**PENERAPAN WORD EMBEDDING FASTTEXT DALAM ANALISIS SENTIMEN REVIEW APLIKASI JAKI**

## **ARYO DININGRAT SALEA**

UNIVERSITAS MUHAMMADIYAH MAKASSAR

e-mail: xxxx@xxxx.xxx\*

***Abstract***

*ARYO DININGRAT SALEA, Application of Word Embedding Fasttext in Sentiment Analysis Review of the Jaki Application Using the CNN Method (supervised by Muhydin A M Hayat, S.Kom., MT and Fachrim Irhamna Rachman, S.Kom., M.T).*

*This research aims to test the level of accuracy of the performance of the FastText engineering model as Word Embedding against the CNN method in processing text data, especially in JAKI application reviews. This study also wants to know the extent of the influence of the CNN method in analyzing sentiment. The dataset used was 4,455 data, but attributes such as reviewId, userName, userImage, score, reviewCreatedVersion, replyContent, and appVersion were removed. After attribute removal, the remaining data was 3,199 reviews. The data preprocessing process involves the removal of punctuation and tokenization. This dataset of 3,199 reviews was tested with three sentiment categories, namely positive, negative, and neutral. The test results show that the combination of Word Embedding FastText – CNN is effective in analyzing sentiment. Models that use this combination achieve high accuracy values ​​between 91% and 100%, and show consistently lower and stable loss values.*

*Keywords: Sentiment Analysis, Jaki Application, Convolutional Neural Network, Word Embedding Fasttext.*

**Abstrak**

ARYO DININGRAT SALEA, Penerapan Word Embedding Fasttext Dalam Analisis Sentimen Review Aplikasi Jaki Menggunakan Metode CNN (dibimbing oleh Muhydin A M Hayat, S.Kom.,MT dan Fachrim Irhamna Rachman, S.Kom., M.T).

Penelitian ini bertujuan untuk menguji tingkat akurasi kinerja model teknik FastText sebagai Word Embedding terhadap metode CNN dalam memproses data teks, khususnya pada ulasan aplikasi JAKI. Penelitian ini juga ingin mengetahui sejauh mana pengaruh metode CNN dalam menganalisis sentimen. Dataset yang digunakan berjumlah 4.455 data, namun dilakukan penghapusan atribut seperti reviewId, userName, userImage, score, reviewCreatedVersion, replyContent, dan appVersion. Setelah penghapusan atribut, data yang tersisa sebanyak 3.199 ulasan. Proses preprocessing data dilakukan dengan melibatkan penghapusan tanda baca (punctuation) dan tokenisasi (tokenizing). Dataset sebanyak 3.199 ulasan ini diuji dengan tiga kategori sentimen yaitu positif, negatif, dan netral. Hasil pengujian menunjukkan bahwa kombinasi “Word Embedding FastText – CNN” efektif dalam menganalisis sentimen. Model yang menggunakan kombinasi ini mencapai nilai akurasi yang tinggi antara 91% hingga 100%, serta menunjukkan nilai loss yang lebih rendah dan stabil secara konsisten.

Kata Kunci: Analisis Sentimen, Aplikasi Jaki, Convolutional Neural Network, Word Embedding Fasttext.

**1. Pendahuluan**

Analisis sentimen muncul sebagai alat penting untuk memahami dan menginterpretasi perasaan atau sikap yang digunakan dalam tulisan. Dalam era digital saat ini, volume data tekstual yang dihasilkan oleh pengguna melalui media sosial, ulasan produk, blog dan platform lainnya terus meningkat.

Analisis sentimen telah menarik perhatian dalam beberapa dekade terakhir dalam penelitian di bidang Natural Language Processing. Analisis sentimen menjadi proses untuk mengidentifikasi atau mengkategorikan opini pengguna dalam bentuk teks terhadap berbagai hal seperti film, produk, acara, dan lainnya, baik itu positif, negatif, maupun netral. Analisis sentimen sangat berguna untuk mengidentifikasi, mengekstraksi, dan mempelajari informasi subjektif tentang produk suatu perusahaan. Terkadang, perusahaan membutuhkan wawasan yang lebih rinci mengenai sentimen terhadap produk mereka, seperti aspek apa saja dari produk yang perlu dievaluasi. (Muhammad Afif Raihan & Erwin Budi Setiawan, 2022)

Pemerintah provinsi DKI Jakarta melalui Badan Layanan Umum Daerah (BLUD) Jakarta Smart City, telah meluncurkan sebuah inovasi dalam bentuk aplikasi super yang dikenal dengan sebutan JAKI (Jakarta Kini). Aplikasi JAKI ini berfungsi sebagai pusat resmi layanan informasi pemerintah Provinsi DKI Jakarta yang dirancang guna memenuhi kebutuhan harian masyarakat. (Putu Sawitra Danda Prasetia et al., 2024). Aplikasi JAKI memiliki rating 3,1-3,8 pada skala 1 hingga 5 pada platform AppStore dan PlayStore. Meskipun aplikasi ini mendapatkan sejumlah ulasan dan peringkat yang cukup baik, namun masih ada beberapa pengguna yang memberikan ulasan buruk. (Rodhi et al., 2022). Maka dengan penulisan skripsi ini diharapkan mampu menjadi rujukan evaluasi terhadap sikap pengguna aplikasi JAKI.

Sekitar tahun 2000, teknik Word Embedding mulai dikembangkan dimana Word Embedding itu sendiri memetakan setiap kata dalam dokumen kedalam vektor padat, dimana setiap vektor mewakili proyeksi kata dalam dimensi vektor. Metode Word Embedding ini menjadi sangat penting dalam proses analisis sentimen karena menyediakan representasi kata yang lebih baik. Word Embedding terbagi atas beberapa metode diantaranya Word2Vec, Glove dan FastText. Salah satu kelebihan dari Word Embedding adalah tidak memerlukan anotasi/label dan dapat dihasilkan langsung dari korpus yang tidak teranotasi sehingga waktu pelatihan data dapat lebih cepat, (Nurdin et al., 2020). Convolutional Neural Network termasuk salah satu jenis jaringan saraf yang ditemukan dalam deep learning. CNN dikenal sebagai Feed Forward Neural Network yang dirancang khusus untuk menangani data yang dipecah menjadi bagian-bagian terpisah. (Yuniarossy et al., 2024).

Penelitian ini diharapkan mampu menerapkan dan menguji efektivitas kombinasi antara Word Embedding FastText dan CNN dalam analisis sentimen review aplikasi JAKI. Maka dengan dilakukannya pendekatan tersebut dapat meningkatkan accuracy klasifikasi sentimen dan memberikan dasar yang kuat kepada pengembang untuk perbaikan layanan aplikasi.

Sehubungan dengan latar belakang yang telah dijelaskan diatas, rumusan masalah yang diangkat dalam penelitian ini adalah terbagi atas dua poin yaitu :

1. Menguji tingkat akurasi kinerja model kombinasi antara Word Embedding FastText dan CNN dalam memproses data teks khususnya pada ulasan aplikasi JAKI.

2. Bagaimana pengaruh metode CNN dalam memproses dan mengkategorikan analisis sentimen.

3. Adapun tujuan dari penelitian ini ialah untuk menguji tingkat keakuratan teknik FastText sebagai Word Embedding terhadap accuracy model CNN dalam memproses data teks, khususnya pada ulasan aplikasi JAKI.

4. Seberapa baik pengaruh CNN dalam berkontribusi meningkatkan kinerja model pada proses mengkategorikan sentimen ulasan.

Diharapkan penelitian ini mampu memberikan kontribusi manfaat baik secara teoritis maupun praktis adalah :

1. Secara Teoritis

2. Untuk pengembangan ilmu pengetahuan terutama dalam bidang teknik informatika.

3. Memberikan wawasan dan literatur ilmiah dalam bidang analisis sentimen dan pengolahan bahasa alami, khususnya pada penggunaan Word Embedding.

4. Secara Praktis

5. Bagi Peneliti

6. Salah satu persyaratan yang harus dipenuhi untuk menyelesaikan program S1.

7. Mendapatkan pengalaman praktis dalam memahami penggunaan teknik Word Embedding dan pengembangan model CNN terhadap analisis sentimen.

8. Bagi Universitas

9. Sebagai bahan referensi untuk penelitian yang akan dilakukan di masa mendatang.

10. Sebagai bahan evaluasi bagi universitas dalam mengembangkan keilmuan, dalam hal ini yang berkaitan dengan penerapan Word Embedding dan pengembangan CNN dalam memproses data teks dan pengolahan bahasa alami.

11. Studi kasus pada penelitian ini terbatas pada analisis sentimen terkait ulasan pengguna aplikasi JAKI.

12. Dataset yang digunakan hanya diperoleh dari ulasan pada platform Google PlayStore.

13. Penerapan Word Embedding terbatas pada penggunaan model FastText saja tanpa mempertimbangkan model lain.

14. Terbatas hanya menggunakan metode Convolutional Neural Network (CNN).

Secara garis besar penulisan laporan tugas akhir ini terbagi menjadi beberapa bab yaitu :

**2. Metode Penelitian**

1. Tempat Penelitian

Penelitian ini dilakukan secara online dengan mengambil beberapa ulasan aplikasi JAKI pada platform Google PlayStore.

