**ANALISIS KESESUAIAN KONTEKS SARAN DAN KRITIK SEBUAH KALIMAT PADA SIMAK UNISMUH MENGGUNAKAN ALGORITMA LSTM**

## **ROSALINDA APRILIA SARI**

UNIVERSITAS MUHAMMADIYAH MAKASSAR

e-mail: xxxx@xxxx.xxx\*

***Abstract***

*Rosalinda Aprilia Sari. Analysis of the Contextual Appropriateness of Suggestions and Criticisms in Sentences on the Simak Unismuh Platform Using LSTM Algorithm (supervised by Fachrim Irhamna Rahman S.Kom., M.T. and Risky Yusliana Bakti S.T., M.T.).*

*In an effort to enhance the academic experience and administrative systems, Universitas Muhammadiyah Makassar (Unismuh) has prioritized the use of technology, including providing a feedback platform for students to express their views, suggestions, and criticisms regarding various aspects of the campus. However, the data collection process often encounters challenges due to the lack of constructive responses and incomplete information, hindering effective improvements. This study aims to develop a strategy using the Long Short Term Memory (LSTM) algorithm to analyze the context of suggestions and criticisms on the Unismuh satisfaction survey platform regarding the services and facilities offered. The LSTM algorithm was chosen for its ability to extract complex patterns in textual data. The results of this study indicate that the LSTM model, tested with various training and validation data ratio*

*Keywords: Context analysis, Long Short Term Memory, Natural Language Processing*

**Abstrak**

ROSALINDA APRILIA SARI. Analisis Kesesuaian Konteks Saran Dan Kritik Sebuah Kalimat Pada Simak Unismuh Menggunakan Algoritma LSTM (dibimbing oleh Fachrim Irhamna Rahman S.Kom., M.T. dan Risky Yusliana Bakti S.T., M.T.).

Dalam upaya meningkatkan pengalaman akademik dan sistem administrasi, Universitas Muhammadiyah Makassar (Unismuh) telah mengedepankan teknologi, termasuk penyediaan platform hak angket bagi mahasiswa untuk menyampaikan pandangan, saran, dan kritik mengenai berbagai aspek kampus. Namun, pengumpulan data ini sering terkendala oleh kualitas tanggapan yang kurang konstruktif dan informasi yang tidak lengkap, sehingga menghambat perbaikan yang efektif. Penelitian ini dilakukan dengan maksud mengembangkan strategi penggunaan algoritma Long Sort Term Memory (LSTM) untuk menganalisis konteks saran dan kritik pada platform survei kepuasan simak unismuh pada layanan dan fasilitas yangg ditawarkan. Penggunaan algoritma LSTM dipilih karena kemampuan dalam mengekstraksi pola – pola kompleks dalam data teks. Hasil penelitian ini menunjukan bahwa model LSTM, yang diuji dengan berbagai rasio data pelatihan dan validasi

Kata Kunci: Analisis Konteks, Long Short Term Memory, Natural Language Processing

**1. Pendahuluan**

1. Dalam konteks pembelajaran dan pengembangan institusi pendidikan seperti universitas semakin mengedepankan teknologi guna meningkatkan pengalaman akademik hingga sistem administrative yang dapat ditawarkan kepada mahasiswa. Salah satu upaya yang dilakukan adalah dengan menyediakan platfrom survei kepuasan, tidak terkecuali pada Universitas Muhammadiyah Makassar (Unismuh). Platform ini dapat digunakan untuk menyampaikan pandangan, pengalaman, saran, serta kritik terhadap berbagai aspek kampus termaksud kualitas layanan dan pengalaman yang diberikan. Dalam hal ini, langkah penting yang dapat dilakukan adalah mendapatkan sumber informasi berupa saran dan kritik yang relevan dari para responden.

2. Namun, efektifitas dari pengumpulan data tersebut sering kali terpengaruh oleh kualitas tanggapan yang diberikan oleh responden dan dalam praktiknya, terdapat kecenderungan dimana responden hanya mengisi jawaban tanpa memberikan umpan balik berupa saran dan kritikan yang konstruktif. Hal ini menyebabkan evaluasi terhadap pengalaman dan kualitas layanan dikampus menjadi tidak efisien, karena informasi yang diberikan tidak memberikan gambaran yang lengkap atau manfaat bagi perbaikan.

3. Untuk mengatasi permasalahan tersebut, diperlukan pendekatan yang lebih cermat dalam menganalisa konteks saran dan kritik yang disampaikan oleh mahasiswa. Salah satu pendekatan yang dapat digunakan adalah dengan memanfaatkan pengolahan Natural Language Processing (NLP) dan deep learning, seperti algoritma Long Short Term Memory (LSTM).

4. Disebut pula bahwa LSTM merupakan jaringan saraf tiruan dengan arsitektur yang fleksibel sehingga dapat diterapkan pada berbagai aplikasi yang berbeda (Setiawan et al., 2022). Dengan mengintegrasikan LSTM kedalam survei kepuasan diharapkan dapat mengembangkan model yang dapat mengekstraksi pola-pola kompleks dalam data text data sehingga dapat menentukan apakah suatu jawaban mengandung saran atau kritik yang sesuai dengan konteks evaluasi pengalaman dan pelayanan akademik di Unismuh.

5. Dengan melakukan penelitian menggunakan algoritma LSTM untuk menganalisis kesesuain konteks saran dan kritik pada survei kepuasan simak Unismuh, diharapkan dapat memberikan kontribusi dalam meningkatkan efektifitas pada evaluasi pelayanan serta memperbaiki pengalaman mahasiswa di lingkungan akademik. Oleh karena itu, penelitian perlu dilakukan untuk memberikan kontribusi yang signifikan dalam penggunaan teknologi dalam mendukung proses evaluasi dan perbaikan kontinu di Unismuh

Berdasarkan pada latar belakang masalah yang telah diuraikan, dapat dirumuskan masalah sebagai berikut:

1. Bagaimana strategi penggunaan dan keberhasilan algoritma LSTM dalam menganalisis serta mengklasifikasikan konteks saran dan kritik pada survei kepuasan layanan akademik unismuh.

Berdasarkan rumusan masalah yang telah dipaparkan diatas maka penelitian ini memiliki dua tujuan sebagai berikut:

1. Bertujuan untuk mengembangkan strategi penggunaan algoritma LSTM untuk menganalisis konteks saran dan kritik serta mengevaluasi keberhasilan algoritma LSTM dalam dalam mengklasifikasikan konteks saran dan kritik guna mengevaluasi dan meningkatkan layanan mutu pendidikan.

Dari hasil penelitian ini, penulis berharap agar manfaat penelitian ini tidak hanya berguna bagi penulis sendiri tetapi juga bagi masyarakat umum, terutama Pelajar. Adapun manfaat yang diharapkan dari penelitian ini adalah :

1. Bagi Masyarakat

2. Meningkatkan partisipasi masyarakat khususnya mahasiswa dalam memberikan pendapat, pandangan, saran, serta kritik yang lebih bermakna, sehingga dapat memberikan gambaran yang lebih akuran tentang kebutuhan dan harapan mahasiswa dan dosen terhadap lingkungan akademik.

