accuracy* di atas 80% dan puncaknya berada pada *epoch* ke-7 dengan *validation accuracy* sebesar 92%. Pada *epoch* selanjutnya, model tidak lagi mengalami peningkatan akurasi, sehingga pelatihan model berhenti pada *epoch* ke-12. Total waktu yang dibutuhkan untuk melatih model IndoBERT-LSTM pada *dataset* yang telah termodifikasi sekitar 45 menit. Model sedikit mengalami *overfit* namun masih terbilang wajar karena perbedaan antara *validation accuracy* dengan *training accuracy* tidak terlalu signifikan.



Gambar 6.5 *Confusion Matrix* Pengujian Model IndoBERT-LSTM (2)

Model IndoBERT-LSTM yang telah dilatih pada *dataset* yang telah termodifikasi dengan *validation accuracy* 92% disimpan agar dapat diuji dengan melakukan prediksi pada data *test* yang juga sudah termodifikasi dan belum pernah ditemui sebelumnya. Dari hasil prediksi tersebut, dibuat *confusion matrix* seperti yang terlihat pada Gambar 6.5. Terlihat model bisa mengklasifikasikan data *test* ke setiap kelas dengan cukup baik walaupun masih mengalami beberapa kesalahan. Kelas dengan kesalahan terbanyak adalah kelas sepakbola yang diprediksi sebagai kelas bulutangkis sebanyak 6 kali. Kelas lain dengan kesalahan cukup banyak adalah kelas bulutangkis yang diprediksi sebagai kelas konser dan kelas pajak yang diprediksi sebagai kelas konser sebanyak 5 kali.

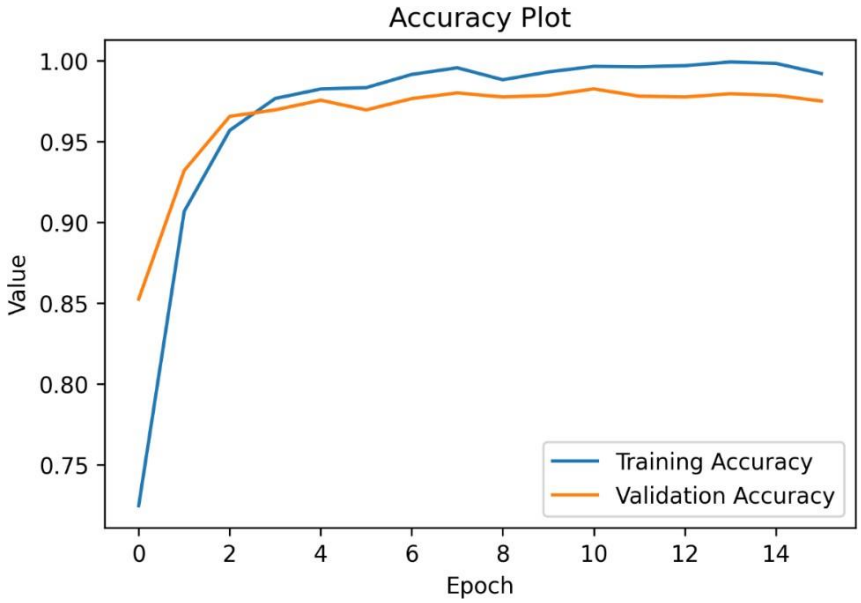
Dapat dilihat jumlah kesalahan prediksi pada *dataset* yang telah termodifikasi yang didapatkan menjadi lebih banyak jika dibandingkan dengan kesalahan pada *dataset* yang tidak termodifikasi. Hal ini wajar terjadi karena dengan dihapusnya kata-kata dengan nilai informasi yang tinggi, membuat model menjadi lebih sulit dalam memahami konteks dan melakukan klasifikasi pada *tweet* tersebut.

Dari *confusion matrix* tersebut dapat dihitung nilai *macro-average* dari *precision* sebesar 92,99%, *recall* sebesar 92,80%, dan F1-*score* sebesar 92,83%. Meskipun telah dilakukan modifikasi pada *dataset*, model IndoBERT-LSTM tetap berhasil mendapatkan nilai di atas 90% untuk ketiga metrik tersebut.