1. Waktu Penelitian

Adapun pelaksanaan penelitian ini dilakukan mulai pada Juni 2024 dan akan berlangsung hingga seluruh proses pengumpulan data selesai.

1. Kebutuhan Hardware

2. Laptop Asus VivoBook X415JAB

3. Kebutuhan Software

4. Google Colaboratory

5. Google Drive

6. Excel

7. Python

8. Scikit-Learn dan TensorFlow

Perancangan sistem adalah aspek penting dalam pengembangan suatu sistem karena menjelaskan proses dari tahap perencanaan hingga implementasi fungsi-fungsi yang diperlukan untuk operasional. Tujuan utama perancangan sistem adalah memastikan bahwa sistem yang akan dibangun mampu mencapai hasil yang diharapkan.

Flowchart atau diagram alur adalah representasi grafis yang menampilkan urutan langkah-langkah dan keputusan yang diperlukan untuk menjalankan suatu proses dalam suatu program. Setiap langkah direpresentasikan dalam bentuk diagram dan dihubungkan oleh garis atau panah untuk menunjukkan arah alur proses.

Diagram tersebut menggambarkan proses dalam mengevaluasi penggunaan Word Embedding dengan FastText untuk meningkatkan kinerja model CNN dalam analisis sentimen terhadap ulasan aplikasi JAKI. Inisiasi proyek dan pengumpulan data dibatasi hanya ulasan aplikasi JAKI pada platform Google PlayStore menggunakan teknik scraping. Setelah data terkumpul, selanjutnya adalah pelabelan data, di mana ulasan diberikan label sentimen positif atau negatif sebagai target untuk pelatihan model diikuti oleh proses pra-pemrosesan seperti pembersihan, tokenisasi, dan normalisasi teks. Berikutnya ialah mengubah teks ulasan menjadi representasi vektor numerik menggunakan teknik Word Embedding dengan FastText, yang menangkap makna semantik dari setiap kata. Setelah mendapatkan embedding dari FastText, representasi vektor kata ini digunakan sebagai input untuk model yang dibangun menggunakan framework seperti TensorFlow.

Dengan data yang telah di-embed, desain arsitektur model CNN kemudian dilatih menggunakan data latih yang telah melalui proses embedding. Setelah pelatihan selesai, model dievaluasi untuk menilai kinerjanya menggunakan metrik seperti accuracy, precision, recall dan F1-score. Hasil evaluasi kemudian dianalisis untuk memahami seberapa baik model bekerja dalam mengklasifikasi sentimen ulasan. Setelah proses pelatihan dan evaluasi, setiap langkah penelitian di dokumentasikan dengan baik, diikuti oleh penyusunan laporan skripsi yang mencakup latar belakang, metodologi, hasil, dan kesimpulan.

Dalam perancangan sistem atau diagram sistem yang akan dibuat yaitu sebagai berikut :

Dalam diagram di atas, alur kerja dimulai dengan persiapan data, pengklasifikasian, dan pemuatan data untuk pelatihan model. Beberapa lapisan menerapkan fungsi aktivasi ReLU, diikuti oleh lapisan output yang menggunakan fungsi aktivasi Softmax untuk menghasilkan probabilitas kelas. Selama pelatihan, accuracy data pelatihan dievaluasi secara berkala, dan jika accuracy terus meningkat, proses dilanjutkan ke langkah berikutnya. Model dimasukkan ke dalam sistem untuk proses pelatihan yang melibatkan iterasi (epoch) guna memperbaiki kinerja model. Setelah pelatihan selesai, proses pun berakhir.

Sistem yang dirancang memiliki dua alur utama: perancangan sistem pelatihan dan perancangan sistem pengujian. Pada perancangan sistem pelatihan, fokus utamanya adalah memantau kurva accuracy untuk mengevaluasi kinerja model selama proses pelatihan. Sementara itu, perancangan sistem pengujian difokuskan pada tahap pengujian, dengan tujuan utama untuk menguji kemampuan sistem dalam mengidentifikasi dan menguji aspek-aspek yang ditargetkan.

Flowchart di atas menjelaskan proses pelatihan model FastText untuk pemrosesan teks. Proses dimulai dengan inisialisasi model FastText, diikuti oleh pembentukan kosakata dari dokumen yang telah dikonversi menjadi daftar kata. Kemudian, model diinisialisasi dengan epoch sebanyak 30. Selanjutnya, model memeriksa apakah epoch kurang dari 30. Jika ya, data pelatihan diacak dan model dilatih, serta nilai alpha dan min\_alpha disesuaikan. Jika tidak, model melanjutkan untuk menghitung jumlah kata dalam kosakata dan mengakses kata-kata tersebut. Setelah itu, dilakukan inisialisasi matriks embedding, diikuti dengan pengisian dan pencetakan matriks embedding. Proses berakhir setelah seluruh langkah ini selesai.

Teknik pengujian sistem yang akan digunakan dalam pengujian ini melibatkan pemisahan data menjadi dua bagian, yaitu data training dan data testing. Dengan demikian, teknik pengujian ini memungkinkan evaluasi yang akurat terhadap kinerja model dalam mengklasifikasikan sentimen pada teks ulasan aplikasi JAKI.

Penelitian ini menggunakan teknik pengujian Confusion Matrix untuk mendapatkan pemahaman yang lebih baik tentang bagaimana model berfungsi untuk menganalisis sentimen dengan tiga variabel yaitu, positif, negatif, dan netral. Confusion Matrix memungkinkan analisis menyeluruh terhadap kemampuan model untuk mengklasifikasikan setiap sentimen.

Pengujian accuracy bertujuan untuk mengukur keberhasilan model dalam mengklasifikasikan sentimen dengan tepat. Accuracy model dihitung dengan membandingkan hasil klasifikasi sentimen dari model dengan label sentimen yang sebenarnya pada data pengujian, menggunakan persamaan tertentu untuk menghitung proporsi prediksi yang benar dari keseluruhan data uji.

Di bawah ini adalah rumus perhitungan Confusion Matrix untuk menghitung precision, recall, dan nilai accuracy, (Dinata et al., 2020) :

1. Accuracy

2. Berfungsi untuk mengukur sejauh mana kinerja model dalam mengkalsifikasikan kelas sentimen dalam dataset

3. (1)

4. Recall

5. Berdasarkan persamaan berikut, berfungsi untuk mengukur tingkat keberhasilan sistem dalam menemukan kembali data

6. (2)

7. Precision

8. Berfungsi untuk mengevaluasi ketepatan hubungan antara informasi yang diminta pengguna dan jawaban yang diberikan sistem.

9. (3)

10. Keterangan :

11. TP : True Positif

12. TN : True Negatif

13. FP : False Positif

14. FN : False Negatif

Analisis data adalah upaya mencari dan menata hasil observasi, wawancara, dan hasil lainnya secara sistematis untuk meningkatkan pemahaman peneliti tentang kasus yang diteliti serta menyajikannya sebagai temuan bagi orang lain. Ini juga mencakup proses mengurutkan dan mengorganisir data ke dalam pola, kategori, dan uraian dasar sejak awal pengumpulan data di lapangan untuk memastikan data terkumpul secara lengkap dan terstruktur, (Nurdewi, 2022). Untuk mencapai hasil yang diinginkan maka peneliti melakukan beberapa tahapan analisis sebagai berikut :

Reduksi Data (Data Reduction)

Reduksi data adalah proses menyederhanakan, memilah, dan mengubah data kasar menjadi bentuk yang lebih mudah dikelola dan dipahami. Dalam tahap ini, data yang relevan dipilih dan dirangkum sementara data yang tidak relevan dibuang. Proses ini melibatkan kegiatan seperti coding, pengelompokan, dan pembuatan tema atau kategori. Tujuannya adalah untuk mengidentifikasi pola dan tema utama yang muncul dari data sehingga memudahkan analisis lebih lanjut.

Penyajian Data (Display Data)

Penyajian data adalah proses menata data yang telah direduksi dalam bentuk yang terorganisir dan mudah dipahami. Data dapat disajikan dalam berbagai bentuk seperti teks naratif, tabel, grafik, atau diagram. Penyajian data bertujuan untuk membantu peneliti memahami informasi yang kompleks dengan lebih baik dan untuk mendukung proses penarikan kesimpulan. Ini juga memudahkan dalam mengkomunikasikan temuan kepada audiens lain.

Penarikan Kesimpulan (Conclucing Drawing Verification)

Penarikan kesimpulan adalah tahap di mana peneliti mengevaluasi data yang telah direduksi dan disajikan untuk membuat interpretasi dan kesimpulan yang valid. Proses ini melibatkan mencari hubungan, pola, dan makna dalam data serta mengaitkan temuan dengan pertanyaan penelitian atau hipotesis yang telah ditetapkan. Kesimpulan yang ditarik harus didasarkan pada bukti yang kuat dan dapat diandalkan dari data yang telah dianalisis.