3. Bagi Mahasiswa

4. Mendorong pengembangan metode analisis teks yang inovatif, seperti algoritma LSTM, yang dapat diterapkan dalam konteks evaluasi dan perbaikan layanan institusi pendidikan lainnya.

5. Menjadi dasar dan acuan untuk penelitian mendatang yang berhubungan dengan algoritma LSTM

6. Penelitian ini akan difokuskan pada Universitas Muhammadiyah Makassar, dengan penekana pada evaluasi layanan, fasilitas, serta kualitas pendidikan yang disediakan oleh pihak kampus untuk mahasiswa dan dosen

7. Penelitian ini akan memperhatikan jenis saran dan kritik yang dikumpulkan melalui platform survei kepuasan Unismuh.

8. Penelitian akan menggunakan algoritma LSTM dalam analisis sentiment

9. Adapun pola umum dalam penulisan ini antara lain :

**2. Metode Penelitian**

1. Tempat Penelitian

2. Penelitian ini dilaksanakan dalam kondisi daring dengan mengumpulkan kontek saran dan kritik pada survei kepuasan simak unismuh

3. Waktu Penelitian

4. Penelitian ini akan berlangsung mulai Mei hingga Juli 2024

5. Kebutuhan Hardware

6. Laptop Asus X441UA

7. Kebutuhan Software

8. Google Colab

9. Python

10. Excel

Perancang sistem adalah proses rincian dalam pengembangan perangkat lunak dimana konsep dan kebutuhan pengguna diterjemahkan menjadi desain teknik yang spesifik. Tujuan dari perancangan ini adalah untuk merancang struktur dan fungsi system secara keseluruhan. Hal ini melibatkan pembuatan diagram, skema, dan spesifikasi teknis yang lebih detail guna memandu tahapan implementasi dan pengujian selanjutnya.

1. Rancangan Penelitian

2. Berdasarkan gambar diatas penelitian ini mulai dengan melakukan studi litelatur yang berkaitan dengan analisis konteks dan LSTM serta kajian teori terkait deep learning, supervised learning, dan pengolahan bahasa alami. Selanjutnya adalah pengumpulan data yang dilakukan secara olen melalui platfrom simak unismuh. Setelah data terkumpul maka dilakukan pemrosesan dan pelabelan terhadap data yang diperoleh. Selanjutnya dilakukan tokenisasi, pelatihan, pengujian model menggunakan LSTM. Terakhir evaluasi dan analisis terhadap model serta dilakukan penarikan kesimpulan.

3. Pembangunan Sistem

4. Dalam penelitian ini, proses pembangunan sistem dimulai dengan memuat dataset yang akan digunakan diikuti dengan melakukan load model. Selanjutnya sistem akan memasuki tahap gate yang harus dilalui dalam model LSTM, dimulai dengan forget gate untuk menentukan informasi mana yang akan disimpan dan dibuang.

5. Kemudian input gate akan menerima informasi berupa hidden state dari cell sebelumnya, diikuti oleh ouput gate yang akan menentukan hidden state mana yang akan dikirim ke cell. Setelah itu dilakukan evaluasi terhadap akurasi pelatihan, jika akurasi pelatihan tinggi model akan dimasukan Kembali namun jika antuk akurasi pelatihan rendah maka jumlah epoch dapat ditambahkan utuk meningkatkan kinerja model. Sesuai yang digambarkan pada dibawah ini,

Langkah awal yang dilakukan dalam penggujian ini adalah mempersiapkan data yang telah dibersihkan dan dinormalkan untuk menghapus noise dan memastikan konsistensi format. Selanjutnya dataset diproses dengan melakukan tokenisasi teks dan menerapkan word embedding untuk menghasilkan representasi vector kata yang dibutuhkan algoritma LSTM.

Setelah data dipersiapkan, model algoritma LSTM dilatih menggunakan data pelatihan yang telah diproses. Dalam proses ini , parameter model dan hyperparameter disesuaikan untuk meningkatkan kinerja model. Setelah pelatihan selesai , model divalidasi dengan validasi silang untuk mengukur kinerja secara objektif.

Hasil pengujian kemudian dievaluasi dengan menggukur akurasi, presisi, recall, dan F1-score dari model. Pengujian ini membantu dalam memahami seberapa baik model dapat mengklasifikasikan sentiment dari data uji. Hasil pengujian dianalis untuk menarik kesimpulan tentang keefektifan algoritma LSTM dalam menganalisis kesesuaian kontek saran dan kritik.

Analisis data merupakan upaya untuk memastikan dan mengolah data hasil wawancara, observasi, dan sumber lainya, sehingga peneliti dapat memperoleh pemahaman yang mendalam mengenai kasus yang sedang diteliti dan menyajikannya untuk temuan yang akan datang (Ahmad & Muslimah, 2021). Berikut ini adalah teknik analisis data yaitu :

1. Reduksi data

2. Sebelum menerapkan model, langkah awal yang harus dilakukan adalah mereduksi data guna memastikan kualitas dan relevansi data yang digunakan. Reduksi data dapat melibatkan penghapusan duplikat, penghapusan data yang tidak relevan, normalisasi data, dan tokenisasi kata.

3. Penyajian data

4. Setelah melakukan reduksi data, Langkah selanjutnya adalah melakukan penyajian data. Data teks perlu diubah menjadi representasi numerik yang dapat dimengerti oleh model. Salah satu pendekatan umum yang digunakan adalah teknik Word2Vec atau Glove untuk mengonversi kata – kata menjadi vector numerik. Data juga perlu dibagi menjadi data latih dan data uji dengan proporsi yang sesuai.

5. Penarikan kesimpulan

6. Berdasarkan hasil pengujian, kesimpulan dapat ditarik mengenai kinerja model dalam analisis system kontek saran dan kritik. Kesimpulan ini mencangkup evaluasi terhadap akurasi dan konsistensi model dalam mengkalsifikasikan kontek saran dan kritik.

**3. Hasil dan diskusi**

Penelitian ini memanfaatkan data yang diporoleh dari sistem manajemen Unismuh yang mencangkup survei kepuasan mahasiswa dan dosen dari berbagai aspek, seperti fasilitas, layanan, dan kualitas pendidikan. Atribut yang terdapat dalam data ini mencangkup id, kategori, konteks saran dan kritik, serta tanggal inser.

Data ini akan digunakan sebagai acuan dalam melakukan analisis kesesuaan kontek saran dan kritik menggunakan algoritma LSTM.

Pada proses pelabelan ini, awalnya peneliti berencana mengambil tiga label yaitu saran, kritik, dan kategori gabungan bukan saran dan kritik. Namun, selama proses pelabelan, ditemukan bahwa jumlah entri data yang termaksud kategori bukan saran dan kritik jauh lebih banyak dibandingkan dengan entri yang mengandung saran dakn kritik. Ketidak seimbngan ini dapat menyebabkan bias dalam analisis dan mengurangi kualitas hasil yang diinginkan.

Untuk mengurangi ketimpangan data dan memastikan analisis lebih representative, peneliti memutuskan hanya focus pada entrain data yang mengandung saran dan kritik. Oleh karena itu , dari total 13.370 entri data aawal, diperlukan proses seleksi untuk mengumpulkan 10.000 entri data yang relevan.

