Sign Language Translator Using Deep Learning

1st Rizqi Alpiansyah
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
rizqialpiansyaht@student.telkomuniversity.ac.id

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

3rd Randy Erfa Saputra
Fakultas Teknik Elektro
Universitas Telkom
Bandung, Indonesia
resaputra@telkomuniversity.ac.id

***Abstrak*— Di era komunikasi digital saat ini, kesempatan untuk interaksi yang efektif antara orang tuli dan orang yang sulit mendengar menjadi lebih penting dari sebelumnya. Meskipun bahasa isyarat telah menjadi jembatan komunikasi yang penting, kemungkinan salah tafsir dan salah tafsir sering muncul sebagai tantangan yang menghambat komunikasi yang efektif. Mengidentifikasi kebutuhan mendesak akan solusi inovatif, penelitian ini bertujuan untuk mengembangkan aplikasi berbasis pembelajaran mesin yang diberdayakan secara khusus untuk berfokus terutama pada pengeditan teks. Dengan memanfaatkan kekuatan teknik LSTM, aplikasi ini dirancang tidak hanya untuk mengidentifikasi tetapi juga memperbaiki kesalahan interpretasi bahasa isyarat secara real time. Manfaat signifikan dari metode ini terbukti dalam peningkatan akurasi yang mencolok, mencapai 92,5% yang mengesankan. Selain itu, penelitian ini tidak hanya tentang teknologi tetapi juga tentang inklusi sosial. Dengan tujuan untuk mengurangi hambatan komunikasi dan mempromosikan interaksi yang lebih inklusif antara komunitas tuna rungu dan tuna rungu, aplikasi ini menandai langkah penting dalam merevolusi cara kita memahami, menerjemahkan, dan berkomunikasi melalui bahasa isyarat, memastikan bahwa setiap pesan tidak hanya dikirimkan tetapi diterima dengan kejelasan dan akurasi yang optimal.**

Kata kunci: **Machine Learning, SIBI, LSTM**

I. PENDAHULUAN

Dalam era digital yang semakin berkembang, aplikasi dengan fitur inovatif semakin menjadi bagian integral kehidupan sehari-hari. Salah satu fitur menarik yang telah memperkaya pengalaman pengguna adalah kemampuan untuk mengubah teks menjadi video animasi bahasa isyarat[1]. Fitur ini, yang dikenal sebagai "text to animation," telah membuka pintu baru untuk berkomunikasi secara lebih inklusif dengan menggunakan bahasa isyarat.

Namun, seperti halnya dalam berbagai bentuk komunikasi tulisan, keakuratan dalam penulisan tetap menjadi aspek yang tidak dapat diabaikan. Kesalahan penulisan dalam teks dapat memengaruhi pesan yang ingin disampaikan, terutama dalam konteks video animasi bahasa isyarat di mana setiap gerakan dan ekspresi memiliki arti yang dalam. Inilah tempat pentingnya peran "text correction" dalam konteks fitur text to animation.

Text correction merupakan mekanisme yang dirancang untuk mengoreksi kesalahan penulisan dalam teks sebelum diubah menjadi video animasi bahasa isyarat. Kegunaan utama text correction adalah memastikan bahwa pesan yang hendak disampaikan dalam bentuk animasi memiliki ketepatan dan kejelasan dalam penulisan. Dengan melakukan koreksi pada kesalahan ejaan, tata bahasa, dan sintaksis, text correction membantu memastikan bahwa teks yang diterjemahkan menjadi animasi bahasa isyarat benar-benar mencerminkan maksud komunikasi yang diinginkan[2].

Dalam konteks fitur text to animation, text correction menjadi pilar penting dalam menjembatani kesenjangan antara bahasa tulisan dan bahasa isyarat. Dengan memiliki mekanisme koreksi yang efektif, aplikasi ini tidak hanya mampu menghasilkan animasi yang lebih akurat, tetapi juga memberikan pengalaman yang lebih kuat dan bermakna bagi semua pengguna, termasuk mereka yang menggunakan bahasa isyarat sebagai bentuk komunikasi utama[3].