**3. Hasil dan diskusi**

1. Data ulasan diambil menggunakan teknik scraping pada python, yang mengumpulkan data dari platform Google PlayStore pada Aplikasi JAKI. Dengan jumlah total data yang berhasil dikumpulkan ialah sebanyak 4.455 ulasan. Berikut proses pengambilan data menggunakan teknik scraping pada pyhton :

2. from google\_play\_scraper import Sort, reviews\_all

3. import pandas as pd

4. # Mengumpulkan semua ulasan untuk aplikasi Jakarta Kini

5. scrapreview = reviews\_all(

6. 'id.go.jakarta.smartcity.jaki',

7. sleep\_milliseconds=0,  # Anda dapat menyesuaikan jeda waktu jika diperlukan

8. lang='id',

9. country='id',

10. sort=Sort.MOST\_RELEVANT

11. )

12. # Menampilkan hasil scrapping

13. print(scrapreview)

14. # Membuat DataFrame dari hasil scrapping

15. app\_reviews\_df = pd.DataFrame(scrapreview)

16. # Menyimpan DataFrame ke dalam file Excel

17. app\_reviews\_df.to\_excel('C:/Users/asus/Downloads/DataRating.xlsx', index=False, header=True)

Kode ini dimulai dengan mengimpor dua library, yaitu `google\_play\_scraper` dan `pandas`. Fungsi `reviews\_all` digunakan untuk mengumpulkan semua ulasan aplikasi "Jakarta Kini" dari Google PlayStore. Parameter `sleep\_milliseconds`, `lang`, `country`, dan `sort` menentukan pengaturan untuk pengambilan data seperti tidak ada jeda waktu, bahasa Indonesia, ulasan dari Indonesia, dan pengurutan berdasarkan relevansi. Hasil ulasan ini disimpan dalam variabel `scrapreview` dan ditampilkan menggunakan `print`. Kemudian, data ulasan tersebut diubah menjadi sebuah DataFrame menggunakan `pandas.DataFrame`. Terakhir, DataFrame tersebut disimpan ke dalam file Excel bernama `DataRating.xlsx` di direktori yang ditentukan, tanpa menyertakan indeks dan dengan menyertakan nama kolom. Pesan konfirmasi dicetak untuk menunjukkan bahwa data telah berhasil disimpan.

Setelah mengumpulkan data ulasan dari Google PlayStore untuk aplikasi JAKI menggunakan Scraping Data, hasilnya disimpan dalam format excel. Berikut adalah hasil pengambilan dataset ulasan yang kemudian di simpan dalam format excel

Setelah dataset ulasan pada aplikasi JAKI telah diperoleh dengan mendaptkan total keseluruhan data sebanyak 4.455 ulasan, selanjutnya dilakukan penghapusan atribut seperti reviewId, userName, userIamage, score, reviewCreatedVersion, replyContent, appVersion dan hanya menyisahkan “content” saja yang nantinya akan dijadikan sebagai atribut ulasan. Setelah itu, dataset diurutkan berdasarkan abjad A-Z untuk memudahkan dalam proses pembersihan data yang memiliki kemiripan ulasan serta menghapus ulasan-ulasan yang hanya menggunakan emoji tanpa teks, yang kemudian menyisahkan total data secara keseluruhan ialah sebanyak 3.199 data ulasan.

Pelabelan sentimen dilakukan secara manual untuk menemukan pola dan karakteristik dalam teks yang menunjukkan sentimen positif, negatif, maupun netral. Ulasan yang dikumpulkan berasal dari platform Google PlayStore pada Aplikasi JAKI yang kemudian disimpan dalam atribut ulasan, sedangkan nilai klasifikasi seperti positif, negatif, atau netral disimpan dalam atribut label. Di bawah ini adalah tabel pelabelan data:

1. Pra-pemrosesan data, juga dikenal sebagai preprocessing data, adalah proses yang dilakukan untuk mempersiapkan data mentah agar dapat digunakan oleh algoritma pengajaran mesin atau analisis data. Tujuan dari pra-pemrosesan data adalah untuk membersihkan, mengubah, dan menyusun data sehingga lebih mudah dianalisis dan diolah. Langkah-langkah pra-pemrosesan data biasanya mencakup beberapa atau semua dari langkah-langkah yang disebutkan di atas, tergantung pada jenis data dan tujuan analisis.

2. Punctuation

3. Punctuation atau dikenal sebagai tanda baca adalah simbol yang digunakan dalam tulisan untuk membedakan bagian-bagian yang berbeda dari teks, seperti titik (.), koma (,), tanda tanya (?), tanda seru (!), dan lainnya. Penanganan tanda baca (punctuation) membantu membersihkan dan menormalisasi data dalam pra-pemrosesan teks untuk mempersiapkan data sebelum digunakan dalam analisis atau model machine learning.

4. import string

5. def remove\_punctuation(text):

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

7. # Menghapus tanda baca dari kolom ULASAN

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

Ini bertujuan untuk menghapus tanda baca dari teks dalam kolom `ULASAN` pada DataFrame `df`, fungsi `remove\_punctuation(text)` di definisikan untuk menghilangkan tanda baca dari sebuah teks dengan menggunakan metode `translate()` dan modul `string.punctuation`, yang berisi semua karakter tanda baca. Fungsi `str.maketrans('', '', string.punctuation)` membuat tabel translasi yang akan menggantikan semua tanda baca dengan string kosong, sehingga menghapusnya dari teks. `df['ULASAN'] = df['ULASAN'].apply(remove\_punctuation)` menerapkan fungsi `remove\_punctuation` pada setiap elemen dalam kolom `ULASAN`, memproses setiap teks untuk menghilangkan semua tanda baca yang ada.

1. Tokenizing

Tokenizing adalah proses pemrosesan teks di mana teks dibagi menjadi bagian kecil yang disebut "token", yang biasanya terdiri dari kata-kata, frasa, atau bahkan karakter tergantung pada tujuan analisis. Tokenisasi adalah langkah penting dalam pra-pemrosesan data teks karena memungkinkan model pengajaran mesin atau algoritma lainnya untuk mengubah teks yang panjang atau kompleks menjadi bentuk yang lebih sederhana yang lebih mudah dianalisis.

1. import nltk

2. # Download the 'punkt' resource

3. nltk.download('punkt')

4. # Tokenisasi teks menggunakan nltk

5. def tokenize\_text(text):

6. tokens = []

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

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

9. if len(word) <= 0:

10. continue

11. tokens.append(word.lower())

12. return token

13. # Memisahkan data menjadi train dan test

14. train, test = train\_test\_split(df, test\_size=0.1, random\_state=42)

15. Kode ini menggunakan pustaka Natural Language Toolkit (nltk) untuk melakukan tokenisasi teks dan memisahkan data menjadi set train dan test. `nltk.download('punkt')` digunakan untuk mengunduh modul punkt, yang berisi algoritma pemisahan teks menjadi kalimat dan kata. Fungsi `tokenize\_text(text)` kemudian didefinisikan untuk mengubah teks menjadi token (kata-kata yang terpisah). Di dalam fungsi ini, `nltk.sent\_tokenize(text)` memisahkan teks menjadi kalimat, dan `nltk.word\_tokenize(sent)` memisahkan setiap kalimat menjadi kata. Kata-kata yang tidak kosong diubah menjadi huruf kecil dan disimpan dalam daftar `tokens`. Setelah itu, `train, test = train\_test\_split(df, test\_size=0.1, random\_state=42)` membagi DataFrame `df` menjadi dua subset: `train` dan `test`, dengan ukuran 10% dari data digunakan sebagai data test. Parameter `random\_state=42` memastikan pemisahan yang konsisten untuk reproduktibilitas.

16. df = pd.read\_excel('/content/drive/Othercomputers/My Laptop/Documents/SKRIPSI/Aryo - CNN - FastText/DATAJAKI.xlsx',sheet\_name="Sheet1")  # Replace 'path\_to\_your\_excel\_file.xlsx' with your actual file path

17. df = df[['ULASAN', 'LABEL']]  # Selecting relevant columns

18. df = df[pd.notnull(df['ULASAN'])]  # Dropping rows with null 'ULASAN' values

19. df.rename(columns={'ULASAN': 'ULASAN'}, inplace=True)  # Rename 'ULASAN' to 'ULASAN' for consistency

20. df.head()

21. df.shape

Dimulai dengan membaca file Excel yang berlokasi di direktori tertentu menggunakan fungsi `pd.read\_excel`. File Excel tersebut diakses dari jalur yang ditentukan, dan data diambil dari sheet bernama "Sheet1". Setelah data dimuat ke dalam DataFrame `df`, kode ini memilih kolom yang relevan, yaitu kolom "ULASAN" dan "LABEL", dan menyimpannya kembali di dalam `df`. Langkah berikutnya adalah menghapus baris-baris yang memiliki nilai kosong (null) pada kolom "ULASAN" dengan menggunakan fungsi `pd.notnull`. Setelah itu, kode ini mengganti nama kolom "ULASAN" menjadi "ULASAN" dengan `df.rename` untuk menjaga konsistensi penamaan kolom, meskipun dalam hal ini tidak terjadi perubahan karena nama awal dan akhir sama. Akhirnya, kode ini menampilkan beberapa baris pertama dari DataFrame `df` dengan menggunakan `df.head()`, yang berguna untuk melihat pratinjau data yang telah dibersihkan dan dipilih, serta `df.shape` yang memberikan informasi tentang dimensi dari DataFrame berisi jumlah baris dan jumlah kolom.