Table 1 Data Labaling

Proses pelabelan melibatkan peninjauan manual yang cermat untuk memastikan bahwa setiap entri dapat ditempatkan dalam kategori yang tepat berdasarkan dengan konteks saran dan kritik yang diberikan responden. Setelah melalui tahap ini, data yang awalnya berjumlah 10.000 entri menyusut menjadi 7.754 entri setelah melakukan penghapusan pada data entri yang tidak mengandung saran dan kritik.

Preprocessing data merupakan langkah penting dalam pemrosesan teks untuk memastikan bahwa data yang akan digunakan dalam pelatihan model LSTM bersih, konsisten, dan siap digunakan. Proses ini melibatkan beberapa tahapan yang bertujuan untuk meningkankan kualitas data serta menggoptimalkan kinelja model. Berikut adalah langkah – langkah preprocessing yang dilakukan dalam penelitian ini:

1. Pembersihan data

2. Pembersihan data bertujuan untuk menghilangkan elemen – elemen yang tidak relevan dan mengganggu analisis teks.

import string

def remove\_punctuation(text):

return text.translate(str.maketrans('', '', string.punctuation))

df['ULASAN'] = df['ULASAN'].apply(remove\_punctuation)

Program ini berhasil menghapus semua tanda baca dari teks dalam kolom ulasan pada data frame dengan menggunakan fungsi ‘remove\_punctuation’. Setiap entri teks dalam kolom tersebut diproses untuk menghapus tanda baca seperti titik, koma, tanda seru, tanda Tanya, dan lainya.

1. Tokenisasi

2. Tokenisasi adalah proses pemecahan teks menjadi kata – kata individu atau token.

import nltk

nltk.download('punkt')

def tokenize\_text(text):

tokens = []

for sent in nltk.sent\_tokenize(text):

for word in nltk.word\_tokenize(sent):

if len(word) <= 0:

continue

tokens.append(word.lower())

return tokens

1. Program ini menggunakan library NLTK untuk melakukan proses tokenisasi teks, yang terdiri dari dua tahap, pertama pemecahan teks menjadi kalimat – kalimat menggunakan ‘nltk.sent\_tokenize’, dan kedua, memecahkan setiap kalimat menjadi kata – kata dengan ‘nltk.word\_tokenize’ setelahnya setiap kata diubah menjadi huruf kecil dan dimasukkan ke dalam daftar ‘tokens’.

max\_features = 500000

max\_sequence\_length = 50

tokenizer = Tokenizer(num\_words=max\_features, split=' ', filters='!"#$%&()\*+,-./:;<=>?@[\]^\_`{|}~', lower=True)

tokenizer.fit\_on\_texts(df['ULASAN'].values)

1. menetapkan parameter ‘max\_feature’ ditetapkan sebanyak 500.000, yang menunjukkan jumlah maksimum kata unik yang akan digunakan oleh tokenizer. Sementara itu ‘max\_sequence\_length’ ditetapkan sebesar 50, yang menentukan panjang maksimum dari setiap urutan teks yang akan diperoleh oleh model.

2. Selanjutnya, menginisialisasi sebuah objek tokenizer dengan beberapa pengaturan. Tokenizer ini akan menggunakan 500.000 kata yang paling sering muncul dalam korpus, dengan spasi sebagai pemisah antar kata. Selain itu karakter – karakter tertentu seperti tanda baca akan dihapus dari teks selama proses tokenisasi, dan semua teks akan diubah menjadi huruf kecil.

3. Dengan objek tokenizer yang telah diinisialisasi, peneliti kemudian melatih tokenizer ini menggunakan teks ulasan yang ada dalam dataset. Metode ‘fit\_on\_text’ digunakan untuk membuat indeks untuk setiap kata berdasarkan frequensi kemunculanya dalam korpus

X = tokenizer.texts\_to\_sequences(df['ULASAN'].values)

X = pad\_sequences(X, maxlen=max\_sequence\_length)

print('Shape dari data tensor:', X.shape)

1. Setelah tokenizer dilatih, dilanjutkan dengan mengubah teks ulasan menjadi urutan angka menggunakan metode ‘texts\_to\_sequences’. Metode ini mengonversi setiap kata dalam teng menjadi angka yang mewakili indeks kata tersebut dalam kosakata yang dihasilkan oleh tokenizer. Untuk memastikan bahwa semua urutan teks memiliki panjang yang sama maka peneliti menggunakan metode ‘pad\_sequences’. Metode ini menyesuaikan panjang setiap urutan teks menjadi 50 kata, sesuai dengan parameter ‘max\_Sequence\_length’ yang telah ditetapkan. Jika urutan teks lebih pendek dari 50 kata, maka akan ditambahkan dengan padding (nilai nol) diawal atau akhir urutan. Sebaliknya, jika lebih panjang, maka urutan akan dipotong.

Berdasarkan gambar 5 diatas menunjukan sebuah array yang berisi objek – objek ‘TeggedDocument’ yang terbagi menjadi dua bagian yaitu daftar kata dan tag. Seperti dokumen pertama mencatat sejumlah kata yang menggambarkan bagaimana seseorang dapat memahami keadaan para mahasiswa yang diberi tag saran.

1. Membangun model Doc2Vec

2. Model Doc2Vec merupakan teknik yang digunakan untuk menghasilkan representasi vektor yang terlatih untuk dolumen teks yang tidak hanya terdiri dari kata –kata individu namun juga dapat menangkap konteks dari keseluruhan dokumen.

vector\_size = 20

d2v\_model = Doc2Vec(dm=1, dm\_mean=1, vector\_size=vector\_size, window=8, min\_count=1, workers=1, alpha=0.065, min\_alpha=0.065)

d2v\_model.build\_vocab(train\_tagged)

1. Penenliti memulai dengan menetapkan berbagai parameter dan menganalisis model Doc2Vec. Model ini akan digunakan untuk menghasilkan vektor representative dari document ulasan yang dimiliki. Parameter yang diatur termaksud ukuran vektor sebesar 20, penggunaan algoritma distributor memori (dm = 1), dan dan laju pembelajaran awal (alpha) sebesar 0.065 disamping itu peneliti juga mengatur jumlah thread yang digunakan selama pelatihan menjadi satu.