II. KAJIAN TEORI

A. Text Correction

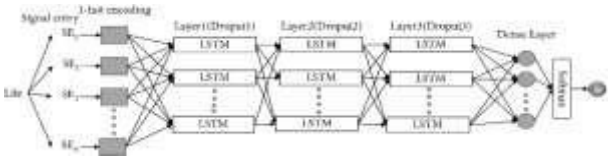
Ketidaktepatan dalam pengejaan saat menulis dapat menurunkan efektivitas, sehingga pesan yang ingin disampaikan bisa dimengerti secara berbeda. Berdasarkan Damerau, sekitar 80% kesalahan pengejaan muncul karena empat faktor utama:

1. Substitusi, merupakan kesalahan yang terjadi akibat pengubahan satu huruf dalam sebuah kata.
2. Insertion, adalah kesalahan yang muncul karena penambahan satu huruf dalam kata.
3. Deletion, terjadi karena penghapusan satu huruf dari kata.
4. Transposition, dihasilkan dari pertukaran posisi dua huruf yang berdampingan.

Spell checker berfungsi untuk mengidentifikasi kesalahan pengejaan dalam suatu teks. Setelah mendeteksi kesalahan, langkah selanjutnya adalah menawarkan kata pengganti yang tepat. Kata pengganti ini merupakan kata yang umumnya ada dalam kamus dan dipilih berdasarkan sejauh mana kemiripannya dengan kata yang dieja salah. Dalam melakukan koreksi, ada dua metode yang bisa diterapkan: secara manual, di mana pengguna memilih kata pengganti dari saran yang diberikan, dan secara otomatis, di mana sistem langsung memilih dan menggantikan kata yang salah dengan kata yang memiliki kesamaan tertinggi[4].

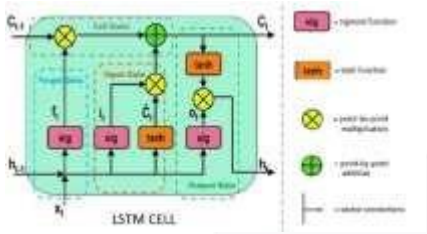
B. LSTM

Model ini menggunakan algoritma utama LSTM. LSTM digunakan untuk melatih data gerakan, seperti koordinat gestur, dengan label bahasa isyarat yang telah ditentukan. Arsitektur LSTM terdiri dari data masukan, proses data, unit LSTM, dense layer, optimizer, dan output[5].



GAMBAR 1
Arsitektur LSTM

Long Short-Term Memory (LSTM) adalah jenis arsitektur jaringan saraf berulang yang mengatasi masalah hilangnya gradien dalam pelatihan RNN. LSTM memiliki "*gated cell*" dengan empat bagian: *cell state*, *input gate, output gate,* dan *forget gate*. Ini memungkinkan LSTM mempertahankan informasi jangka panjang dan memutuskan informasi yang penting. LSTM lebih baik dalam belajar dari dependensi waktu dibandingkan RNN biasa[6].



GAMBAR 2
LSTM *Cell*[12]

$$f_t = \sigma\,(w_f \cdot [h_{t-1}, x_t] + b_f) \quad (1)$$

Forget gate bertugas untuk menentukan informasi mana yang perlu diperhatikan dan mana yang dapat diabaikan. Data dari input saat ini x_t dan state tersembunyi sebelumnya h_{t-1} diolah melalui fungsi sigmoid. Fungsi sigmoid menghasilkan nilai antara 0 dan 1, yang menunjukkan apakah beberapa aspek output sebelumnya pentingatau tidak. Nilai hasil f_t nantinya digunakan oleh sel untuk perkalian berdasarkan elemen.

$$i_t = \sigma\,(w_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{C}_t = \tanh\,(w_C \cdot [h_{t-1}, x_t] + b_C)$$

Input Gate melakukan operasi-operasi berikut untuk memperbarui status sel. Pertama, status saat ini x_t dan status tersembunyi sebelumnya h_{t-1} dilewatkan melalui fungsi sigmoid kedua. Nilai-nilai ini diubah antara 0 (penting) dan 1 (tidak penting)

Kemudian, informasi yang sama dari status tersembunyi dan status saat ini akan dilewatkan melalui fungsi tanh. Untuk mengatur jaringan, operator tanh akan membuat vektor $\tilde{C}_t$ dengan semua nilai mungkin antara -1 dan 1. Nilai-nilai keluaran yang dihasilkan dari fungsi aktivasi siap untuk perkalian berdasarkan elemen.