1. # TaggedDocument untuk train dan test set

2. train\_tagged = train.apply(

3. lambda r: TaggedDocument(words=tokenize\_text(r['ULASAN']), tags=[r.LABEL]), axis=1)

4. test\_tagged = test.apply(

5. lambda r: TaggedDocument(words=tokenize\_text(r['ULASAN']), tags=[r.LABEL]), axis=1)

6. # Pengaturan tokenizer

7. max\_features = 500000  # Jumlah maksimum kata yang akan digunakan

8. max\_sequence\_length = 50  # Panjang maksimum setiap teks

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

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

11. # Konversi teks ke dalam urutan angka (sequences)

12. X\_train = tokenizer.texts\_to\_sequences(train['ULASAN'].values)

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

14. X\_test = tokenizer.texts\_to\_sequences(test['ULASAN'].values)

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

16. print('Found %s unique tokens.' % len(tokenizer.word\_index))

17. Kode ini menyiapkan data teks untuk pemodelan dengan langkah-langkah sistematis. Kode membuat instance `TaggedDocument` untuk dataset pelatihan dan pengujian, di mana setiap `TaggedDocument` berisi kata-kata yang telah ditokenisasi dari kolom 'ULASAN' serta tag yang sesuai dengan label ulasan. Ini mempersiapkan data untuk model yang memerlukan format teks tertentu, seperti model dalam pustaka Gensim. Selanjutnya, sebuah `Tokenizer` diinisialisasi dengan batas kosakata maksimum 500.000 kata dan diatur untuk memproses teks dengan memisahkan kata berdasarkan spasi dan mengabaikan tanda baca.

18. Tokenizer ini juga mengonversi semua teks menjadi huruf kecil untuk menjaga konsistensi. Tokenizer kemudian dilatih menggunakan data teks dari kolom 'ULASAN' dalam DataFrame `df`, sehingga mempelajari kosakata yang ada. Setelah itu, teks dari dataset pelatihan dan pengujian dikonversi menjadi urutan angka, di mana setiap angka mewakili kata dalam kosakata tokenizer, dan urutan ini dipadatkan menjadi panjang yang seragam sebanyak 50 token untuk memastikan konsistensi ukuran input bagi model. Terakhir, kode mencetak jumlah token unik yang ditemukan dalam kosakata tokenizer, memberikan gambaran mengenai ukuran kosakata data teks. Langkah-langkah ini secara efektif menyiapkan data teks dengan mengonversinya ke format yang sesuai untuk model pembelajaran mesin.

19. from gensim.models import FastText

20. from gensim.models.FastText import TaggedDocument

21. # Ubah ukuran vektor (vector\_size) sesuai kebutuhan Anda

22. vector\_size = 500

23. # Inisialisasi model FastText

24. FastText\_model = FastText(vector\_size=vector\_size, window=8, min\_count=1, workers=1, sg=1, alpha=0.065, min\_alpha=0.065)

25. # Membangun kosakata dari tagged documents pada data pelatihan

26. train\_tagged = [TaggedDocument(words=tokenize\_text(row['ULASAN']), tags=[row['LABEL']]) for index, row in train.iterrows()]

27. # Ubah train\_tagged menjadi list kata-kata

28. train\_corpus = [doc.words for doc in train\_tagged]

29. # Bangun vocab dari train\_corpus

30. FastText\_model.build\_vocab(train\_corpus)

31. Kode ini mengatur dan membangun model FastText menggunakan pustaka Gensim. Ukuran vektor (`vector\_size`) diatur menjadi 500 untuk model FastText. Model kemudian diinisialisasi dengan berbagai parameter, termasuk ukuran jendela konteks (`window`), jumlah minimum kemunculan kata (`min\_count`), jumlah thread (`workers`), tipe model (`sg` untuk Skip-gram), serta tingkat pembelajaran awal (`alpha`) dan minimum (`min\_alpha`). Setelah inisialisasi, kosakata dibangun dari data pelatihan. Ini dilakukan dengan membuat daftar `TaggedDocument` dari data pelatihan, di mana setiap `TaggedDocument` berisi kata-kata yang telah ditokenisasi dari kolom 'ULASAN' dan tag yang sesuai dengan label. Daftar kata-kata ini kemudian disiapkan sebagai `train\_corpus`, dan kosakata dibangun untuk model FastText menggunakan `build\_vocab` berdasarkan `train\_corpus`.

32. from sklearn.utils import shuffle

33. # Latih model

34. for epoch in range(30):

35. FastText\_model.train(shuffle(train\_corpus), total\_examples=len(train\_corpus), epochs=1)

36. FastText\_model.alpha -= 0.002  # Reduksi alpha setiap epoch

37. FastText\_model.min\_alpha = FastText\_model.alpha  #Tetapkan min\_alpha sesuai alpha saat ini

38. print(FastText\_model)

Kode ini melatih model FastText selama 30 epoch dengan data pelatihan. Proses pelatihan dilakukan dengan mengacak urutan data pelatihan menggunakan fungsi `shuffle` dari `sklearn.utils`, untuk memastikan bahwa data yang digunakan dalam setiap epoch bervariasi. Model dilatih dengan data yang sudah diacak, dan parameter `total\_examples` diatur sesuai dengan jumlah contoh dalam `train\_corpus`. Setelah setiap epoch, tingkat pembelajaran awal (`alpha`) dikurangi sebesar 0.002, dan nilai `min\_alpha` diperbarui agar sesuai dengan nilai `alpha` saat ini. Ini membantu mengurangi laju pembelajaran seiring berjalannya pelatihan. Setelah semua epoch selesai, model FastText yang telah dilatih dicetak untuk menampilkan hasil akhir dari pelatihan.

1. # Mendapatkan jumlah kata dalam kosakata

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

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

4. # Mengakses kata-kata dalam kosakata

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

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

7. Kode ini bertujuan untuk memperoleh dan menampilkan informasi tentang kosakata yang dibangun oleh model FastText. Pertama, jumlah kata dalam kosakata dihitung dengan menggunakan `len(FastText\_model.wv.key\_to\_index)`, yang mengembalikan jumlah total kata yang ada dalam kosakata model. Hasilnya dicetak untuk menunjukkan berapa banyak kata yang telah dipelajari oleh model. Selanjutnya, kode mengakses semua kata dalam kosakata dengan mengambil kunci dari kamus `key\_to\_index` dan mengubahnya menjadi daftar menggunakan `list()`. Daftar kata-kata ini kemudian dicetak untuk menampilkan semua kata yang ada dalam kosakata model.

8. # Inisialisasi matriks embedding kosong

9. embedding\_matrix = np.zeros((len(FastText\_model.wv.key\_to\_index), FastText\_model.vector\_size))

10. # Mengisi matriks embedding dengan vektor-vektor kata dari model FastText

11. for i, word in enumerate(FastText\_model.wv.index\_to\_key):

12. embedding\_vector = FastText\_model.wv.get\_vector(word)

13. if embedding\_vector is not None:

14. embedding\_matrix[i] = embedding\_vector

15. # Contoh penggunaan matriks embedding

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

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

18. Kode ini menginisialisasi dan mengisi matriks embedding berdasarkan vektor kata dari model FastText. Sebuah matriks embedding kosong dengan ukuran yang sesuai dibuat menggunakan `np.zeros`, dimana jumlah barisnya adalah jumlah kata dalam kosakata model (`len(FastText\_model.wv.key\_to\_index)`) dan jumlah kolomnya adalah ukuran vektor embedding (`FastText\_model.vector\_size`). Kemudian, kode mengisi matriks embedding dengan vektor-vektor kata dari model FastText. Untuk setiap kata dalam kosakata model (`FastText\_model.wv.index\_to\_key`), vektor kata diambil menggunakan `FastText\_model.wv.get\_vector(word)`. Jika vektor kata tidak kosong, vektor tersebut dimasukkan ke dalam baris yang sesuai dalam matriks embedding.