2. Setelah parameter diatur, peneliti membangun kosakata dari data pelatihan yang sudah diberi tag menggunakan metode ‘build\_vocab’. Proses ini membuat model memahami kata – kata apa saja yang ada dalam korpus dan mempersiapkan model untuk pelatihan lebih lanjut.

for epoch in range(30):

d2v\_model.train(utils.shuffle(train\_tagged), total\_examples=len(train\_tagged), epochs=1)

d2v\_model.alpha -= 0.002

d2v\_model.min\_alpha = d2v\_model.alpha

1. Kemudian melatih model selama 30 epoch. Dimana setiap epoch, peneliti mengacak urutan dokumen pelatihan untuk memastikan model tidak overfit pada urutan tertentu. Model dilatih dengan semua dokumen dalam data pelatihan untuk setiap epoch. Setiap epoch laju, laju pembelajaran dikurangi sebesar 0.002, dan laju pembelajaran minimum disesuaikan dengan nilai alpha saat ini. Hal ini dilakukan untuk memastikan bahwa laju pembelajaran berkurang secara bertahap untuk membantu stabilitas pelatihan.

num\_words = len(d2v\_model.wv.key\_to\_index)

print("Jumlah kata dalam kosakata:", num\_words)

words\_in\_vocab = list(d2v\_model.wv.key\_to\_index.keys())

print("Kata-kata dalam kosakata:", words\_in\_vocab)

1. setelah proses pelatihan selesai, peneliti mendapatkan jumlah kata dalam kosakata model. Jumlah ini mencerminkan berapa banyak kata unik yang telah dipelajari oleh model dari data pelatihan. Peneliti juga mengumpulkan daftar kata – kata dalam kosakata dan menampilkannya. Informasi ini memberikan wawasan tentang dimensi dan konten kosakata yang digunakan oleh model untuk merepresentasikan dokumen – dokumen ulasan

2. Matrix Embedding

3. Matrix embedding adalah representasi numerik dari teks yang digunakan dalam NLP. Matriks ini memetakan kata – kata atau dokumen ke dalam vektor – vektor numerik dalam ruagan berdimensi tinggi.

embedding\_matrix = np.zeros((len(d2v\_model.dv.vectors), d2v\_model.vector\_size))

for i in range(len(d2v\_model.dv.vectors)):

embedding\_matrix[i] = d2v\_model.dv.vectors[i]

print("Ukuran matriks embedding:", embedding\_matrix.shape)

print("Contoh vektor untuk dokumen pertama:", embedding\_matrix[0])

1. program ini dimulai dengan inisialisasi matriks embedding kosong. Matriks ini dirancang untuk menampung vektor – vektor dokumen yang dihasilkan oleh model Doc2Vec yang telah dilatih sebelumnya. Ukuran matriks ini ditentukan oleh jumlah dokumen atau ukuran vektor yang digunakan dalam model. Misalnya, jika ada 2 dokumen dan ukuran vektornya 20 maka, Matriks embendding akan memiliki ukuran (2, 20).

2. Berikutnya, matriks embedding ini di isi dengan vektor – vektor dokumen dari model Doc2Vect. Untuk setiap dokumen dalam model, program mengambil vektor dokumen terdebut dan menempatkannya pada baris yang sesuai dalam matriks embedding. Proses ini dilakuka dalam loop yang berjalan dari indeks 0 hingga jumlah dokumen dalam model Doc2Vec.

3. Setelah matriks embendding terisis penuh, program menampilka informasi mengenai ukuran matriks dan memberikan contoh vektor untuk dokumen pertama.

from keras.models import Sequential

from keras.layers import LSTM, Dense, Embedding

Pustaka – pustaka yang diperlukan untuk diimport antaranya, ‘sequential’, ‘LSTM’, ‘Dense’, dan ‘Embedding’ dari library. Pustaka – pustaka ini digunakan untuk membangun model neural network.

MAX\_SEQUENCE\_LENGTH = 50

num\_unique\_words = len(tokenizer.word\_index) + 1

embedding\_matrix = np.random.rand(num\_unique\_words, 20)

Selanjutnya, beberapa parameter didefenisikan ‘MAX\_SEQUENCE\_LENGTH’ ditetakan dengan nilai 50, yang berarti panjang maksimum dari setiap urutan teks yang akan digunakan sebagai input ke dalam model. ‘num\_unique\_words’dihiting dari tokenizer untuk mendapatkan jumlah kata unik dalam dataset dan ‘embedding\_matrix’ diinisialisasi secara acak dengan ukuran yang sesuai dengan jumlah kata unik dan dimensi embedding sebesar 20.

model = Sequential()

Model kemudian diinisialisasi menggunakan objek ‘sequential’. Hal ini memungkinkan untuk membangun model secara berurutan dengan menambahkan layer – layer satu per satu.

model.add(Embedding(num\_unique\_words, 20, input\_length=MAX\_SEQUENCE\_LENGTH, weights=[embedding\_matrix], trainable=True))

Langkah berikutnya menambahkan layer embedding ke dalam model. Layer embending ini menggunakan jumlah kata unuk (‘num\_unique\_words’), dimana dimensi embendding sebesar 20, dan panjang input sebesar ‘MAX\_SEQUENCE\_LENGTH’. Bobot embedding layer di inisialisasikan dengan ‘embedding\_matrix’ yang telah dibuat sebelumnya. Dengan ‘trainable-True’, bobot embedding ini akan diperbarui selama proses pelatihan model.

model.add(LSTM(50, return\_sequences=False))

Setelah itu, layer LSTM ditambahkan ke dalam model dengan 50 unit. LSTM adalah jenis recurrent neural network yang mampu menangani data urutan dan mengingat informasi untuk jangka waktu yang lama. ‘return\_sequences=False’ menunjukkan bahwa hanya output dari langkah waktu terakhir yang akan digunakan, bukan seluruh urutan.

model.add(Dense(2, activation="softmax"))

Kemudian, layer fully connected (Dense) dengan 2 unit output ditambahkan ke dalam model, menggunakan fungsi aktivasi ‘softmax’. Fungsi aktivasi ini sering digunakan untuk tugas klasifikasi dua kelas, di mana hasilnya adalah probabilitas dari setiap kelas.

model.summary()

model.compile(optimizer="adam", loss="binary\_crossentropy", metrics=['acc'])

Model yang telah dibangun kemudian ditampilkan menggunakan ‘model.summary()’ yang memberikan ringkasan tentang struktur model, jumlah parameter, dan informasi lainya.

Kemudian model dikompilasikan dengan menggunakan optimizer ‘adam’ untuk optimisasi yang efisien untuk pembelajaran mesin, loss function ‘binary\_crossentropy’ yang umum digunakan untuk klasifikasi biner, dan matriks akurasi (‘acc’) yang akan digunakan untuk mengevaluasi kinerja model selama pelatihan.

def split\_input(sequence):

return sequence[:-1], sequence[1:]

sequence\_example = np.array([1, 2, 3, 4, 5])

x, y = split\_input(sequence\_example)

print("Input:", x)

print("Output:", y)

Terakhir fungsi ‘split\_input’digunakan untuk mengambil urutan dan membagi semua elemen kecuali yang terakhir sebagai inputan (‘x’), an semua elemen kecuali yang pertama sebagai output (‘y’). Contoh penggunaannya menunjukkan bagaimana fungsi ini bekerja dengan array contoh [1, 2, 3, 4, 5].

Berdasarkan gambar 7 diatas, model dengan nama ‘sequential\_2’ ini terdiri dari tiga lapisan utama dimana setiap lapisan memiliki fungsi dan karakter khusus masing – masing diantaranya.

1. Lapisan Embedding

2. Lapisan pertama ini bertanggung jawab untuk mengubah indeks kata dari input menjadi vektor berdimensi tetap. Setiap kata dalam urutan input dipetakan ke dalam vektor dengan panjang 20. Jika input memiliki panjang urutan 50, maka output dari lapisan ini akan berupa tensor dengan bentuk (batch\_size, 50, 20). Jumlah total parameter yang dilatih di lapisan ini adalah 107,260.