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \quad (3)$$

Jaringan memiliki informasi yang cukup dari gerbang lupakan dan gerbang masukan. Langkah selanjutnya adalah memutuskan dan menyimpan informasi dari status baru dalam status sel. Status sel sebelumnya C_{t-1} akan dikalikan dengan vektor lupakan f_t. Jika hasilnya adalah 0, maka nilainilai akan diabaikan dalam status sel. Selanjutnya, jaringan mengambil nilai keluaran vektor masukan i_t dan melakukan penambahan berdasarkan elemen, yang memperbarui status sel memberikan jaringan status sel baru C_t.

$$o_t = \sigma\,(w_o \cdot [h_{t-1}, x_t] + b_o) \quad (4)$$
$$h_t = o_t \times \tanh(C_t)$$

Gerbang keluaran menentukan nilai dari state tersembunyi berikutnya. State ini berisi informasi tentang input sebelumnya. Pertama, nilai dari status saat ini dan status tersembunyi sebelumnya dilewatkan melalui fungsi sigmoid ketiga. Kemudian, status sel baru yang dihasilkan dari status sel dilewatkan melalui fungsi tanh. Kedua hasil keluaran ini dikalikan berdasarkan elemen. Berdasarkan nilai akhir, jaringan menentukan informasi apa yang harus dibawa oleh state tersembunyi. State tersembunyi ini digunakan untuk prediksi. Akhirnya, status sel baru dan state tersembunyi baru diangkut ke langkah waktu berikutnya.

RNN memiliki fitur krusial: kemampuan mengingat informasi dari data sebelumnya melalui lingkaran balik. Namun, RNN menghadapi masalah ketergantungan jangka panjang saat memproses data besar[7]. LSTM mengatasi hal ini dengan sel memori untuk menjaga informasi jangka panjang. Dalam deep learning, Dense adalah layer yang sepenuhnya terkoneksi, sementara LSTM adalah jenis RNN untuk ketergantungan jangka panjang. Menggunakan lapisan Dense setelah beberapa lapisan LSTM bermanfaat untuk model yang berurutan dan berorientasi prediksi atau klasifikasi[8].

III. METODE

GAMBAR 3
Arsitektur LSTM

Proses dimulai dengan pengumpulan data dari tweet dan artikel berita. Setelah itu, data dibersihkan dengan menghapus tanda baca, angka, mengkonversi teks menjadi huruf kecil, serta menghapus karakter non-alfabet. Selanjutnya, kata-kata dicek dalam KBBI menggunakan teknik scraping dari situs web kbbi.web.id. Setelah itu, dilakukan pembuatan kata-kata salah dengan teknik tertentu. Hasilnya disimpan dalam file CSV dengan kolom index, true, dan false. Selanjutnya, data ditokenisasi dan dijalani proses padding untuk menjalankan model dengan efisien. Model yang digunakan adalah LSTM dan akan dikompilasi dan dilatih. Performa atau akurasi model dievaluasi, dan jika perlu, proses mendefinisikan model dan pelatihan diulang hingga mencapai akurasi yang diinginkan. Jika akurasi baik, model akan disimpan, dan proses implementasi sistem deep learning selesai.

A. Koleksi data

Mengumpulkan data dalam sistem text-to-animation penting untuk akurasi dan keandalan sistem. Sumber data utama adalah berita aktual dan Dataset dari universitas Leipzig dianggap berkualitas dan legal, sementara data dari Twitter membantu sistem mengatasi kasus khusus dan percakapan informal. Kombinasi kedua sumber data ini diharapkan meningkatkan performa sistem text-to-animation.

B. Data cleaning

Pembersihan data adalah proses penting dalam pengolahan bahasa alami untuk membersihkan dan mempersiapkan teks mentah sebelum diolah lebih lanjut. Langkah-langkah dalam pembersihan data meliputi penghapusan tanda baca, angka, konversi teks menjadi huruf kecil, dan penghapusan karakter non-alfabet[9].

TABEL 1
Tabel Proses Data Cleaning

Fungsi	*Input*	*Output*
Menghapus Tanda Baca	Hari ini adalah hari yang cerah! Saya senang sekali.	Hari ini adalah hari yang cerah Saya senang sekali
Menghapus Angka	Tahun 2023 akan menjadi tahun yang luar biasa!	Tahun akan menjadi tahun yang luar biasa
Mengubah Teks Menjadi Huruf Kecil	Proyek ini Sedang Dalam Pengembangan	proyek ini sedang dalam pengembangan
Menghapus Karakter Non-Abjad	[Jangan terlalu @berlebihan!!!!!!!]	Jangan terlalu berlebihan

C. Cek kata dalam KBBI

GAMBAR 4
Halaman KBBI

Pada Gambar 4, tampilan halaman KBBI saat mencari kata "makan". Sistem menggunakan web scraping untuk request ke KBBI dan cek hasil pencarian. Di halaman hasil, sistem cari elemen HTML class "highlight" yang berisi arti kata. Bila ada, teks elemen itu dibandingkan dengan kata yang dicari. Jika cocok, arti kata disimpan untuk digunakan dalam text correction.