4.3 Hasil Model Baseline Word2Vec-LSTM

4.3.1 Hasil pada Dataset tidak Termodifikasi

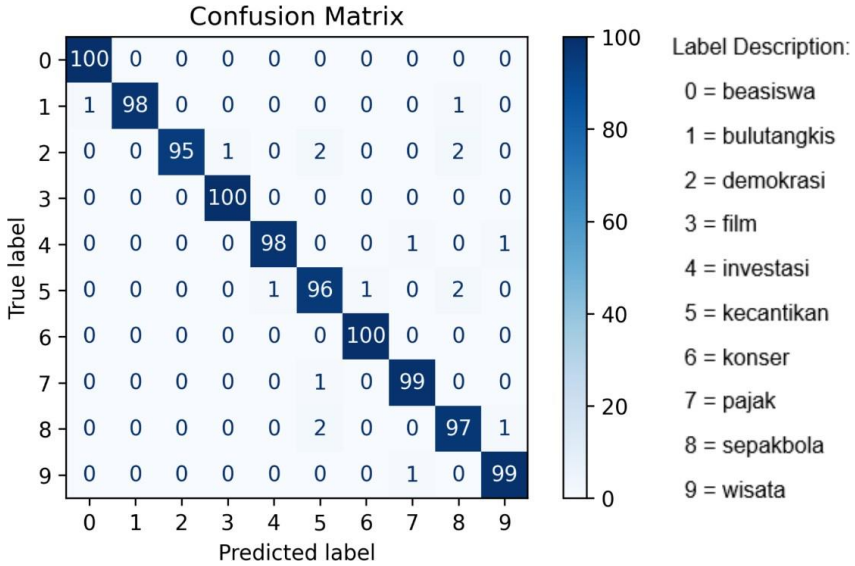
Pelatihan model *baseline* Word2Vec-LSTM dilakukan dengan menggunakan *hyperparameter* yang sebagian besar mengacu pada model milik Muhammad et al., (2021). Grafik perkembangan akurasi hasil pelatihan model pada *dataset* yang tidak termodifikasi dapat dilihat pada Gambar 6.6. Pada *epoch* pertama model ini mendapatkan *validation accuracy* di atas 85% dan terus mengalami peningkatan di mana puncaknya berada pada *epoch* ke-11 dengan *validation accuracy* sebesar 98,25%. Pada *epoch* selanjutnya, model tidak lagi mengalami peningkatan akurasi dan berhenti pada *epoch* ke-16. Total waktu yang dibutuhkan untuk melatih model Word2Vec-LSTM pada *dataset* yang tidak termodifikasi sekitar 3 menit. Model juga terlihat tidak mengalami *overfit*.



Gambar 6.6 Plot Akurasi Pelatihan Model Word2Vec-LSTM (1)

Model Word2Vec-LSTM dengan *validation accuracy* sebesar 99,20% disimpan agar dapat diuji dengan melakukan prediksi pada data *test* yang belum pernah ditemui sebelumnya. Dari hasil prediksi tersebut, dibuat *confusion matrix* seperti yang terlihat pada Gambar 6.7. Dapat dilihat bahwa model dapat mengklasifikasikan data *test* ke setiap kelas dengan sangat baik. Terdapat 3 kelas yang diprediksi dengan benar 100% dan kelas lain hanya mengalami sedikit kesalahan. Kelas dengan kesalahan terbanyak adalah kelas demokrasi yang diprediksi sebagai kelas kecantikan dan kelas sepakbola serta kelas kecantikan yang diprediksi sebagai kelas sepakbola dan sebaliknya dengan jumlah kesalahan sebanyak 2 kali.

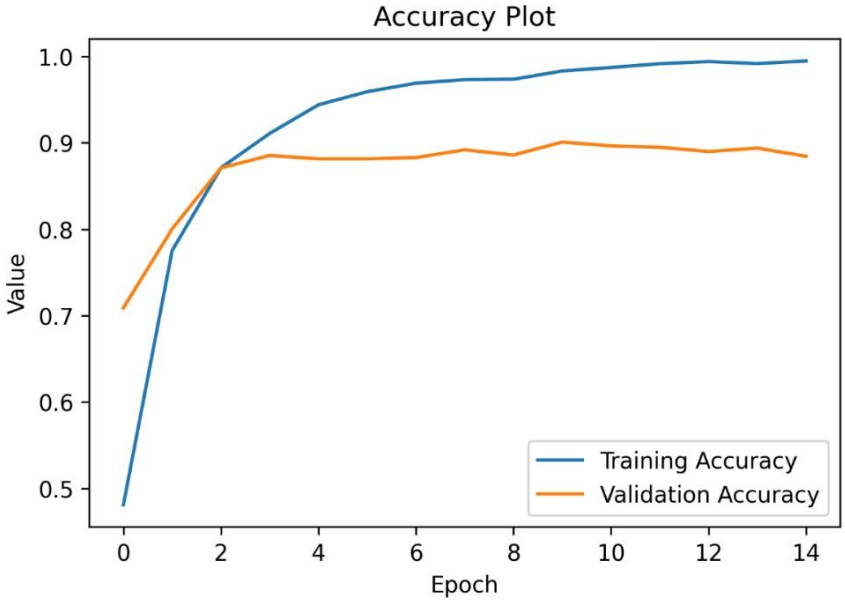
Dari *confusion matrix* tersebut dapat dihitung nilai *macro-average* dari *precision* sebesar 98,22%, *recall* sebesar 98,20%, dan F1-*score* sebesar 98,20%. Terlihat model Word2Vec-LSTM menunjukkan performa yang sangat baik dalam mengklasifikasikan *dataset* yang tidak termodifikasi.



Gambar 6.7 *Confusion Matrix* Pengujian Model Word2Vec-LSTM (1)