19. from keras.models import Sequential

20. from keras.layers import Conv1D, GlobalMaxPooling1D, Dense, Embedding, Dropout

21. # Definisikan panjang maksimum urutan

22. MAX\_SEQUENCE\_LENGTH = 50

23. # Definisikan jumlah kata unik

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

25. # Pastikan bahwa embedding\_matrix memiliki bentuk yang sesuai

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

27. # Inisialisasi model Sequential

28. model = Sequential()

29. # Menambahkan lapisan Embedding dengan bobot yang sesuai

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

31. # Menambahkan lapisan Conv1D

32. model.add(Conv1D(50, 3, activation='relu'))

33. Kode ini mendefinisikan dan menginisialisasi model jaringan saraf konvolusional (CNN) untuk pemrosesan teks menggunakan pustaka Keras. Panjang maksimum urutan (`MAX\_SEQUENCE\_LENGTH`) diatur ke 50, yang menentukan panjang setiap input teks. Jumlah kata unik dalam kosakata ditentukan dengan menambahkan 1 pada jumlah kata yang ada dalam `tokenizer.word\_index`.

34. Selanjutnya, sebuah matriks embedding diinisialisasi dengan ukuran `num\_unique\_words` x 20, di mana 20 adalah dimensi dari vektor embedding. Ini adalah matriks acak untuk sementara waktu, yang seharusnya digantikan dengan matriks embedding yang sebenarnya.

35. Model Sequential diinisialisasi, dan lapisan `Embedding` ditambahkan sebagai lapisan pertama. Lapisan ini menggunakan `embedding\_matrix` yang telah diinisialisasi sebelumnya untuk memetakan indeks kata menjadi vektor berdimensi 20, dengan panjang input yang ditentukan oleh `MAX\_SEQUENCE\_LENGTH`. Lapisan ini dapat dilatih (`trainable=True`). Lapisan `Conv1D` ditambahkan dengan 50 filter dan ukuran jendela 3, menggunakan fungsi aktivasi ReLU. Lapisan ini akan mengaplikasikan konvolusi pada input teks untuk menangkap fitur spasial dalam data.

36. model.add(Dropout(0.25))

37. # Menambahkan lapisan GlobalMaxPooling1D

38. model.add(GlobalMaxPooling1D())

39. # Menambahkan lapisan Dense untuk output

40. model.add(Dense(3, activation="softmax"))

41. # Menampilkan ringkasan model

42. model.summary()

43. # Kompilasi model

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

45. # Contoh pemanggilan fungsi split\_input

46. def split\_input(sequence):

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

48. # Contoh penggunaan split\_input

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

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

51. print("Input:", x)

52. print("Output:", y)

53. Kode ini menyelesaikan definisi dan konfigurasi model jaringan saraf konvolusional (CNN) untuk pemrosesan teks serta menyertakan contoh fungsi untuk manipulasi data. Lapisan `Dropout` dengan rasio 0.25 ditambahkan setelah lapisan konvolusi untuk mengurangi risiko overfitting dengan menonaktifkan 25% neuron secara acak selama pelatihan. Selanjutnya, lapisan `GlobalMaxPooling1D` ditambahkan untuk merangkum fitur penting dari output lapisan konvolusi dengan mengambil nilai maksimum sepanjang dimensi temporal. Lapisan `Dense` dengan 3 unit dan fungsi aktivasi `softmax` ditambahkan sebagai lapisan output untuk klasifikasi, di mana 3 unit mewakili jumlah kelas dan `softmax` mengonversi output menjadi probabilitas kelas. Model dirangkum dengan `model.summary()` untuk memberikan ringkasan struktur model. Model kemudian dikompilasi menggunakan optimizer Adam dan fungsi kehilangan `categorical\_crossentropy`, dengan akurasi sebagai metrik evaluasi. Terakhir, kode menyediakan contoh fungsi `split\_input`, yang membagi urutan data menjadi pasangan input dan output, misalnya mengubah urutan `[1, 2, 3, 4, 5]` menjadi input `[1, 2, 3, 4]` dan output `[2, 3, 4, 5]`, yang berguna untuk pelatihan model berbasis sekuens.

54. Embedding Layer : Mengubah kata menjadi vektor dengan dimensi (50, 20), yang berarti panjang urutan 50 dan embedding berukuran 20.

55. Conv1D : Menggunakan filter konvolusi untuk menangkap pola dalam data sekuensial, menghasilkan output (48, 50).

56. Dropout : Mencegah overfitting dengan secara acak mengabaikan beberapa unit selama pelatihan.

57. Global Max Pooling 1D : Mengambil nilai maksimum dari tiap filter, mengurangi dimensi menjadi (50).

58. Dense Layer : Lapisan akhir untuk klasifikasi dengan 3 kelas.

Model ini menggunakan total 144,043 parameter untuk mendeteksi fitur penting dari teks dan mengklasifikasikannya. Input `[1,2,3,4]` dan Output `[2,3,4,5]` ini mungkin adalah contoh tokenized data yang menunjukkan pergeseran satu posisi, sering kali digunakan dalam pengaturan pemrosesan sekuens.

1. from keras.callbacks import EarlyStopping, ModelCheckpoint

2. # Membuat callback EarlyStopping dan ModelCheckpoint

3. model\_checkpoint = ModelCheckpoint('/content/drive/Othercomputers/My Laptop/Documents/SKRIPSI/Aryo - CNN - FastText/CNN\_FastText.h5', monitor='val\_acc', save\_best\_only=True, verbose=2)

4. # Pelatihan model dengan callback

5. history = model.fit(X\_train, Y\_train,

6. epochs=50,

7. batch\_size=32,

8. validation\_data=(X\_test, Y\_test),

9. callbacks=[model\_checkpoint],

10. verbose=2)

11. # Mendapatkan histori pelatihan

12. print(history.history.keys())

13. # Menampilkan val\_loss dan val\_accuracy

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

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

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

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

18. Kode ini digunakan untuk melatih model dengan dua callback untuk memantau dan menyimpan model selama proses pelatihan. Pertama, sebuah callback `ModelCheckpoint` diinisialisasi untuk menyimpan model yang memiliki kinerja terbaik pada data validasi dengan nama file yang ditentukan. Parameter `monitor` diatur untuk memantau akurasi validasi (`val\_acc`), `save\_best\_only=True` memastikan hanya model terbaik yang disimpan, dan `verbose=2` memberikan informasi detail selama pelatihan. Model kemudian dilatih menggunakan `model.fit`, dengan data pelatihan dan data validasi yang telah ditentukan, berlangsung selama 50 epoch dengan ukuran batch 32. Callback `ModelCheckpoint` diterapkan untuk menyimpan model terbaik. Setelah pelatihan selesai, riwayat pelatihan diperiksa untuk metrik yang dilacak, dan nilai `val\_loss` dan `val\_acc` dicetak untuk menunjukkan kerugian dan akurasi model pada data validasi.

19. # Mendapatkan histori pelatihan

20. history\_dict = history.history

21. # Ekstrak nilai untuk setiap metrik

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

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

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

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

26. # Buat range untuk jumlah epoch

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

28. # Plot Loss

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

30. plt.subplot(1, 2, 1)

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

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

33. plt.title('Training and Validation Loss')

34. plt.xlabel('Epochs')

35. plt.ylabel('Loss')

36. plt.legend()

37. # Plot Accuracy

38. plt.subplot(1, 2, 2)

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

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

41. plt.title('Training and Validation Accuracy')

42. plt.xlabel('Epochs')

43. plt.ylabel('Accuracy')

44. plt.legend()

45. Kode ini digunakan untuk memvisualisasikan hasil pelatihan model dengan grafik kerugian (loss) dan akurasi. Pertama, riwayat pelatihan diambil dari objek `history` dan disimpan dalam `history\_dict`, yang menyimpan nilai-nilai untuk setiap metrik pelatihan seperti `loss`, `val\_loss`, `acc`, dan `val\_acc`. Nilai-nilai ini diekstrak untuk digunakan dalam plot. Rentang epoch dibuat berdasarkan jumlah epoch pelatihan. Dengan menggunakan `matplotlib`, dua subplot dalam satu gambar dibuat: subplot pertama menunjukkan grafik kerugian pelatihan dan validasi terhadap epoch, dengan kerugian pelatihan ditampilkan sebagai titik biru dan kerugian validasi sebagai garis biru; subplot kedua menunjukkan grafik akurasi pelatihan dan validasi terhadap epoch, dengan akurasi pelatihan sebagai titik biru dan akurasi validasi sebagai garis biru. Grafik ini memberikan gambaran visual tentang bagaimana kerugian dan akurasi model berubah selama proses pelatihan.

46. from sklearn.metrics import classification\_report, f1\_score, precision\_score, recall\_score

47. # Melakukan prediksi

48. predictions = model.predict(X\_test)

49. predicted\_labels = (predictions > 0.5).astype(int)  # Konversi probabilitas menjadi label biner (0 atau 1)

50. true\_labels = Y\_test

51. # Menghitung metrik evaluasi tambahan

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

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

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

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

56. print(f"Precision: {precision}")

57. print(f"Recall: {recall}")

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

59. # Tampilkan hasil prediksi dalam array

60. print("Array hasil prediksi:")

61. print(true\_labels)

62. print(predicted\_labels)

63. Kode ini digunakan untuk mengevaluasi performa model klasifikasi menggunakan beberapa metrik evaluasi, seperti F1 Score, Precision, dan Recall, serta menampilkan laporan klasifikasi terperinci. Pertama, prediksi dilakukan pada data pengujian (`X\_test`) menggunakan model terlatih, menghasilkan probabilitas untuk setiap sampel. Probabilitas ini dikonversi menjadi label biner (0 atau 1) dengan menerapkan threshold 0.5. Proses ini dilakukan dengan memeriksa apakah nilai probabilitas lebih besar dari 0.5, dan hasilnya disimpan dalam `predicted\_labels`. Label sebenarnya (`true\_labels`) diambil dari `Y\_test`.