D. Membuat dataset dari kata yang sudah divalidasi

Pembuatan dataset berdasarkan paragraf di bawah ini melibatkan ketidaktepatan dalam pengejaan saat menulis, yang dapat menyebabkan pesan yang ingin disampaikan dimengerti dengan cara yang berbeda.

Tabel 2 Contoh Isi Pada Dataset

Teknik	Kata Awal	Kata Hasil
Menambahkan	rumah	rhumuah, ruxmah
Menghapus	mobil	mobil, mbl
Menukar	lampu	lapmu, lmapu
Mengganti	kamar	kamar, kamal

E. Pengolahan Data

Pengolahan data dimulai dengan memuat file CSV ke dalam dataframe, yaitu tabel terstruktur. Pengecekan data kosong menggunakan "isnull()" atau "isna()" pada dataframe, karena data kosong bisa mengganggu analisis. Data kosong dapat dihapus untuk penanganan. Pemisahan data menjadi latih dan uji penting untuk model text correction. Data latih memperbaiki kesalahan teks, sementara data uji menguji model pada teks tak dikenal, mencegah overfitting. Penggunaan data uji saat latihan memastikan model generalisasi pada data baru. Data splitting menjamin model text correction lebih handal pada teks sebenarnya dalam aplikasi teks..

F. Tokenization

Tokenization adalah langkah krusial dalam text correction, yakni membagi teks menjadi unit-unit kecil seperti kata atau karakter. Ini memungkinkan model mengenali kata secara individual untuk memperbaiki kesalahan ejaan atau mendeteksi kata tak sesuai[10].

TABEL 3
Proses Tokenisasi

No	Sebelum tokenisasi	Sesudah tokenisasi
1	adan	[1, 15, 1, 2]
2	bdan	[14, 15, 1, 2]
3	cdan	[20, 15, 1, 2]
4	ddan	[15, 15, 1, 2]
5	edan	[3, 15, 1, 2]

Tabel 3 menunjukkan contoh tokenisasi teks. Misalnya, teks "adan" dipecah menjadi token-token seperti "[1, 15, 1, 2]". Hasil tokenisasi untuk teks lainnya ditunjukkan dalam kolom "sesudah tokenisasi" sebagai daftar token terpisah.

G. Padding

Padding adalah teknik umum dalam pemrosesan bahasa alami, termasuk dalam proyek text correction. Tujuannya adalah menyamakan panjang teks atau kalimat agar dapat diproses dalam model deep learning yang memerlukan ukuran input yang konsisten. Dalam text correction, panjang kata mungkin berbeda, sedangkan model deep learning memerlukan ukuran input tetap. Padding digunakan untuk menambahkan token pada akhir kalimat sehingga semua kalimat memiliki panjang seragam[11].

TABEL 4
Proses *Padding*

No	Sebelum tokenisasi	Sesudah tokenisasi	Sesudah melakukan padding
1	adan	[1, 15, 1, 2]	[1, 15, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,]
2	bdan	[14, 15, 1, 2]	[14, 15, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,]
3	cdan	[20, 15, 1, 2]	[20, 15, 1, 2, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,]

Tabel tersebut menunjukkan contoh penggunaan padding pada dataset. Padding dilakukan untuk membuat panjang

setiap kalimat seragam, sesuai dengan panjang maksimum yang ditetapkan sebelumnya. Teknik ini melibatkan penambahan angka nol pada bagian akhir kalimat sehingga panjangnya mencapai panjang maksimum.
Misalnya, kalimat "adan" dengan token [1, 15, 1, 2] akan dipadding menjadi [1, 15, 1, 2, 0, 0, 0, ..., 0] untuk mencapai panjang maksimum yang telah ditetapkan.