4.3.2 Hasil pada Dataset Termodifikasi

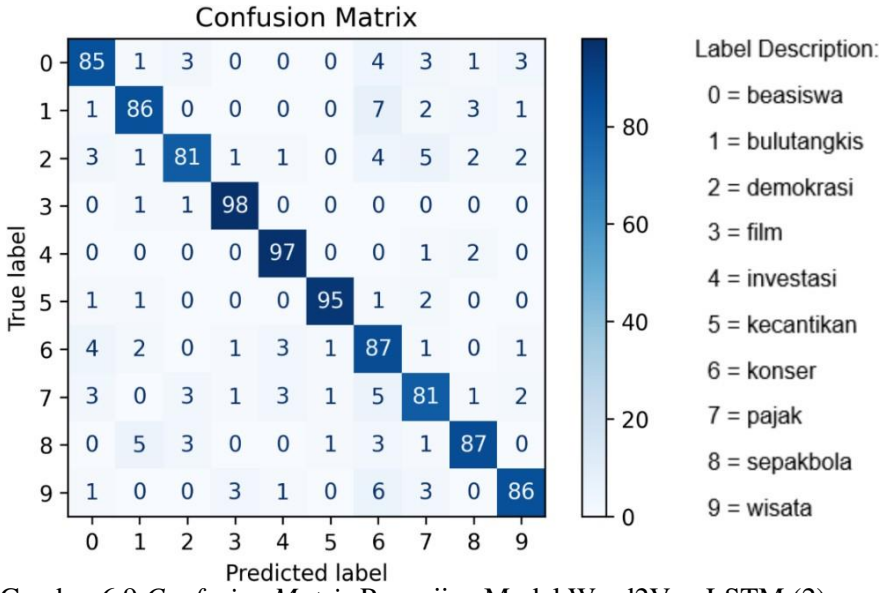
Pelatihan model *baseline* Word2Vec-LSTM juga dilakukan pada *dataset* yang telah termodifikasi. Grafik perkembangan akurasi hasil pelatihan model Word2Vec-LSTM pada *dataset* yang telah termodifikasi dapat dilihat pada Gambar 6.8. Pada *epoch* pertama model ini mendapatkan *validation accuracy* di atas 70% dan terus mengalami peningkatan di mana puncaknya berada pada *epoch* ke-10 dengan *validation accuracy* sebesar 90,10%. Pada *epoch* selanjutnya, model tidak lagi mengalami peningkatan akurasi dan berhenti pada *epoch* ke-15. Total waktu yang dibutuhkan untuk melatih model Word2Vec-LSTM pada *dataset* yang telah termodifikasi sekitar 3 menit. Akan tetapi, walaupun model berhasil mendapatkan *validation accuracy* yang cukup tinggi, model ini terlihat mengalami *overfit* yang cukup jelas jika dibandingkan dengan grafik perkembangan *training accuracy*.



Gambar 6.8 Plot Akurasi Pelatihan Model Word2Vec-LSTM (2)

Model Word2Vec-LSTM yang telah dilatih pada *dataset* yang telah termodifikasi dengan *validation accuracy* 90,10% disimpan agar dapat diuji dengan melakukan prediksi pada data *test* yang juga sudah termodifikasi dan belum pernah ditemui sebelumnya. Dari hasil prediksi tersebut, dibuat *confusion matrix* seperti yang terlihat pada Gambar 6.9. Kelas dengan kesalahan terbanyak adalah kelas bulutangkis yang diprediksi sebagai kelas konser sebanyak 7 kali. Kelas lain dengan kesalahan cukup banyak adalah kelas wisata yang diprediksi sebagai kelas konser sebanyak 6 kali.

Dari *confusion matrix* tersebut dapat dihitung nilai *macro-average* dari *precision* sebesar 88,53%, *recall* sebesar 88,30%, dan F1-*score* sebesar 88,32%. Terlihat model Word2Vec-LSTM belum cukup baik dalam mengklasifikasikan *dataset* yang telah termodifikasi, di mana nilai dari ketiga metrik evaluasipengujian tidak mencapai 90%.

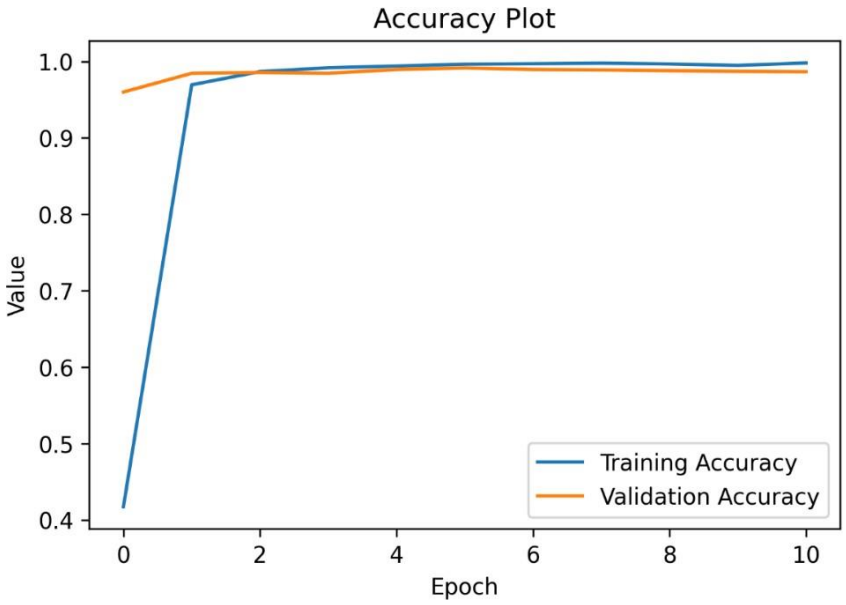


Gambar 6.9 *Confusion Matrix* Pengujian Model Word2Vec-LSTM (2)