3. LSTM Layer

4. Lapisan kedua adlah LSTM yang dirancang untuk menangkap hubungan temporal dalam kata sequensial.LSTM menerima inputan dari lapisa embedding dan menghasilkan representasi dengan panjang urutan yang sama, yaitu 50, namun dengan dimensi yang berbeda, yakni 50 unit. Lapisan ini memiliki 14.200 parameter yang dilatih.

5. Dense Layer

6. Lapisan terakhir ini adalah lapisan full connected yang memproses output dari lapisan LSTM untuk menghasilkan klasifikasi akhir. Lapisan ini memetakan 50 unit dari LSTM ke dalam dua kelas output. Jumlah parameter yang dilatih dilapisan ini adalah 102

Dengan total 121,562 parameter yang dilatih, model ini siap untuk belajar dari data dan memberikan prediksi yang akurat.

1. Pembagian Data Training Dan Data Testing

2. Dalam proses ini, peneliti melakukan beberapa tahapan untuk mempersiapkan data ulasan yang akan digunakan dalam pelatihan odel dan pembelajaran mesin. Langkah – langkah tersebut meliputi pengubahan lebel menjadi format hot-one encoding dan pembagian dataset. Berikut adalah penjelasan lebih rinci terkait proses yang dilakukan.

Y = pd.get\_dummies(df['LABEL']).values

1. Langkah awal yang dilakukan adalah mengubah label kategori yang terdapat dalam kolom ‘LABEL’ pada data ulasan menjadi format One-Hot encoding. Format ini merupakan teknik representasi yang mengubah label kategori menjadi vektor biner. Pada vektor ini, setiap kategori diwakili oleh sebuah vektor dengan panjang yang sama dengan jumlah kategori, dimana hanya satu elemen yang bernilai 1 (yang menunjukan kategori tersebut) dan sisanya bernilai 0.

2. Missal, jika kita memiliki dua kategori yaitu saran dan kritik maka:

3. Saran akan diwakili dengan vector [1, 0]

4. Kritik akan diwakili dengan vector [0, 1]

5. from sklearn.model\_selection import train\_test\_split

6. X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(X, Y, test\_size=0.01, random\_state=42)

Setelah label diubah menjadi format yang sesuai, langkah berikutnya memisahkan data menjadi set pelatihan dan pengujian. Ini dilakukan menggunakan fungsi ‘training\_test\_split’ dari pustaka Sklearn. Dalam pembagian ini, kita menentukan bahwa 10% dari data akan digunakan sebagai data uji dan sisahnya akan digunakan sebagai data latih. Pembagian ini dilakukansecara acak namu dengan menetapkan ‘random\_state=42’ untuk memastikan bahwa pembagian data tetap konsisten setiap kali kode dijalankan.

print("Shape of X\_train:", X\_train.shape)

print("Shape of Y\_train:", Y\_train.shape)

print("Shape of X\_test:", X\_test.shape)

print("Shape of Y\_test:", Y\_test.shape)

Setelah data dibagi, program menampilkan bentuk dimensi dari set pelatihan dan pengujian untuk memasatikan bahwa pembagian telah dilakukan dengan benar. Bentuk data ini menampilkan jumlah sempel dan fitur dalam set pelatihan dan pengujian. Misalnya, hasil yang ditampilkan sebagai berikut

Model LSTM yang digunakan dalam penelitian ini terdiri dari beberapa lapisan, masing – masing dengan peran yang spesifik untuk memproses data teks dan menghasilkan prediksi yang akurat.

1. Pelatihan Model LSTM

2. Pada bagian ini, program melatih model LSTM yang telah dibangun menggunakan data pelatihan dan validasi , serta menampilkan hasil matrik evaluasi selama proses pelatihan.

history = model.fit(X\_train, Y\_train, epochs=20, batch\_size=batch\_size, verbose=2, validation\_data=(X\_test, Y\_test))

1. Model dilatih dengan memanggil fungsi ‘model.fit()’. Fungsi ini menerima beberapa parameter penting seperti data pelatihan (‘X\_train’ dan ‘Y\_train’), jumlah epoch (‘epochs=20’), ukuran batch (‘batch\_size’), tingkat verbosity (‘verbose=2’), dan data validasi (‘validation\_data=(X\_test, Y\_test)’). Pelatihan dilakukan selama 20 epoch, di mana pada setiap epoch model akan memproses seluruh data pelatihan dalam beberapa batch, kemudian dievaluasi dengan data validasi untuk memantau kinerja model secara berkala.

print(history.history.keys())

1. Setelah pelatihan selesai, hasilnya disimpan dengan objek ‘history’, yang merupakan directionari berisi matrik yang diacak selama pelatihan. Untuk melihat matrik yang aa kita dapat mencetak kunci dari dictionary.

val\_loss = history.history['val\_loss']

val\_acc = history.history['val\_acc']

print("Validation Loss:", val\_loss)

print("Validation Accuracy:", val\_acc)

1. Biasanya, metrik yang dilacak mencakup ‘loss’ (kerugian pelatihan), ‘val\_loss’ (kerugian validasi),’ acc’ (akurasi pelatihan), dan ‘val\_acc’ (akurasi validasi). Selanjutnya, nilai dari kerugian validasi dan akurasi validasi diekstraksi dari objek history dan ditampilkan. Hal ini memberikan wawasan tentang kinerja model pada data yang tidak terlihat selama pelatihan, membantu dalam mengevaluasi kemampuan generalisasi model.

2. Nilai ‘val\_loss’ menunjukan seberapa baik model memprediksi data validasi dalam hal kerugian. Sementara ‘val\_acc’ menunjukan presentase prediksi yang benar pada data validasi. Matrik – matrik ini penting untuk memastikan bahwa model tidak hanya belajar mengingat data pelatihan namun juga mampu menggeneralisasi ke data baru yang tidak terlihat selama proses pelatihan.

Pada awal pelatihan, di epoch pertama, model menunjukkan tingkat kesalahan (loss) sebesar 0.0472 dengan akurasi 98.60% pada data pelatihan. Namun, ketika diuji pada data validasi, tingkat kesalahan naik menjadi 0.3883, dan akurasi turun menjadi 90.88%. Ini adalah langkah awal yang menunjukkan model sedang mulai belajar pola dari data yang diberikan.

Memasuki epoch kedua, model menunjukkan peningkatan signifikan. Loss pada data pelatihan menurun menjadi 0.0278 dan akurasi meningkat menjadi 99.15%. Meskipun demikian, validation loss sedikit meningkat menjadi 0.3979, tetapi validation accuracy naik menjadi 92.23%, menunjukkan bahwa model mulai menggeneralisasi lebih baik.

Seiring berjalannya waktu, dari epoch ketiga hingga kelima, model terus memperbaiki diri dengan loss yang semakin menurun dan akurasi yang semakin tinggi pada data pelatihan. Validation loss dan validation accuracy berfluktuasi sedikit tetapi tetap dalam rentang yang menunjukkan performa yang baik.

Pada epoch keenam hingga kesepuluh, model mencapai stabilitas yang lebih tinggi. Pada epoch kedelapan, loss sangat rendah di 0.0184 dengan akurasi 99.42%, sementara validation loss mencapai 0.4709 dengan validation accuracy 91.96%. Ini menunjukkan model mampu mempertahankan performa yang baik bahkan pada data yang tidak terlihat selama pelatihan.