H. Pembuatan Kategori Pada Data

Proses ini melibatkan mengubah kolom indeks dalam DataFrame menjadi kategori. Langkah-langkahnya melibatkan mengidentifikasi nilai unik dalam kolom, memetakan nilai-nilai ini menjadi angka berurutan, dan akhirnya mengonversi angka menjadi kategori menggunakan one-hot encoding. Ini penting dalam pelatihan model machine learning, terutama untuk klasifikasi.

I. Menyiapkan Model

GAMBAR 5
Model Summary

Model ini berstruktur secara berurutan dengan beberapa layer. Pertama, ada layer Embedding yang mengubah katakata dalam teks menjadi vektor numerik. Pengaturan Input_dim menentukan jumlah token unik dalam data, sementara Output_dim menentukan dimensi vektor untuk tiap token. Selanjutnya, ada layer LSTM yang berfungsi menganalisis data berurutan, seperti urutan kata dalam teks. Layer LSTM membantu model memahami konteks dan pola urutan kata. Akhirnya, ada Dense layer dengan aktivasi softmax untuk melakukan klasifikasi pada hasil dari LSTM dan menghasilkan prediksi probabilitas kelas yang berbeda.

J. Training model

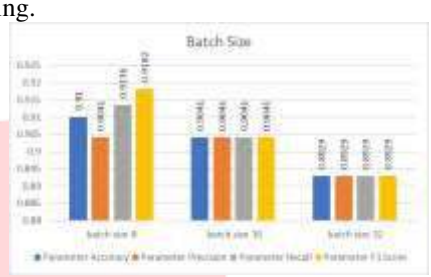
Pelatihan model untuk text correction melibatkan data yang telah di-padding dan di-encode menjadi one-hot encoding. Proses ini menggunakan fungsi *fit* pada model dengan iterasi data dalam beberapa epoch untuk mengoptimasi parameter dan meningkatkan akurasi pada data latih. Hasil pelatihan disimpan dalam variabel history yang mencatat perubahan nilai loss dan metrik lain pada setiap epoch. Informasi ini membantu menganalisis kinerja model dan memungkinkan penyesuaian untuk peningkatan performa text correction. Pelatihan diharapkan menghasilkan model yang lebih baik dalam koreksi teks dengan akurasi dan efektivitas yang lebih tinggi.

IV. HASIL DAN PEMBAHASAN

Pada Bagian ini akan membahas tentang pengujian pada model yang sudah dibuat

GAMBAR 6
Hasil Pengujian Partisi Data

Eksperimen dengan variasi training-test ratio mengindikasikan bahwa peningkatan jumlah training data meningkatkan performa model, termasuk akurasi, precision, recall, dan F1 Score. Pembagian data 90:10, 80:20, dan 70:30 menunjukkan hasil baik dengan akurasi di atas 88% dan F1 Score di atas 0.85. Namun, pada pembagian 60:40 dan 50:50, performa model menurun drastis, terutama 60:40 dengan akurasi 79.7% dan F1 Score 0.7544. Oleh karena itu, untuk hasil lebih baik, sebaiknya gunakan lebih banyak data sebagai data training.

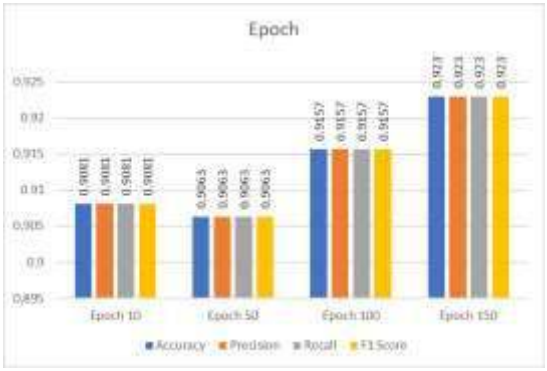


GAMBAR 7
Hasil Pengujian *Batch Size*

Hasil eksperimen dengan berbagai ukuran batch menunjukkan dampak signifikan pada performa model. Batch size 8 menghasilkan kinerja terbaik dengan akurasi sekitar 91% dan F1 Score 0.9182. Namun, saat batch size ditingkatkan menjadi 16, performa sedikit menurun dengan akurasi dan F1 Score sekitar 90.4%. Batch size 32 mengakibatkan penurunan performa lebih lanjut menjadi akurasi 89.3% dan F1 Score 0.8929. Ini menunjukkan bahwa batch size 8 adalah pilihan terbaik untuk prediksi yang lebih akurat. Batch size yang terlalu besar dapat merugikan performa model. Oleh karena itu, sebaiknya menggunakan batch size 8 untuk hasil prediksi lebih baik.