4.4 Hasil Model Baseline Fine-tuned IndoBERT

4.4.1 Hasil pada Dataset tidak Termodifikasi

Pelatihan model *baseline fine-tuned* IndoBERT dilakukan dengan menggunakan *hyperparameter* yang mengacu pada model milik Koto et al., (2020) dalam melakukan *fine-tuning* untuk tugas *sentiment analysis*. Grafik perkembangan akurasi hasil pelatihan model *fine-tuned* IndoBERT pada *dataset* yang tidak termodifikasi dapat dilihat pada Gambar 6.10. Terlihat pada *epoch* pertama model ini telah mencapai *validation accuracy* sekitar 95% dan puncaknya berada pada *epoch* ke-6 dengan *validation accuracy* sebesar 99,15%. Pada *epoch* selanjutnya, model tidak lagi mengalami peningkatan akurasi, sehingga pelatihan model berhenti pada *epoch* ke-11. Total waktu yang dibutuhkan untuk melatih model *fine-tuned* IndoBERT pada *dataset* yang tidak termodifikasi sekitar 40 menit. Model juga terlihat tidak mengalami *overfit*.

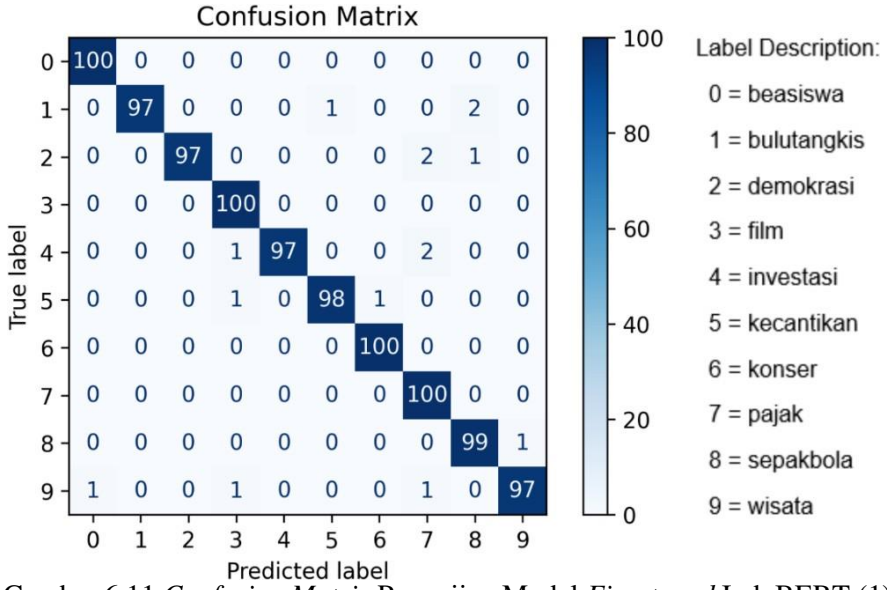


Gambar 6.10 Plot Akurasi Pelatihan Model *Fine-tuned* IndoBERT (1)

Model *fine-tuned* IndoBERT dengan *validation accuracy* sebesar 99,15% disimpan agar dapat diuji dengan melakukan prediksi pada data *test* yang belum pernah ditemui sebelumnya. Dari hasil prediksi

tersebut, dibuat *confusion matrix* seperti yang terlihat pada Gambar 6.11. Dapat dilihat bahwa model dapat mengklasifikasikan data *test* ke setiap kelas dengan sangat baik. Terdapat 4 kelas yang diprediksi dengan benar 100% dan kelas lain hanya mengalami sedikit kesalahan. Kelas dengan kesalahan terbanyak adalah kelas bulutangkis yang diprediksi sebagai kelas sepakbola, kelas demokrasi yang diprediksi sebagai kelas pajak, dan kelas investasi yang diprediksi sebagai kelas pajak dengan jumlah kesalahan sebanyak 2 kali.

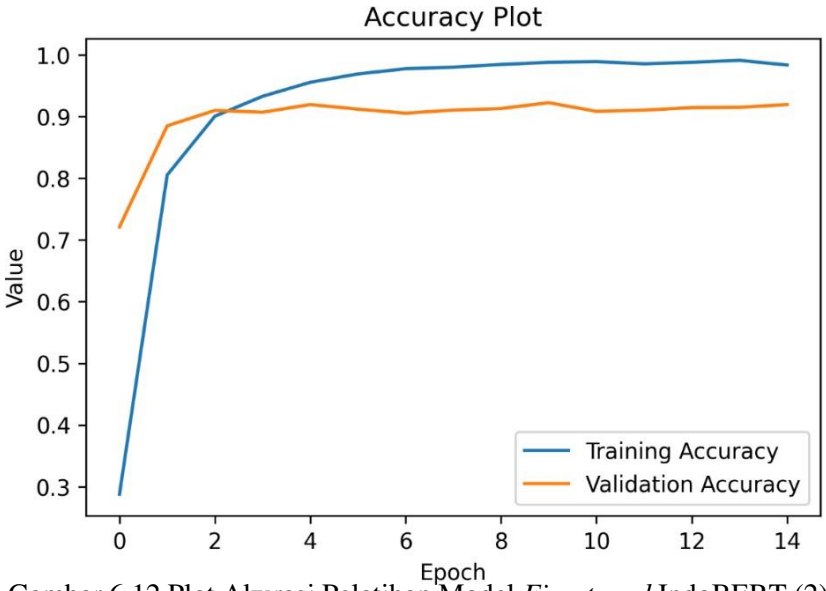
Dari *confusion matrix* tersebut dapat dihitung nilai *macro-average* dari *precision* sebesar 98,54%, *recall* sebesar 98,50%, dan F1-*score* sebesar 98,50%. Terlihat model *fine-tuned* IndoBERT menunjukkan performa yang sangat baik dalam mengklasifikasikan *dataset* yang tidak termodifikasi.