64. Setelah itu, metrik evaluasi dihitung untuk menilai kinerja model. F1 Score dihitung menggunakan `f1\_score()` dari `sklearn.metrics`, dengan `average='weighted'` yang memperhitungkan ketidakseimbangan antar kelas. Precision dan Recall dihitung menggunakan `precision\_score()` dan `recall\_score()` dengan cara yang sama. Hasil dari setiap metrik dicetak untuk memberikan wawasan tentang kualitas prediksi model.

65. Selanjutnya, laporan klasifikasi lengkap dicetak menggunakan `classification\_report()`, yang menyediakan metrik evaluasi untuk setiap kelas dalam data, termasuk precision, recall, F1 Score, dan support (jumlah sampel per kelas). Terakhir, hasil prediksi dalam bentuk array dicetak, yang memungkinkan pengguna untuk membandingkan label sebenarnya (`true\_labels`) dengan label yang diprediksi (`predicted\_labels`). Ini membantu dalam memahami performa model dan mengidentifikasi area yang memerlukan perbaikan.

66. print("Panjang Tes Ulasan:", len(test['ULASAN']))

67. print("Panjang X\_test:", len(X\_test))

68. print("Panjang Y\_test:", len(Y\_test))

69. print("Panjang true\_labels:", len(true\_labels))

70. print("Panjang predicted\_labels:", len(predicted\_labels))

Kode ini digunakan untuk memverifikasi konsistensi ukuran data pengujian dalam proses evaluasi model. Pertama, panjang ulasan teks dalam dataset pengujian asli dicetak menggunakan `print("Panjang Tes Ulasan:", len(test['ULASAN']))`, memberikan gambaran tentang jumlah sampel dalam data pengujian. Selanjutnya, panjang array `X\_test`, yang berisi fitur-fitur ulasan setelah melalui pra-pemrosesan seperti tokenisasi dan padding, dicetak dengan `print("Panjang X\_test:", len(X\_test))`. Kemudian, panjang `Y\_test`, yang berisi label asli untuk setiap ulasan dalam `X\_test`, dicetak menggunakan `print("Panjang Y\_test:", len(Y\_test))`.

Setelah itu, panjang `true\_labels`, yang diambil langsung dari `Y\_test`, dicetak dengan `print("Panjang true\_labels:", len(true\_labels))`. Terakhir, panjang `predicted\_labels`, yang dihasilkan setelah model melakukan prediksi pada `X\_test`, dicetak menggunakan `print("Panjang predicted\_labels:", len(predicted\_labels))`. Pernyataan-pernyataan ini memastikan bahwa panjang `X\_test`, `Y\_test`, `true\_labels`, dan `predicted\_labels` sesuai satu sama lain, yang penting untuk memastikan integritas dan konsistensi data sebelum melakukan evaluasi atau analisis lebih lanjut terhadap hasil prediksi model.

1. import pandas as pd

2. from sklearn.metrics import classification\_report,f1\_score, precision\_score, recall\_score

3. # Melakukan prediksi dengan softmax (misalnya, jika menggunakan TensorFlow/Keras)

4. predictions = model.predict(X\_test)

5. predicted\_labels = predictions.argmax(axis=1)  # Mengambil kelas dengan probabilitas tertinggi sebagai prediksi

6. # Pastikan true\_labels adalah dalam bentuk indeks kelas yang sama dengan predicted\_labels

7. true\_labels = Y\_test.argmax(axis=1)  # Jika Y\_test adalah dalam bentuk one-hot encoded, konversi ke indeks kelas

8. # Menghitung metrik evaluasi tambahan

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

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

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

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

13. print(f"Precision: {precision}")

14. print(f"Recall: {recall}")

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

16. # Ambil ulasan, label sebenarnya, dan prediksi label

17. # Pastikan panjang semua array sama

18. min\_length = min(len(test['ULASAN']), len(true\_labels), len(predicted\_labels))

19. test\_results = pd.DataFrame({

20. 'Ulasan': test['ULASAN'].values[:min\_length],

21. #Gunakan min\_length untuk memastikan panjang yang sama

22. 'Label Sebenarnya': true\_labels[:min\_length],

23. 'Prediksi': predicted\_labels[:min\_length]

24. })

25. # Klasifikasi label 'Negatif', 'Positif', dan 'Netral' berdasarkan nilai

26. def classify\_label(label):

27. if label == 0:

28. return 'Negatif'

29. elif label == 1:

30. return 'Netral'

31. else:

32. return 'Positif'

33. # Menambahkan kolom klasifikasi label

34. test\_results['Label Sebenarnya'] = test\_results['Label Sebenarnya'].apply(classify\_label)

35. test\_results['Prediksi'] = test\_results['Prediksi'].apply(classify\_label)

36. # Export ke Excel

37. test\_results.to\_excel('/content/drive/Othercomputers/My Laptop/Documents/SKRIPSI/Aryo - CNN - FastText/HASIL PREDIKSI.xlsx', index=False)

38. # Tampilkan hasil

39. print("\nHasil Prediksi:\n", test\_results)

40. print("Data berhasil diekspor ke 'hasil\_prediksi2.xlsx'.")

41. Kode ini bertujuan untuk mengevaluasi kinerja model klasifikasi yang telah dilatih dan mengekspor hasil prediksi ke dalam file Excel. Pada awalnya, prediksi dibuat dengan model menggunakan data pengujian `X\_test`, yang memberikan probabilitas untuk setiap kelas. Fungsi `argmax(axis=1)` digunakan pada output prediksi ini untuk menentukan indeks kelas dengan probabilitas tertinggi dan menyimpannya dalam variabel `predicted\_labels`. Langkah ini mengkonversi probabilitas softmax menjadi label kelas yang dapat digunakan untuk perbandingan langsung dengan label sebenarnya. Label sebenarnya (`true\_labels`), yang diambil dari `Y\_test`, juga dikonversi dari bentuk one-hot encoding ke indeks kelas yang sesuai menggunakan `argmax(axis=1)`, sehingga keduanya memiliki format yang sama untuk evaluasi.

42. Selanjutnya, kode ini menghitung tiga metrik evaluasi: F1 Score, Precision, dan Recall. Ketiga metrik ini dihitung dengan metode `f1\_score`, `precision\_score`, dan `recall\_score` dari `sklearn.metrics`, masing-masing dengan opsi `average='weighted'`, yang digunakan untuk menangani kemungkinan ketidakseimbangan antar kelas dalam data. Nilai-nilai ini dicetak untuk memberikan wawasan tentang performa model, dan laporan klasifikasi yang lebih rinci dihasilkan dengan menggunakan `classification\_report()`, yang memberikan metrik untuk setiap kelas, termasuk precision, recall, F1 Score, dan support.

43. Kemudian, kode ini membangun DataFrame `test\_results` yang menyimpan informasi ulasan dari `test['ULASAN']`, serta label sebenarnya dan prediksi label. Untuk memastikan bahwa panjang dari semua elemen yang digunakan dalam DataFrame konsisten, `min\_length` dihitung sebagai nilai minimum dari panjang `test['ULASAN']`, `true\_labels`, dan `predicted\_labels`. Hal ini untuk memastikan bahwa jumlah data yang diambil sama pada semua variabel tersebut dan mencegah adanya kesalahan data.

44. Selain itu, fungsi `classify\_label` di definisikan untuk mengubah indeks kelas menjadi label teks yang lebih mudah dipahami, seperti 'Negatif', 'Netral', dan 'Positif'. Fungsi ini diterapkan pada kolom 'Label Sebenarnya' dan 'Prediksi' dari DataFrame `test\_results` untuk mengganti indeks numerik dengan label yang lebih intuitif. Setelah DataFrame diisi dengan data yang diformat ulang ini, kode menggunakan metode `to\_excel()` untuk mengekspor `test\_results` ke file Excel dengan nama file yang ditentukan, yaitu `'HASIL PREDIKSI.xlsx'`. Akhirnya, hasil prediksi dicetak ke konsol, dan pesan sukses ditampilkan untuk menginformasikan bahwa data telah berhasil diekspor ke file Excel. Kode ini tidak hanya memfasilitasi evaluasi performa model tetapi juga memudahkan penyimpanan dan interpretasi hasil prediksi klasifikasi, sehingga pengguna dapat menganalisis data dengan lebih efektif dan menyajikan hasil dalam format yang dapat diakses dan mudah dibaca.