GAMBAR 8
Hasil Pengujian *Optimizer*

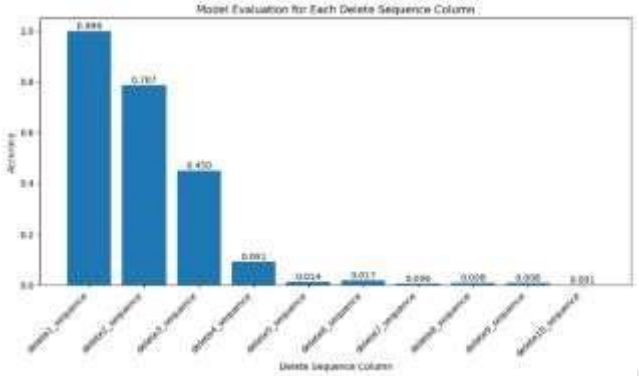
Berdasarkan hasil evaluasi model dengan menggunakan berbagai optimizer, dapat disimpulkan bahwa optimizer Adam memberikan hasil prediksi terbaik dengan akurasi sekitar 90.81% dan nilai F1 Score 0.9081. Hal ini menunjukkan bahwa Adam mampu mengoptimalkan proses training model dengan baik dan menghasilkan prediksi yang lebih akurat dibandingkan optimizer lainnya yang diuji. Di sisi lain, optimizer SGD, RMSprop, dan Adagrad menunjukkan performa yang jauh lebih rendah dengan akurasi dan F1 Score di bawah 10%. Oleh karena itu, untuk tugas ini, penggunaan optimizer Adam disarankan untuk mencapai hasil prediksi yang lebih baik.



GAMBAR 9
Hasil Pengujian *Epoch*

Dari Gambar 8, disimpulkan model hasil pelatihan memiliki kinerja klasifikasi sangat baik. Selama empat epoch (10, 50, 100, dan 150), model tetap konsisten dan mencapai akurasi, presisi, recall, serta nilai F1 yang tinggi. Akurasi di atas 90% menunjukkan kemampuan model dalam mengklasifikasikan data secara benar. Presisi, recall, dan F1 yang tetap tinggi menunjukkan model dapat mengenali dan membedakan berbagai kelas data dengan baik. Keseluruhan, hasil ini menunjukkan efektivitas dan keandalan model klasifikasi yang dapat digunakan dalam tugas serupa.

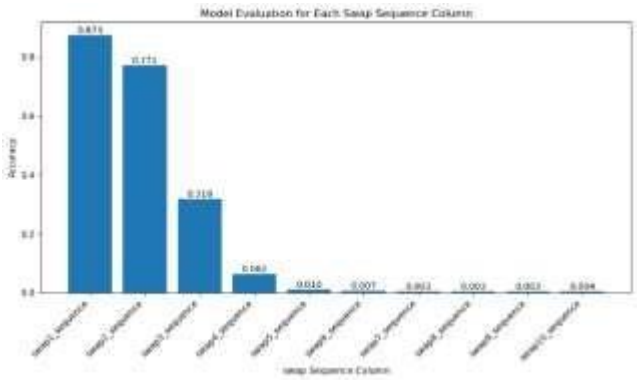
Model yang sudah dibuat juga dilakukan pengujian dengan menghapus huruf pada kata. Dimulai dari menghapus 10% dari total huruf pada kata sampai menghapus 100% huruf pada kata.



GAMBAR 10
Hasil Pengujian Menghapus Huruf Pada Kata

Berdasarkan Gambar 9 kita dapat menyimpulkan bahwa proses penghapusan kata memiliki dampak yang signifikan pada kinerja model. Tingkat akurasi model menurun secara drastis seiring dengan peningkatan persentase penghapusan kata. Pada tingkat penghapusan 10%, model masih memiliki akurasi sebesar 85%. Namun, ketika 50% kata dihapus, akurasi menurun tajam menjadi 13%, dan terus menurun dengan peningkatan persentase penghapusan kata. Hal ini menunjukkan bahwa penghapusan kata mengakibatkan banyak informasi penting hilang dari teks, sehingga kemampuan model untuk melakukan prediksi menurun secara signifikan.