Gambar 6.11 *Confusion Matrix* Pengujian Model *Fine-tuned* IndoBERT (1)

4.4.2 Hasil pada Dataset Termodifikasi

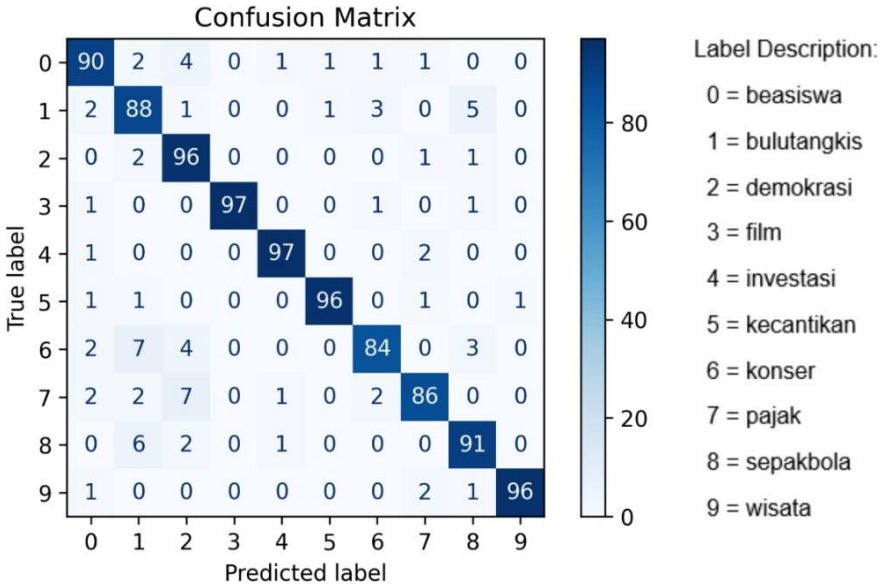
Pelatihan model *baseline fine-tuned* IndoBERT juga dilakukan pada *dataset* yang telah termodifikasi. Grafik perkembangan akurasi hasil pelatihan model *fine-tuned* IndoBERT pada *dataset* yang telah termodifikasi dapat dilihat pada Gambar 6.12. Pada *epoch* pertama model ini mendapatkan *validation accuracy* di atas 70% dan terus mengalami peningkatan di mana puncaknya berada pada *epoch* ke-10 dengan *validation accuracy* sebesar 92,25%. Pada *epoch* selanjutnya, model tidak lagi mengalami peningkatan akurasi dan berhenti pada *epoch* ke-15. Total waktu yang dibutuhkan untuk melatih model *fine-tuned* IndoBERT pada *dataset* yang telah termodifikasi sekitar 55 menit. Model sedikit mengalami *overfit* namun masih terbilang wajar karena perbedaan antara *validation accuracy* dengan *training accuracy* tidak terlalu signifikan.



Gambar 6.12 Plot Akurasi Pelatihan Model *Fine-tuned* IndoBERT (2)

Model *fine-tuned* IndoBERT yang telah dilatih pada *dataset* yang telah termodifikasi dengan *validation accuracy* 92,25% disimpan agar dapat diuji dengan melakukan prediksi pada data *test* yang juga sudah termodifikasi dan belum pernah ditemui sebelumnya. Dari hasil prediksi tersebut, dibuat *confusion matrix* seperti yang terlihat pada Gambar 6.13. Terlihat model bisa mengklasifikasikan data *test* ke setiap kelas dengan cukup baik walaupun masih mengalami beberapa kesalahan. Kelas dengan kesalahan terbanyak adalah kelas konser yang diprediksi sebagai kelas bulutangkis dan kelas pajak yang diprediksi sebagai kelas demokrasi dengan jumlah kesalahan sebanyak 7 kali. Kelas lain dengan kesalahan cukup banyak adalah kelas sepakbola yang diprediksi sebagai kelas bulutangkis sebanyak 6 kali.

Dari *confusion matrix* tersebut dapat dihitung nilai *macro-average* dari *precision* sebesar 92,36%, *recall* sebesar 92,10%, dan F1-*score* sebesar 92,14%. Meskipun telah dilakukan modifikasi pada *dataset*, model *fine-tuned* IndoBERT tetap berhasil mendapatkan nilai di atas 90% untuk ketiga metrik tersebut.



Gambar 6.13 *Confusion Matrix* Pengujian Model *Fine-tuned* IndoBERT (2)

4.5 Perbandingan Model

Hasil perhitungan nilai *precision*, *recall*, dan F1-*score* dari masing-masing model dapat dilihat pada

Tabel 6.2. Dapat dilihat bahwa model IndoBERT-LSTM dengan skenario kombinasi *hyperparameter* terbaik berhasil mendapatkan nilai tertinggi di setiap metrik, baik pada *dataset* yang tidak termodifikasi maupun pada*dataset* yang telah termodifikasi.