45. Pengujian dan Hasil Metode

46. Pengujian data terdiri dari 3.199 ulasan dari aplikasi JAKI (Jakarta Kini). Data latih dan uji dibagi dengan perbandingan 90:10, 80:20, dan 70:30, untuk menunjukkan bahwa data latih akan digunakan untuk melatih model dan data uji akan digunakan untuk menguji kinerja model yang dilatih. Berikut tampilan hasil akurasi terhadap pengujian data berdasarkan pembagian dengan masing-masing perbandingan baik yang menggunakan kombinasi antara Word Embedding FastText dan CNN maupun yang hanya menggunakan CNN saja :

47. Pengujian Word Embedding FastText dan CNN

48. Pembagian data 90:10

49. Epoch

Serangkaian baris teks yang berasal dari proses pelatihan model pembelajaran mesin digambarkan di sini. Setiap baris dimulai dengan "epoch", diikuti oleh pecahan yang menunjukkan kemajuan pelatihan (misalnya, 1/50, 2/50, dst.) dan mencakup berbagai metrik untuk setiap epoch, termasuk loss, akurasi (acc), val\_loss, dan val\_acc. Angka-angka yang mengikuti metrik ini menunjukkan nilai pada titik tertentu dalam proses pelatihan. Di akhir setiap baris, jumlah sampel yang diproses per langkah dan total langkah per epoch disebutkan.

Pada gambar diatas menunjukkan bahwa nilai pada iterasi atau epoch ke-10 di dapatkan accuracy tertinggi yaitu sebesar 92%.

Sedangkan pada gambar diatas menunjukkan bahwa nilai pada iterasi akhir atau epoch 41-50 di dapatkan accuracy yang konsisten di sebesar 92%.

1. Grafik

2. Prediksi

Metrik Evaluasi Model :

Kelas 0 (Negatif) : Kinerja sangat baik dengan f1-score 93%.

Kelas 1 (Netral) : Kinerja buruk dengan f1-score 12%, kemungkinan karena sampel sedikit.

Kelas 2 (Positif) : Kinerja baik dengan f1-score 90%.

Rata-rata Keseluruhan :

Micro Avg: 0.90 untuk precision, recall, dan f1-score, menghitung keseluruhan rata-rata berdasarkan jumlah total prediksi yang benar.

Macro Avg: 0.66 untuk precision, 0.64 untuk recall, dan 0.65 untuk f1-score, menghitung rata-rata untuk setiap kelas tanpa mempertimbangkan jumlah sampel.

Weighted Avg: Menghitung rata-rata dengan mempertimbangkan jumlah sampel di setiap kelas, menunjukkan kinerja model yang lebih representatif dari distribusi kelas.

1. Klasifikasi

Hasil evaluasi menunjukkan model bekerja baik untuk kelas 0 dan 2 dengan f1-score masing-masing 93% dan 90%, namun kinerja sangat buruk pada kelas 1 dengan f1-score hanya 11%, kemungkinan disebabkan oleh jumlah sampel yang sedikit. Secara keseluruhan, model memiliki akurasi 90%, tetapi performa tidak merata di semua kelas.

1. Pembagian data 80:20

2. Epoch

Pada gambar diatas menunjukkan bahwa nilai pada iterasi awal atau epoch ke-12 di dapatkan accuracy tertinggi yaitu sebesar 90%.

Sedangkan pada gambar diatas menunjukkan bahwa nilai pada iterasi akhir atau epoch 41-50 di dapatkan accuracy yang sedikit menurun dikisaran 88%.

1. Grafik

Dalam grafik loss, kita melihat bahwa training loss terus menurun karena model belajar dari 80% data pelatihan, sementara validation loss (untuk 20% data validasi) mulai meningkat seiring waktu. Karena kemampuan model untuk memprediksi data pelatihan yang lebih baik maka training accuracy terus meningkat, seperti yang ditunjukkan pada grafik accuracy. Namun, validation accuracy mencapai puncaknya sekitar epoch ke-12, dan kemudian tetap stabil atau sedikit menurun. Ini menunjukkan bahwa model mungkin terlalu berkonsentrasi pada data pelatihan dan mulai kehilangan kemampuan untuk menggeneralisasi pada data validasi. Kondisi ini disebut overfitting.

1. Prediksi

Metrik Evaluasi Model :

Kelas 0 (Negatif) : Kinerja sangat baik dengan f1-score 93%.

Kelas 1 (Netral) : Kinerja buruk dengan f1-score 31%, kemungkinan karena sampel sedikit.

Kelas 2 (Positif) : Kinerja baik dengan f1-score 87%.

Rata-rata Keseluruhan :

Micro Avg: 0.89 untuk precision, 0.88 untuk recall, dan 0.88 untuk f1-score, menghitung keseluruhan rata-rata berdasarkan jumlah total prediksi yang benar.

Macro Avg: 0.73 untuk precision, 0.68 untuk recall, dan 0.70 untuk f1-score, menghitung rata-rata untuk setiap kelas tanpa mempertimbangkan jumlah sampel.

Weighted Avg: Menghitung rata-rata dengan mempertimbangkan jumlah sampel di setiap kelas, menunjukkan kinerja model yang lebih representatif dari distribusi kelas.

1. Klasifikasi

Model ini bekerja sangat baik pada kelas 0 dan 2 dengan F1-score masing-masing 93% dan 88%, namun kinerja pada kelas 1 sangat rendah (F1-score 33%) karena jumlah sampel yang sedikit. Secara keseluruhan, akurasi model adalah 88%, menunjukkan performa yang cukup baik meskipun ada ketidakseimbangan antara kelas.

1. Pembagian data 70:30

2. Epoch

Pada gambar diatas menunjukkan bahwa nilai pada iterasi awal atau epoch 1-10 di dapatkan accuracy tertinggi yaitu sebesar 87%.

Sedangkan pada gambar diatas menunjukkan bahwa nilai pada iterasi akhir atau epoch ke-42 di dapatkan accuracy meningkat yaitu sebesar 90%.

1. Grafik

Tidak berbeda jauh dengan dua grafik sebelumnya yang membandingkan kinerja model pada data pelatihan dan data validasi selama 50 epoch. terlihat bahwa training loss terus menurun, dan validation loss juga meningkat setelah beberapa epoch. Namun pada perbandingan 70:30 memperlihatkan bahwa training accuracy terus meningkat hingga mendekati 100%, sedangkan validation accuracy menunjukkan kondisi yang sama seperti kedua perbandingan sebelumnya.

1. Prediksi

Metrik Evaluasi Model :

Kelas 0 (Negatif) : Kinerja sangat baik dengan f1-score 94%.

Kelas 1 (Netral) : Kinerja buruk dengan f1-score 19%, kemungkinan karena sampel sedikit.

Kelas 2 (Positif) : Kinerja baik dengan f1-score 89%.

Rata-rata Keseluruhan :

Micro Avg: 0.91 untuk precision, 0.89 untuk recall, dan 0.90 untuk f1-score, menghitung keseluruhan rata-rata berdasarkan jumlah total prediksi yang benar.

Macro Avg: 0.72 untuk precision, 0.65 untuk recall, dan 0.68 untuk f1-score, menghitung rata-rata untuk setiap kelas tanpa mempertimbangkan jumlah sampel.

Weighted Avg: Menghitung rata-rata dengan mempertimbangkan jumlah sampel di setiap kelas, menunjukkan kinerja model yang lebih representatif dari distribusi kelas.

1. Klasifikasi

Model ini memiliki performa yang sangat baik pada kelas 0 dan 2 dengan F1-score masing-masing 0.94 dan 0.90, menunjukkan bahwa model mampu memprediksi kedua kelas ini dengan baik. Namun, performa pada kelas 1 sangat rendah (F1-score 0.22) karena jumlah sampel yang sedikit dan deteksi yang buruk. Secara keseluruhan, model memiliki akurasi 90%, dengan precision dan recall yang tinggi, tetapi ketidakseimbangan kelas mempengaruhi kinerja pada kelas minoritas.

1. Pengujian Convolutional Neural Network (CNN)

2. Pembagian data 90:10

Pada gambar diatas menunjukkan bahwa nilai pada iterasi atau epoch ke-3 di dapatkan accuracy tertinggi yaitu sebesar 90%.

1. Pembagian data 80:20

Pada gambar diatas menunjukkan bahwa nilai pada iterasi atau epoch ke-3 di dapatkan accuracy tertinggi yaitu sebesar 88%.

1. Pembagian data 70:30

Pada gambar diatas menunjukkan bahwa nilai pada iterasi atau epoch ke-8 di dapatkan accuracy atau hasil training tertinggi yaitu sebesar 89%.

Berikut tabel yang membandingkan hasil akurasi dan loss model pada data validasi dengan menggunakan dua pendekatan yang berbeda yaitu kombinasi Word Embedding FastText - CNN dan yang hanya CNN saja, untuk tiga skenario pembagian data (90:10, 80:20, dan 70:30):

Sedangkan tabel dan grafik diagram diatas menunjukkan hasil perbandingan kinerja antara kedua model berdasarkan nilai Loss dengan tiga skenario pembagian data yang berbeda.