Selanjutnya, dari epoch kesebelas hingga kelima belas, model mempertahankan performa pelatihannya dengan loss yang terus menurun dan akurasi yang tetap tinggi. Pada epoch kelima belas, loss mencapai 0.0140 dan akurasi mencapai 99.64%, sementara validation loss dan validation accuracy menunjukkan hasil yang konsisten.

Menjelang akhir pelatihan, dari epoch keenam belas hingga dua puluh, model menunjukkan performa yang stabil. Pada epoch terakhir, model mencapai loss sangat rendah di 0.0106 dengan akurasi 99.58% pada data pelatihan. Meskipun validation loss berada di 0.5534, validation accuracy tetap tinggi di 91.29%, menunjukkan model telah belajar pola data dengan baik tanpa mengalami overfitting yang signifikan.

Secara keseluruhan, proses pelatihan ini menunjukkan bahwa model berhasil mempelajari data pelatihan dengan sangat baik, dengan performa yang memuaskan juga pada data validasi. Waktu pelatihan per epoch berkisar antara 5 hingga 7 detik, menunjukkan efisiensi dalam proses pelatihan.

1. Visualisasi Hasil Pelatihan

2. Setelah melatih model LSTM untuk mengklasifikasikan teks, langkah berikutnya adalah menganalisis kinerja model selama pelatihan dari objek ‘history’ yang berisi matrik kinerja pada setiap epoch pelatihan

history\_dict = history.history

1. Pertama, mengekstrak nilai –nilai penting dari history, termaksud nilai keraguan dan akurasi baik untuk data latih maupun validasi. Ekstraksi ini dilakukan untuk setiap epoch, memungkinkan untuk memvisualisasikan bagaimana kinerja model berkembang seiring waktu.

loss\_values = history\_dict['loss']

val\_loss\_values = history\_dict['val\_loss']

acc\_values = history\_dict['acc']

val\_acc\_values = history\_dict['val\_acc']

1. Kemudian membuat range untuk jumlah epoch, yang akan digunakan sebagai sumbu x pada plot. Ini memberikan konteks waktu untuk bagaimana kinerja matrik berubah dengan setiap iterasi pelatihan.

epochs = range(1, len(loss\_values) + 1)

1. Dengan data ini peneliti membuat dua plot terpisah dalam satu figure untuk visualisasi yang lebih baik. Plot pertama menunjukan nilai keraguan dari data latih dan validasi. Plot ini membantu melihat apakah model mampu mengurangi kesalahan prediksi selama pelatihan dan bagaimana hal ini dapat dibandingkan dengan kinerja pada data yang belum pernah dilihat sebelumnya

plt.figure(figsize=(12, 4))

plt.subplot(1, 2, 1)

plt.plot(epochs, loss\_values, 'bo', label='Training loss')

plt.plot(epochs, val\_loss\_values, 'b', label='Validation loss')

plt.title('Training and Validation Loss')

plt.xlabel('Epochs')

plt.ylabel('Loss')

plt.legend()

1. Plot kedua menunjukkan nilai akurasi pada data pelatihan dan validasi. Akurasi mengukur seberapa baik model kami mengklasifikasikan ulasan dengan benar. Melalui plot ini, kami dapat mengamati apakah model belajar secara efektif dari data pelatihan dan seberapa baik generalisasi model pada data validasi.

plt.subplot(1, 2, 2)

plt.plot(epochs, acc\_values, 'bo', label='Training accuracy')

plt.plot(epochs, val\_acc\_values, 'b', label='Validation accuracy')

plt.title('Training and Validation Accuracy')

plt.xlabel('Epochs')

plt.ylabel('Accuracy')

plt.legend()

1. Hasil visualisasi yang ditampilkan menggambarkan performa model selama proses pelatihandan validasi dalam hal loss dan akurasi seperti gambar dibawa ini.

2. Berdasarkan gambar 10 diatas menunjukan perubahan loss selama proses pelatihan dan validasi. Pada sumbu Y, dapat dilihat bahwa nilai loss yang mengidentifikasikan seberapa baik atau buruk model melakukan prediksi. Pada sumbu X, dapat dilihat jumlah epoch, yang merupakan iterasi pelatihan yang telah dilakukan

3. Titik biru mewakili nilai loss pada data latih untuk setiap epoch. Pada awal pelatihan, nilai loss sangat tinggi, namun seiring dengan bertambahnya epoch, nilai loss mengalami penurunan dengan signifikan. Hal ni menunjukan bahwa mode belajar dari data dan meningkatkan performanya.

4. Garis biru mewakili nilai loss pada data validasi untuk setiap epoch. Garis ini juga menunjukkan penurunan di awal. Tetapi menunjukan fluktuasi yang lebih dibandingkan training loss setelah sekitar epoch ke-5, setelah sekitar 10 epoch, validasi loss mulai stabil tetapi sedikit meningkat diakhir, yang mungkin mengidentifikasikan adanya overfitting

Pada gambar 11 menunjukan perubahan akurasi selama proses pelatihan dan validasi. Pada sumbu Y, dapat dilihat nilai akurasi yang mengidentifikasikan presentase prediksi yang benar oleh model. Pada sumbu X, untuk melihat jumlah epoch.

1. Titik biru mewakili nilai akurasi pada data pelatihan untuk setiap epoch. Pada awal pelatihan, nilai akurasi cukup rendah, tetapi meningkat dengan cepat dan stabil pada nilai yang tinggi setelah beberapa epoch. Ini menunjukkan bahwa model semakin baik dalam memprediksi dengan benar selama pelatihan berlangsung.

2. Garis biru mewakili nilai akurasi pada data validasi untuk setiap epoch., Setelah sekitar epoch ke-5, validation accuracy cenderung stabil di sekitar 95%, menunjukkan performa yang baik pada data yang tidak dilatih. Namun terdapat fluktuasi dalam validation accuracy setelah epoch ke-5, tetapi secara keseluruhan tetap berada pada nilai yang cukup tinggi.

Secara keseluruhan, hasil visualisasi ini menunjukkan bahwa model belajar dengan baik dari data pelatihan dan mampu melakukan generalisasi yang cukup baik terhadap data validasi. Penurunan training loss dan peningkatan training accuracy menunjukkan bahwa model semakin baik dalam mempelajari pola dari data pelatihan. Sementara itu, validation loss dan validation accuracy yang cukup stabil menunjukkan bahwa model memiliki kemampuan generalisasi yang baik, meskipun ada sedikit indikasi overfitting setelah epoch ke-10.

Model ini menunjukkan performa yang baik dan kemampuan generalisasi yang cukup kuat. Namun, fluktuasi yang terlihat pada validation loss dan accuracy menunjukkan bahwa ada ruang untuk lebih banyak fine-tuning atau penerapan teknik regularisasi untuk meningkatkan stabilitas dan performa model lebih lanjut.

Pada tahap ini model LSTM digunakan untuk mengklasifikasikan ulasan dari mahasiswa dan dosen terkait fasilitas, layanan, dan kualitas pendidikan di Unismuh terkait kategori saran dan kritik. Untuk memastikan bahwa model ini berfungsi dengan baik, beberapa matriks evaluasi yang digunakan.