Tabel 6.2 Hasil Evaluasi pada Pengujian Ketiga Model Klasifikasi Teks

Dataset	**Model**	***Precision***	***Recall***	**F1-*score***	***Epoch***	**Waktu Pelatihan**
Tidak Termodifikasi	IndoBERT-LSTM	98,92%	98,90%	98,90%	9	± 30 menit
	Word2Vec-LSTM	98,22%	98,20%	98,20%	16	± 3 menit
	Fine-tuned IndoBERT	98,54%	98,50%	98,50%	11	± 40 menit
Termodifikasi	IndoBERT-LSTM	92,99%	92,80%	92,83%	12	± 45 menit
	Word2Vec-LSTM	88,53%	88,30%	88,32%	15	± 3 menit
	Fine-tuned IndoBERT	92,36%	92,10%	92,14%	15	± 55 menit

4.5.1 Perbandingan Penggunaan Word Embedding

Model IndoBERT-LSTM mengalami peningkatan yang cukup signifikan dibandingkan dengan model *baseline* Word2Vec-LSTM. Dilihat dari hasil perhitungan F1-*score*, model IndoBERT-LSTM berhasil mengalami peningkatan sebanyak 0,70% pada *dataset* yang tidak termodifikasi dan 4,51% pada *dataset* yang telah termodifikasi. Apabila dilakukan perbandingan hasil *confusion matrix* pada *dataset* yang telah termodifikasi, hampir di setiap kelas jumlah kesalahan prediksi yang dihasilkan oleh model IndoBERT-LSTM lebih sedikit jika dibandingkan dengan model Word2Vec-LSTM. Persebaran kesalahan prediksi pada model IndoBERT-LSTM juga terlihat lebih terfokus pada kelas yang secara kontekstual cukup berdekatan dengan kelas yang sesungguhnya, di mana pada model Word2Vec-LSTM kesalahan prediksinya lebih tersebar ke beberapa kelas.

Model IndoBERT-LSTM dapat bekerja lebih baik dibandingkan model Word2Vec-LSTM dikarenakan IndoBERT merupakan *context-dependent embedding* yang dapat menghasilkan lebih dari satu representasi vektor untuk kata yang sama berdasarkan konteks di mana kata tersebut digunakan dalam suatu kalimat. Di sisi lain, Word2Vec merupakan *context-independent embedding* yang hanya dapat menghasilkan satu representasi vektor untuk kata yang sama meskipun memiliki konteks yang berbeda dalam suatu kalimat. IndoBERT juga dapat menangani permasalahan *out-of-vocabulary* dengan baik karena menggunakan WordPiece *tokenizer* yang dapat memecah suatu kata menjadi sub-kata apabila kata tersebut tidak ada di *vocabulary*. Sedangkan Word2Vec hanya menggunakan *tokenizer* biasa, sehingga apabila terdapat kata-kata yang tidak tercantum dalam *vocabulary*, kata-kata tersebut akan diberi nilai vektor nol.

Jika diperhatikan dari kompleksitas waktunya, total *epoch* yang dibutuhkan oleh model IndoBERT-LSTM memang lebih sedikit dari model Word2Vec-LSTM. Akan tetapi, waktu yang dibutuhkan untuk menyelesaikan 1 *epoch* pada model IndoBERT-LSTM cukup lama, yaitu sekitar 3 - 4 menit, dibandingkan dengan model Word2Vec-LSTM yang hanya membutuhkan beberapa detik saja. Sehingga, total waktu pelatihan dari model Word2Vec-LSTM masih jauh lebih cepat dibandingkan model IndoBERT-LSTM. Hal ini dikarenakan IndoBERT perlu mempelajari konteks dari setiap kata dalam suatu kalimat, sedangkan Word2Vec tidak terikat pada konteks dan hanya melakukan *pairing* kata pada *vocabulary*.

4.5.2 Perbandingan Penggunaan Metode Pengklasifikasi

Model IndoBERT-LSTM juga mengalami peningkatan jika dibandingkan dengan model *baseline fine-tuned* IndoBERT meskipun peningkatannya tidak terlalu signifikan. Dilihat dari hasil perhitungan F1-*score*, model IndoBERT- LSTM berhasil mengalami peningkatan sebanyak 0,40% pada *dataset* yang tidak termodifikasi dan 0,69% pada *dataset* yang telah termodifikasi. Apabila dilakukan perbandingan hasil *confusion matrix* pada *dataset* yang telah termodifikasi, jumlah kesalahan prediksi yang dihasilkan oleh model IndoBERT-LSTM lebih sedikit jika dibandingkan dengan model *fine-tuned* IndoBERT meskipun tidak terjadi di setiap kelas. Dilihat dari persebaran kesalahan prediksinya, kedua model sama-sama terfokus pada kelas yang secara kontekstual cukup berdekatan dengan kelas yang sesungguhnya.

Model IndoBERT-LSTM dapat bekerja lebih baik dibandingkan model *fine-tuned* IndoBERT dikarenakan LSTM memiliki *feedback connection* yang memungkinkan untuk mempertahankan informasi dalam memori dari waktu ke waktu dengan jangka waktu yang lama. Hal ini yang membuat LSTM dapat memproses seluruh rangkaian token dan memberikan pembelajaran yang lebih baik, dibandingkan dengan *feedforward neural network* standar yang hanya dapat memproses token tunggal.