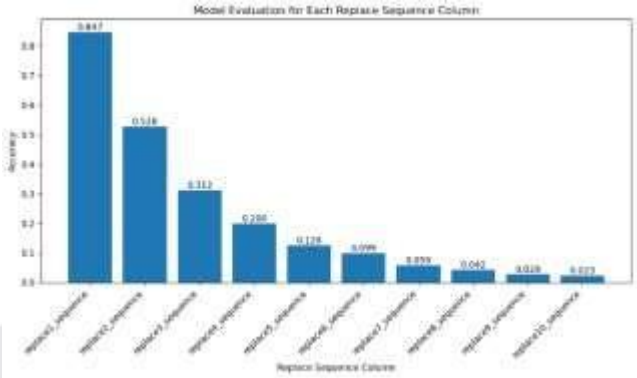
Selanjutnya model di ujikan dengann menukar huruf pada kata. Pengujianya dilakukan dengan menukar 1 kali susunan huruf pada kata dan dilakukan sampai 10 kali penukaran.



GAMBAR 11
Hasil Pengujian Menukar Huruf Pada Kata

dapat disimpulkan bahwa proses menukar huruf dalam teks memiliki dampak yang signifikan pada kinerja model. Pada awalnya, ketika hanya terjadi pertukaran 1 huruf, akurasi model masih tinggi sebesar 87.3%. Namun, ketika jumlah pertukaran huruf meningkat, performa model menurun secara tajam. Pada tingkat 5 pertukaran huruf, akurasi turun menjadi hanya 1%, dan semakin menurun seiring dengan peningkatan jumlah pertukaran huruf. Hal ini menunjukkan bahwa proses menukar huruf dapat menyebabkan hilangnya struktur dan makna dalam teks, sehingga model tidak dapat melakukan prediksi dengan baik.

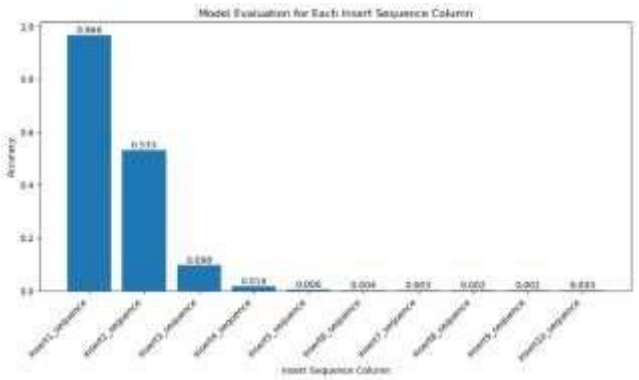
Pada pengujian berikutnya yaitu menguji penukaran huruf pada kata. Pengujiann ini dimulai dengan menukar 1 kali huruf yang ada pada kata dan dilakukan sampai menukar 10 kali huruf pada kata tersebut.



GAMBAR 12
Hasil Pengujian Mengganti Huruf Pada Kata

Pada Gambar 12 dapat disimpulkan bahwa proses mengganti huruf dalam teks juga berdampak signifikan terhadap performa model. Semakin banyak penggantian huruf yang terjadi, akurasi model menurun secara drastis. Pada tingkat penggantian satu huruf, akurasi model masih mencapai 84.7%. Namun, ketika dilakukan penggantian hingga sepuluh huruf, akurasi model turun menjadi hanya sekitar 2.3%. Hal ini menunjukkan bahwa proses mengganti huruf dapat menyebabkan banyak informasi penting dalam teks hilang, sehingga kemampuan model dalam melakukan prediksi menurun secara drastis

Selanjutnya dilakukan uji dengan memasukan huruf pada sela-sela huruf dalam kata. Proses ini akan menambahkan 1 huruf sampai 10 huruf secara acak.



GAMBAR 13
Hasil Pengujian Mengganti Huruf Pada Kata

Berdasarkan hasil pengujian pada Gambar 13 yang sudah dilakukan, dapat disimpulkan bahwa proses memasukkan huruf dalam teks memiliki dampak signifikan terhadap performa model. Pada tingkat penambahan satu huruf, akurasi model masih tinggi mencapai 96.6%. Namun, ketika dilakukan penambahan huruf hingga sepuluh huruf, akurasi model menurun drastis menjadi hanya sekitar 0.3%. Hal ini menunjukkan bahwa penambahan huruf dapat menyebabkan ketidakcocokan besar antara input yang diharapkan dan input sebenarnya yang diberikan ke model, sehingga kemampuan model dalam melakukan prediksi menjadi tidak dapat diandalkan