1. Hasil Matrik Evaluasi

Model yang telah dilatih digunakan untuk melakukan prediksi pada data uji (‘x\_text’). Hasil prediksi ini berupa probabilitas yang menunjukan kemungkinan setiap ulasan termaksud dalam kategori tertentu. Probabilitas tersebut kemudian dikonversi menjadi label biner (0 dan 1) dengan menggunakan threshold 0.5. dengan kata lain, jika probabilitas lebih besar dari 0.5, maka ulasan tersebut diklasifikasikan sebagai kategori positif, jika sebaliknya diklasifikasikan sebagai kategori negatif.

predictions = model.predict(X\_test)

predicted\_labels = (predictions > 0.5).astype(int)

Setelah memperoleh prediksi, langkah selanjutnya adalah menghitung matriks evaluasi untuk menilai performa model. Matriks – matriks ini dihitung menggunakan fungsi – fungsi scikit-learn, yaitu ‘f1\_score’, ‘recision\_score’, dan ‘recall\_score’, yang semuanya menggunakan parameter ‘average='weighted’ untuk memberikan rata – rata pertimbangan berdasarkan jumlah setiap kelas.

f1 = f1\_score(true\_labels, predicted\_labels, average='weighted')

precision = precision\_score(true\_labels, predicted\_labels, average='weighted')

recall = recall\_score(true\_labels, predicted\_labels, average='weighted')

Hasil dari perhitungan matrik evaluasi kemudian ditampilkan. Selain itu, digunakan fungsi ‘classification\_report’ untuk memberikan ringkasan lebih rinci tentang performa model, termaksud precision, recall, F1 score, dan support untuk setiap kelas.

print(f"F1 Score: {f1}")

print(f"Precision: {precision}")

print(f"Recall: {recall}")

print(classification\_report(true\_labels, predicted\_labels))

Untuk memberikan gambaran lebih lanjut tentang hasil prediksi, program juga menampilkan array dari label asli dan label yang diprediksi.

print("Array hasil prediksi:")

print(true\_labels)

print(predicted\_labels)

Split Data 90:10

Table 2 Hasil Akurasi 90:10

1. Split Data 80:20

Table 3 Hasil Akurasi 80:20

1. Split Data 70:30

Table 4 Hasil Akurasi 70:30

1. Berdasarkan hasil evaluasi performa model dengan berbagai rasio perbandingan data prlatidan dan validasi berikut hal yang dapat dirangkum

2. Dengan meningkatnya rasio data validasi (dari 10% menjadi 30%), model menunjukkan peningkatan performa, terutama dalam hal precision dan recall untuk kedua kelas.

3. Dalam pembagian data 70:30, model mencapai nilai precision, recall, dan F1-Score yang tertinggi, menunjukkan bahwa model memiliki generalisasi yang lebih baik dengan data validasi yang lebih banyak.

4. Semua hasil evaluasi menunjukkan bahwa model memiliki performa yang sangat baik dalam melakukan klasifikasi dengan nilai precision, recall, dan F1-Score yang tinggi, menunjukkan bahwa model mampu melakukan prediksi dengan akurat dan sensitif terhadap kedua kelas.

5. Analisis Hasil Prediksi

6. Untuk lebih memahami performa model, beberapa contoh prediksi dibandingkan dengan label sebenarnya. Sebagian besar prediksi model sesuai dengan label sebenarnya, hal ini dapat menunjukan bahwa model dapat memahami konteks ulasan dengan baik. Namun, ada beberapa kasus dimana model membuat kesalahan prediksi seperti table dibawah ini.

Table 5 Hasil Prediksi

1. Ulasan “Hadirkan dosen dan siap dalam mengajar”

2. Meskipun model prediksi model berbeda dengan label sebenarnya, dalam konteks ini, prediksi dapat dianggap benar. Ulasan ini memberikan saran agar dosen selalu hadir dan siap dalam mengajar, yang lebih bersifat kontruktif dari pada kritik. Hal ini menunjukan model mampu menangkap esensi dari ulasan yang lebih positif dan memberikan rekomendasi. Jadi meskipun tidak sesuai dengan label sebenarnya, prediksi sebagai saran tetp relevan dan dapat diterima

3. Ulasan “Fasilitas pengolahan limbah kampus tidak memadai”

4. Pada ulasan ini, prediksi model tentu saja salah. Ulasan ini jelas merupakan kritik terhadap fasilitas pengolahan limbah yang dianggap tidak memadai. Model tidak berhasil menangkap konteks negative dari ulasan ini, sehingga kesalahan ini menunjukan bahwa adanya ruang untuk perbaikan dalam menangani ulasan dengan nada negative yang jelas

5. Ulasan “Sistem simak perlu ditingkatkan dikarenakan sistem sering mengalami error”

6. Ulasan ini sebenarnya dapat dikategorikan sebagai saran maupun kritik. Meskipun label sebenarnya adalah kritik, ulasan ini juga menyampaikan solusi, yaitu perlunya peningkatan sistem. Hal ini menunjukkan bahwa ulasan tersebut memiliki unsur saran yang signifikan, sehingga prediksi model sebagai saran tidak sepenuhnya salah. Ulasan ini mencerminkan ambiguitas yang sering ditemukan dalam teks ulasan, di mana sebuah pernyataan dapat memiliki dua interpretasi yang valid.

7. Hasil evaluasi secara keseluruhan menunjukkan bahwa model LSTM memiliki kemampuan yang baik dalam mengklasifikasikan ulasan. Dari 75 entri, hanya terdapat 3 entri yang tidak sesuai dengan label sebenarnya, menunjukkan tingkat kesalahan yang rendah. Kesalahan-kesalahan ini sebagian besar disebabkan oleh ambiguitas dalam teks ulasan dan kurangnya data pelatihan yang memadai untuk menangani semua variasi konteks ulasan.

8. Secara keseluruhan, model berhasil mengidentifikasi sebagian besar ulasan dengan benar, dan sebagian besar prediksi yang salah masih dapat dipertimbangkan sebagai interpretasi yang valid. Misalnya, dalam kasus ulasan pertama dan ketiga, prediksi yang berbeda dari label sebenarnya tetap relevan dan dapat diterima dalam konteks tertentu.

**4. Kesimpulan**

1. Bab ini berisikan simpulan dan saran-saran dari penelitan yang nantinya dapat digunakan untuk penelitian selanjutnya.

Penelitian ini menyatakan bahwa model LSTM menunjukkan performa yang baik dalam mengklasifikasikan konteks saran dan kritik, terutama ketika rasio data pelatihan dan validasi diatur dengan baik dan dapat disimpulkan bahwa, model menunjukan performa baik dalam berbagai skenario pembagian data. Namun dengan meningkatkan rasio data validasi dari 10% menjadi 30%, terdapat peningkatan performa model, terutama dalam hal precision dan recall untuk kedua kelas dengan tingkat akurasi precision 91%, recall 93%, F1-Score 91% untuk kelas 0 dan precision 96%, recall 96%, F1-Score 96% untuk kelas 1. Hal ini menunjukan bahwa model mampu menggenerelisasi dengan lebih baik dan memiliki performa optimal Ketika menggunakan lebih banyak data validasi. Disamping itu sebagian besar prediksi model sesuai dengan label sebenarnya, meskipun ada beberapa kesalahan prediksi yang disebabkan oleh ambiguitas dalam konteks saran dan kritik.