Jika diperhatikan dari kompleksitas waktunya, total *epoch* yang dibutuhkan oleh model IndoBERT-LSTM lebih sedikit dari model *fine-tuned* IndoBERT. Waktu yang dibutuhkan untuk menyelesaikan 1 *epoch* hampir sama untuk kedua model, yaitu sekitar 3 – 4 menit. Karena total *epoch* dari model IndoBERT-LSTM lebih sedikit, maka total waktu pelatihan model ini juga lebih cepat dibandingkan dengan model *fine-tuned* IndoBERT. Hal ini menunjukkan proses pelatihan pada model IndoBERT-LSTM mampu mendapatkan *validation accuracy* tertinggi dengan lebih cepat.

5. Kesimpulan

Pada penelitian ini telah dibahas mengenai model klasifikasi teks berbasis *multiclass classification* pada *tweet* berbahasa Indonesia yang diberi nama IndoBERT-LSTM. Adapun kesimpulan yang dapat diambil dari penelitian yang telah dilakukan yaitu:

1. Berdasarkan hasil pengujian dan perbandingan, kombinasi model *pre- trained* IndoBERT dan *Long Short-Term Memory* (LSTM) terbukti dapat memberikan pemahaman yang lebih baik dalam mengklasifikasikan teks, baik pada *dataset* yang tidak termodifikasi maupun *dataset* yang telah termodifikasi.
2. Model IndoBERT-LSTM dengan skenario kombinasi *hyperparameter* terbaik (*batch size* sebesar 16, *learning rate* sebesar 2e-5, dan menggunakan *average pooling*) berhasil mendapatkan F1-*score* sebesar 98,90% pada *dataset* yang tidak termodifikasi (peningkatan 0,70% dari model Word2Vec-LSTM dan 0,40% dari model *fine-tuned* IndoBERT) dan 92,83% pada *dataset* yang telah termodifikasi (peningkatan 4,51% dari model Word2Vec-LSTM dan 0,69% dari model *fine-tuned* IndoBERT).
3. Peningkatan performa model IndoBERT-LSTM dari model *fine-tuned* IndoBERT tidak terlalu signifikan.
4. Total waktu pelatihan model Word2Vec-LSTM masih jauh lebih cepat, yaitu sekitar 3 menit untuk kedua *dataset*, dibandingkan dengan model IndoBERT-LSTM yang membutuhkan waktu sekitar 30 dan 45 menit. Akan tetapi, model IndoBERT-LSTM masih lebih cepat jika dibandingkan dengan model *fine-tuned* IndoBERT yang membutuhkan waktu sekitar 40 dan 55 menit.

5.1 Saran

Pada penelitian ini telah dikembangkan model klasifikasi teks berbasis *multiclass classification* pada *tweet* berbahasa Indonesia dengan kombinasi IndoBERT dan *Long Short-Term Memory* (LSTM). Oleh sebab itu, saran untuk penelitian-penelitian berikutnya bisa mengombinasikan IndoBERT dengan metode pengklasifikasi yang lebih bervariasi, seperti *Convolutional Neural Network* (CNN), *Bidirectional Long Short-Term Memory* (Bi-LSTM), dsb, atau dengan melatih model pada *multiclass dataset* berbasis emosi atau sentimen karena memiliki tingkat pemahaman bahasa yang lebih sulit.

Referensi

Alammar, J. (2018a, June 27). *The Illustrated Transformer – Jay Alammar – Visualizing machine learning one concept at a time.* https://jalammar.github.io/illustrated-transformer/

Alammar, J. (2018b, December 3). *The Illustrated BERT, ELMo, and co. (How NLP Cracked Transfer Learning)–Jay Alammar–Visualizing machine learning one concept at a time.* http://jalammar.github.io/illustrated-bert/

Alwehaibi, A., Bikdash, M., Albogmi, M., & Roy, K. (2021). A study of the performance of embedding methods for Arabic short-text sentiment analysis using deep learning approaches. *Journal of King Saud University-Computer and Information Sciences*.

Aydoğan, M., & Karci, A. (2020). Improving the accuracy using pre-trained word embeddings on deep neural networks for Turkish text classification. *Physica A: Statistical Mechanics and Its Applications*, 541, 123288. https://doi.org/10.1016/j.physa.2019.123288

Ayo, F. E., Folorunso, O., Ibharalu, F. T., & Osinuga, I. A. (2020). Machine learning techniques for hate speech classification of twitter data: State-of-The-Art, future challenges and research directions. *Computer Science Review*, 38, 100311. https://doi.org/10.1016/j.cosrev.2020.100311

Brownlee, J. (2021, January 18). *How to Choose an Activation Function for Deep Learning*. https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/

Cai, R., Qin, B., Chen, Y., Zhang, L., Yang, R., Chen, S., & Wang, W. (2020). Sentiment analysis about investors and consumers in energy market based onBERT-BILSTM. *IEEE Access*, *8*, 171408–171415. https://doi.org/10.1109/ACCESS.2020.3024750

Chauhan, N. S. (2021, August 2). *Loss Functions in Neural Networks*. https://www.theaidream.com/post/loss-functions-in-neural-networks

Chaumond, J., Delangue, C., & Wolf, T. (2016). *huggingface (Hugging Face)*. https://huggingface.co/huggingface

Cournapeau, D. (2007). *scikit-learn: machine learning in Python—scikit-learn 1.1.1 documentation*. https://scikit-learn.org/stable/#

Devlin, J., Chang, M.-W., Lee, K., Google, K. T., & Language, A. I. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. http://arxiv.org/abs/1810.04805

Digmi, I. (2018, January 25). *Memahami Epoch Batch Size Dan Iteration - JournalToday*. https://imam.digmi.id/post/memahami-epoch-batch-size-dan-iteration/

Géron, A. (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*. O'Reilly Media, Inc.