V. KESIMPULAN

Dalam analisis hasil pengujian text correction, temuan penting terkait dampak berbagai jenis perubahan teks pada kinerja model telah ditemukan. Penghapusan huruf, sebagai contoh, memiliki dampak signifikan terhadap kinerja model. Semakin banyak kata dihapus, akurasi model menurun drastis karena hilangnya informasi penting. Misalnya, pada penghapusan 10%, akurasi masih 85%, tetapi pada penghapusan 50%, akurasi tajam turun menjadi 13%. Analisis pertukaran huruf (swap) menunjukkan dampak serupa, di mana pertukaran satu huruf mempertahankan akurasi 87.3%, namun dengan lima pertukaran huruf, akurasi menurun menjadi 1%.

Penggantian huruf (replace) juga memiliki dampak serupa. Pada satu penggantian huruf, akurasi tetap 84.7%, namun pada penggantian sepuluh huruf, akurasi turun drastis menjadi 2.3%. Penambahan huruf (insert) juga signifikan, dengan satu penambahan akurasi 96.6%, tetapi dengan sepuluh penambahan, akurasi turun menjadi 0.3%. Temuan ini menekankan pentingnya mempertimbangkan dampak perubahan teks pada performa model text correction. Hal ini memberikan wawasan penting bagi pengembangan sistem koreksi teks yang lebih efektif dan akurat, serta perlunya mempertimbangkan pengaruh akurasi pada tingkat perubahan teks.

REFERENSI

[1] R. Kumar Attar, V. Goyal, and L. Goyal, “State of the art of automation in sign language: A systematic review,” ACM Transactions on Asian and Low-Resource Language Information Processing, vol. 22, no. 4, pp. 1–80, 2023. doi:10.1145/3564769

[2] I. Kissos and N. Dershowitz, “OCR error correction using character correction and feature-based word classification,” 2016 12th IAPR Workshop on Document Analysis Systems (DAS), 2016. doi:10.1109/das.2016.44

[3] W. Zhuang, J. Qi, P. Zhang, B. Zhang, and P. Tan, “Text/speech-driven full-body animation,” Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, 2022. doi:10.24963/ijcai.2022/863

[4] A. P. Wibawa, P. Yuliawati, P. Santoso, R. Shalahuddin, and I. M. Wirawan, “Damerau Levenshtain Distance Dengan metode Empiris Untuk Koreksi ejaan Bahasa Indonesia,” ILKOM Jurnal Ilmiah, vol. 12, no. 3, pp. 176–182, 2020. doi:10.33096/ilkom.v12i3.600.176-182

[5] C. C. Chatterjee, “Implementation of RNN, LSTM, and gru,” Medium, https://towardsdatascience.com/implementation-of-rnnlstm-and-gru-a4250bf6c090 (accessed Aug. 8, 2023).

[6] E. Elbasani and J.-D. Kim, “Llad: Life-log anomaly detection based on Recurrent Neural Network LSTM,” Journal of Healthcare Engineering, vol. 2021, pp. 1–7, 2021. doi:10.1155/2021/8829403

[7] C. Kang, “RNN - many-to-many,” Chan`s Jupyter, https://goodboychan.github.io/python/deep_learning/tensorflow-keras/2020/12/09/01-RNN-Many-to-many.html (accessed Aug. 9, 2023).

[8] K. Team, “Keras documentation: Dense layer,” Keras, https://keras.io/api/layers/core_layers/dense (accessed Aug. 9, 2023).

[9] K. Rastogi, “Text cleaning methods in NLP,” Analytics Vidhya, https://www.analyticsvidhya.com/blog/2022/01/textcleaning-methods-in-nlp/ (accessed Aug. 9, 2023).

[10] N. Project, NLTK, https://www.nltk.org/api/nltk.tokenize.html (accessed Aug. 9, 2023).

[11] A. Lopez-del Rio, M. Martin, A. Perera-Lluna, and R. Saidi, “Effect of sequence padding on the performance of deep learning models in archaeal protein functional prediction,” Scientific Reports, vol. 10, no. 1, 2020. doi:10.1038/s41598-020-71450-8

[12] G. Singhal, “Gaurav Singhal,” Pluralsight, https://www.pluralsight.com/guides/introduction-tolstm-units-in-rnn (accessed Aug. 9, 2023).