Berdasarkan hasil evaluasi dan analisis prediksi, saran perbaikan yang dapat diberikan adalah menggunakan data pelatihan dengan variasi konteks ulasan yang lebih banyak agar model dapat menangkap maksud ulasan dengan lebih tepat. Hal ini dilakukan untuk memastikan bahwa ulasan dengan makna ambigu dapat ditentukan secara objektif dan akurat. Selain itu, disarankan untuk mengeksplorasi penggunaan berbagai teknik yang dapat membantu mengurangi kesalahan prediksi dan menangani ambiguitas dalam teks ulasan.

**Referensi**

[1] Abijono, H., Santoso, P., & Anggreini, N. L. (2021). Algoritma Supervised Learning Dan Unsupervised Learning Dalam Pengolahan Data. Jurnal Teknologi Terapan: G-Tech, 4(2), 315–318. https://doi.org/10.33379/gtech.v4i2.635

[2] Ahmad, & Muslimah. (2021). Memahami Teknik Pengolahan dan Analisis Data Kualitatif. Proceedings, 1(1), 173–186.

[3] Alghifari, D. R., Edi, M., & Firmansyah, L. (2022). Implementasi Bidirectional LSTM untuk Analisis Sentimen Terhadap Layanan Grab Indonesia Bidirectional LSTM Implementation for Sentiment Analysis Against Grab Indonesia Services. 12, 89–99.

[4] Algoritma. (2022). Long Short Term Memory Network (LSTM). Algoritma. https://algorit.ma/blog/lstm-network-adalah-2022/

[5] Aljabar, A., & Karim, A. A. A. (2022). Analisis sentimen menggunakan algoritma lstm pada media sosial. 1(3).

[6] Andiyantama, M. Q., Zahira, I., & Irawan, A. (2021). Prediksi Energi Listrik Kincir Angin Berdasarkan Data Kecepatan Angin Menggunakan LSTM. JITCE (Journal of Information Technology and Computer Engineering), 5(01), 1–7. https://doi.org/10.25077/jitce.5.01.1-7.2021

[7] Fajar Ramadhan, D., Noertjahjono, S., & Dedy Irawan, J. (2020). PENERAPAN CHATBOT AUTO REPLY PADA WHATSAPP SEBAGAI PUSAT INFORMASI PRAKTIKUM MENGGUNAKAN ARTIFICIAL INTELLIGENCE MARKUP LANGUAGE. In Jurnal Mahasiswa Teknik Informatika (Vol. 4, Issue 1).

[8] Giarsyani, N. (2020). Komparasi Algoritma Machine Learning dan Deep Learning untuk Named Entity Recognition : Studi Kasus Data Kebencanaan. Indonesian Journal of Applied Informatics, 4(2), 138. https://doi.org/10.20961/ijai.v4i2.41317

[9] Hanifa, A., Fauzan, S. A., Hikal, M., & Ashfiya, M. B. (2021). Perbandingan Metode LSTM dan GRU (RNN) untuk Klasifikasi Berita Palsu Berbahasa Indonesia. Dinamika Rekayasa, 17(1), 33. https://doi.org/10.20884/1.dr.2021.17.1.436

[10] Husamuddin, H., Prasetyo, D. B., & Rustamadji, H. C. (2020). Otomatisasi Layanan Frequently Ask Questions Berbasis Natural Langugae Processing Pada Telegram Bot. Telematika, 17(2), 145. https://doi.org/10.31315/telematika.v1i1.3383

[11] Isnain, A. R., Sulistiani, H., Hurohman, B. M., & Nurkholis, A. (2022). Analisis Perbandingan Algoritma LSTM dan Naive Bayes untuk Analisis Sentimen. 8(2), 299–303.

[12] Mestika, J. C., Selan, M. O., & Qadafi, M. I. (2022). Menjelajahi Teknik-Teknik Supervised Learning untuk Pemodelan Prediktif Menggunakan Python. BIIKMA : Buletin Ilmiah Ilmu Komputer Dan Multimedia, 99(99), 216–219.

[13] Monalisa, S., Anggara, P. P., & Kurnia, F. (2018). Analisis Kesuksesan Penerapan Sistem Administrasi Akademik Menggunakan Human Organization Technology Fit Model. Jurnal Ilmiah Rekayasa Dan Manajemen Sistem Informasi, 4(1), 36–41.

[14] Oktriwina, A. S. (2021). NLP: Kecerdasan Buatan yang Bantu Komputer Pahami Bahasa Manusia. Glints. https://glints.com/id/lowongan/natural-language-processing-adalah/

[15] Oracle. (n.d.). What is natural language processing? Oracle. https://www.oracle.com/id/artificial-intelligence/what-is-natural-language-processing/

[16] Putri, H. M., Fuadi, W., Informatika, T., Teknik, F., Malikussaleh, U., & Holistic, M. (n.d.). PENDETEKSIAN BAHASA ISYARAT INDONESIA SECARA REAL-TIME MENGGUNAKAN LONG.

[17] Rachman, F. P., & Santoso, H. (2021). Perbandingan Model Deep Learning untuk Klasifikasi Sentiment Analysis dengan Teknik Natural Languange Processing. Jurnal Teknologi Dan Manajemen Informatika, 7(2), 103–112.

[18] Rahman, M. Z., Sari, Y. A., & Yudistira, N. (2021). Analisis Sentimen Tweet COVID-19 menggunakan Word Embedding dan Metode Long Short-Term Memory (LSTM). Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 5(11), 5120–5127. http://j-ptiik.ub.ac.id

[19] Riani, E., Yonathan, J., & Oliver, L. (2021). Audit Sistem Informasi Akademik (SIMAK) Menggunakan Framework COBIT 5 di Universitas Universal, Journal of Digital Ecosystem for Natural Sustainability (JoDENS). Journal of Digital Ecosystem for Natural Sustainability (JoDENS), 1(2), 2798–6179.

[20] Rizki, M., Basuki, S., & Azhar, Y. (2020). Implementasi Deep Learning Menggunakan Arsitektur Long Short Term Memory(LSTM) Untuk Prediksi Curah Hujan Kota Malang. Jurnal Repositor, 2(3), 331–338. https://doi.org/10.22219/repositor.v2i3.470

[21] Setiawan, A. Y., Darmawiguna, I. G. M., & Pradnyana, G. A. (2022). Sentiment Summarization Evaluasi Pembelajaran Menggunakan Algoritma LSTM (long short term memory). Kumpulan Artikel Mahasiswa Pendidikan Teknik Informatika (KARMAPATI), 11(2), 183–191.

[22] Talita, A. S., & Wiguna, A. (2019). IMPLEMENTASI ALGORITMA LONG SHORT-TERM MEMORY ( LSTM ) UNTUK MENDETEKSI UJARAN KEBENCIAN ( HATE SPEECH ) PADA KASUS PILPRES 2019. 19(1), 37–44.

[23] Wiranda, L., & Sadikin, M. (2019). Penerapan Long Short Term Memory pada Data Time Series untuk Memprediksi Penjualan Produk PT. Metiska Farma. Jurnal Nasional Pendidikan Teknik Informatika (JANAPATI), 8(3), 184–196.