Google Brain Team. (2015, November 9). *TensorFlow*. https://www.tensorflow.org/

Goyal, A., Gupta, V., & Kumar, M. (2021). A deep learning-based bilingual Hindi and Punjabi named entity recognition system using enhanced word embeddings. *Knowledge-Based Systems*, 107601. https://doi.org/10.1016/j.knosys.2021.107601

Gupta, V., & Lehal Professor, G. S. (2009). *A Survey of Text Mining Techniques and Applications*. www.alerts.yahoo.com

Hilmiaji, N., Lhaksmana, K. M., & Purbolaksono, M. D. (2021). Identifying Emotion on Indonesian Tweets using Convolutional Neural Networks. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 5(3), 584–593. https://doi.org/10.29207/RESTI.V5I3.3137

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8), 1735–1780. https://doi.org/10.1162/NECO.1997.9.8.1735

Keras Team. (2015, March 27). *Dropout layer*. https://keras.io/api/layers/regularization_layers/dropout/

Koto, F., Rahimi, A., Lau, J. H., & Baldwin, T. (2020). *IndoLEM and IndoBERT: A Benchmark Dataset and Pre-trained Language Model for Indonesian NLP*. 757–770. https://doi.org/10.18653/v1/2020.coling-main.66

Kowsari, K., Meimandi, K. J., Heidarysafa, M., Mendu, S., Barnes, L., & Brown, D. (2019). Text Classification Algorithms: A Survey. *Information 2019, Vol. 10, Page 150*, 10(4), 150. https://doi.org/10.3390/INFO10040150

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings*. https://arxiv.org/abs/1301.3781v3

Muhammad, P. F., Kusumaningrum, R., & Wibowo, A. (2021). Sentiment Analysis Using Word2vec And Long Short-Term Memory (LSTM) For Indonesian Hotel Reviews. *Procedia Computer Science*, 179, 728–735. https://doi.org/10.1016/J.PROCS.2021.01.061

Nguyen, Q. T., Nguyen, T. L., Luong, N. H., & Ngo, Q. H. (2020). Fine-Tuning BERT for Sentiment Analysis of Vietnamese Reviews. *Proceedings – 2020 7th NAFOSTED Conference on Information and Computer Science, NICS 2020*, 302–307. https://doi.org/10.1109/NICS51282.2020.9335899

Pahwa, B., Kasliwal, N., Scholar, R., Vidyapith, B., & Taruna, R. S. (2018). Sentiment Analysis-Strategy for Text Pre-Processing Indianization and customization for Indian consumers View project Aspect level sentiment analysis View project Sentiment Analysis-Strategy for Text Pre-Processing Bhumika Pahwa. *Article in International Journal of Computer Applications*, 180(34), 975–8887. https://doi.org/10.5120/ijca2018916865

Putra, J. W. G. (2020). *Pengenalan Pembelajaran Mesin dan Deep Learning*.

Rahman, D. (2019). *deryrahman/word2vec-bahasa-indonesia: Word2Vec untuk bahasa Indonesia dari korpus Wikipedi* https://github.com/deryrahman/word2vec-bahasa-indonesia

Ramadhan, N. G. (2021). Indonesian Online News Topics Classification using Word2Vec and K-Nearest Neighbor. *Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi)*, 5(6), 1083–1089. https://doi.org/10.29207/RESTI.V5I6.3547

Rao, A., & Spasojevic, N. (2016). *Actionable and Political Text Classification using Word Embeddings and LSTM*. https://arxiv.org/abs/1607.02501v2

Robbani, H. A. (2018, September 24). *GitHub - har07/PySastrawi: Indonesian stemmer. Python port of PHP Sastrawi project.* PySastrawi.https://github.com/har07/PySastrawi

Sharma, A. K., Chaurasia, S., & Srivastava, D. K. (2020). Sentimental Short Sentences Classification by Using CNN Deep Learning Model with Fine Tuned Word2Vec. *Procedia Computer Science*, 167, 1139–1147. https://doi.org/10.1016/J.PROCS.2020.03.416

Sun, Z., Zemel, R., & Xu, Y. (2021). A computational framework for slang generation. *Transactions of the Association for Computational Linguistics*, 9, 462–478. https://doi.org/10.1162/TACL_A_00378/1921784/TACL_A_00378.PDF

Sutanto, T. (2020). *nlptm-01*. Tau-Data Indonesia. https://tau-data.id/d/nlptm- 01.html

Uysal, A. K., & Gunal, S. (2014). The impact of preprocessing on text classification. *Information Processing & Management*, 50(1), 104–112. https://doi.org/10.1016/J.IPM.2013.08.006

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł. ukasz, & Polosukhin, I. (2017). Attention is All you Need. In I. Guyon, U. v Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems* (Vol. 30). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Wang, Z., Huang, Z., & Gao, J. (2020). Chinese Text Classification Method Based on BERT Word Embedding. *ACM International Conference Proceeding Series*, 66–71. https://doi.org/10.1145/3395260.3395273

Wu, Y., Schuster, M., Chen, Z., Le, Q. v., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, Ł., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., … Dean, J. (2016). *Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation*. https://arxiv.org/abs/1609.08144v2