Sedangkan tabel dan grafik diagram diatas menunjukkan hasil perbandingan kinerja antara kedua model berdasarkan nilai Validation Loss dengan tiga skenario pembagian data yang berbeda.

Berdasarkan semua hasil pengujian yang telah dilakukan, perbandingan antara model kombinasi “Word Embedding FastText – CNN” dan model “CNN” saja menunjukkan bahwa model kombinasi cenderung memiliki kinerja yang lebih baik dengan rata-rata 91% - 100% pada nilai accuracy dan 90% - 92% pada nilai validation accuracy. Sedangkan yang hanya menggunakan model “CNN” memiliki rata-rata 92% - 98% pada nilai accuracy dan 88% - 90% pada nilai validation accuracy. Meskipun accuracy pada data pelatihan bisa sangat tinggi, validation accuracy memberikan gambaran yang lebih realistis tentang seberapa baik model akan bekerja pada data yang tidak terlihat sebelumnya. Perbedaan besar antara accuracy dan validation accuracy bisa menjadi tanda overfitting, di mana model bekerja sangat baik pada data pelatihan tetapi kurang baik pada data baru.

Namun ketika melihat nilai loss, model “Word Embedding FastText – CNN” konsisten menunjukkan nilai loss yang lebih rendah dibandingkan dengan model “CNN” saja. Begitupun pada validation loss, model kombinasi “Word Embedding FastText – CNN” lebih efisien dan stabil, dengan error prediksi yang lebih rendah. Perbedaan yang cukup signifikan pada pembagian data 70:30 (36% vs 65%).

1. Hasil Prediksi Model

Secara keseluruhan, model yang menggabungkan Word Embedding FastText dengan CNN menunjukkan kinerja yang baik dalam mengklasifikasikan sentimen ulasan pengguna. Namun, kesalahan dalam beberapa kasus mengindikasikan adanya ruang untuk peningkatan, terutama dalam menangani bahasa informal dan ulasan yang lebih ambigu. Pembahasan ini juga menyoroti pentingnya validasi model pada dataset yang lebih beragam untuk mengurangi kesalahan prediksi dan meningkatkan akurasi klasifikasi sentimen.

**4. Kesimpulan**

Berdasarkan pengujian yang telah dilakukan dalam penelitian ini, beberapa kesimpulan dapat ditarik sebagai berikut:

1. Dataset dengan 3199 ulasan pada aplikasi JAKI diuji dengan tiga kategori yaitu positif, negatif dan netral. Hasil pengujian menunjukkan bahwa kombinasi “Word Embedding FastText – CNN” efektif dalam menganalisis sentimen. Dimana model yang menggunakan kombinasi ini mencapai nilai accuracy tinggi antara 91% - 100% serta degan nilai loss yang lebih rendah dan stabil secara konsisten ditunjukkan oleh model kombinasi.

2. Penggunaan model CNN memberikan pengaruh positif dalam menganalisis sentimen yang mampu memproses dan mengidentifikasi pola dalam teks dengan baik terutama dalam hal akurasi. Namun juga memiliki kecenderungan overfitting, yang ditandai dengan nilai validation loss yang lebih tinggi antara 37% - 65% dibandingkan dengan training loss terutama ketika tidak dikombinasikan dengan teknik embedding seperti FastText. Hal ini menunjukkan bahwa meskipun CNN efektif dalam mengenali pola, tetapi kurang optimal dalam meminimalkan error prediksi tanpa dukungan representasi kata yang lebih kaya.

Dengan demikian, dapat disimpulkan bahwa penerapan Word Embedding FastText dalam analisis sentimen terhadap pengaruh model CNN terbukti efektif dan memberikan manfaat dalam analisis sentimen, terutama dalam meningkatkan stabilitas akurasi model serta meminimalkan kesalahan dalam memprediksi sentimen. ini menunjukkan kemampuannya dalam generalisasi yang lebih baik pada data baru.

Meskipun kombinasi Word Embedding FastText dan CNN telah terbukti efektif. Berdasarkan hasil penelitian yang telah dilakukan, penelitian selanjutnya disarankan untuk dapat mengeksplorasi penggunaan kombinasi metode lain seperti Long Short-Term Memory (LSTM) ataupun Transformer-based models seperti BERT yang lebih baik dalam memperbaiki pemahaman konteks dan mengurangi overfitting dan meminimalkan error prediksi (loss) dalam analisis sentimen.

**Referensi**

[1] Al-Areef, M. H., & Saputra S, K. (2023). Analisis Sentimen Pengguna Twitter Mengenai Calon Presiden Indonesia Tahun 2024 Menggunakan Algoritma LSTM. Jurnal SAINTIKOM (Jurnal Sains Manajemen Informatika Dan Komputer), 22(2), 270. https://doi.org/10.53513/jis.v22i2.8680

[2] Alghifari, D. R., Edi, M., & Firmansyah, L. (2022). Implementasi Bidirectional LSTM untuk Analisis Sentimen Terhadap Layanan Grab Indonesia. Jurnal Manajemen Informatika (JAMIKA), 12(2), 89–99. https://doi.org/10.34010/jamika.v12i2.7764

[3] Alvi Hasanah, N., Nanik Suciati, & Diana Purwitasari. (2021). Pemantauan Perhatian Publik terhadap Pandemi COVID-19 melalui Klasifikasi Teks dengan Deep Learning. Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi), 5(1), 193–202. https://doi.org/10.29207/resti.v5i1.2927

[4] Amalia, P. R., & Winarko, E. (2021). Aspect-Based Sentimen Analysis on Indonesian Restaurant Review Using a Combination of Convolutional Neural Network and Contextualized Word Embedding. IJCCS (Indonesian Journal of Computing and Cybernetics Systems), 15(3), 285. https://doi.org/10.22146/ijccs.67306

[5] Andriyanto, D., Said, F., Titiani, F., & Erni, E. (2021). Analisis Kesuksesan Aplikasi Jakarta Kini (JAKI) Menggunakan Model Delone and McLean. Paradigma - Jurnal Komputer Dan Informatika, 23(1), 43–48. https://doi.org/10.31294/p.v23i1.10018

[6] Azhar, K. M., Santoso, I., & Adi, A. (2021). Implementasi Deep Learning Menggunakan Metode Convolutional Neural Network Dan Algoritma Yolo Dalam Low Vision. Transient: Jurnal Ilmiah Teknik Elektro, 10(3), 502–509.

[7] Daniati, E., & Utama, H. (2023). Analisis Sentimen Dengan Pendekatan Ensemble Learning Dan Word Embedding Pada Twitter. Journal of Information System Management (JOISM), 4(2), 125–131. https://doi.org/10.24076/joism.2023v4i2.973

[8] Dinata, R. K., Safwandi, S., Hasdyna, N., & Azizah, N. (2020). Analisis K-Means Clustering pada Data Sepeda Motor. INFORMAL: Informatics Journal, 5(1), 10. https://doi.org/10.19184/isj.v5i1.17071

[9] Hermanto, D. T., Setyanto, A., & Luthfi, E. T. (2021). Algoritma LSTM-CNN untuk Binary Klasifikasi dengan Word2Vec pada Media Online. Creative Information Technology Journal, 8(1), 64. https://doi.org/10.24076/citec.2021v8i1.264

[10] Jihad, M. A. A., Adiwijaya, & Astuti, W. (2021). Analisis Sentimen Terhadap Ulasan Film Menggunakan Word2Vec dan SVM. E-Proceeding of Engineering, 8(4), 4136–4144.

[11] Kristiawan, K., Somali, D. D., Linggan jaya, T. A., & Widjaja, A. (2020). Deteksi Buah Menggunakan Supervised Learning dan Ekstraksi Fitur untuk Pemeriksa Harga. Jurnal Teknik Informatika Dan Sistem Informasi, 6(3), 541–548. https://doi.org/10.28932/jutisi.v6i3.3029

[12] Manggopa, R., Rantung, V. P., Kembuan, O., Informatika, T., Teknik, F., & Manado, U. N. (2024). Aplikasi Analisis Sentimen Terhadap Kebijakan MBKM Menggunakan Algoritma Convolutional Neural Network ( CNN ) Berbasis Web. 45–53.

[13] Muhammad Afif Raihan, & Erwin Budi Setiawan. (2022). Aspect Based Sentimen Analysis with FastText Feature Expansion and Support Vector Machine Method on Twitter. Jurnal RESTI (Rekayasa Sistem Dan Teknologi Informasi), 6(4), 591–598. https://doi.org/10.29207/resti.v6i4.4187

[14] Mustasaruddin, M., Budianita, E., Fikry, M., & Yanto, F. (2023). Klasifikasi Sentimen Review Aplikasi MyPertamina Menggunakan Word Embedding FastText dan SVM (Support Vector Machine). Jurnal Sistem Komputer Dan Informatika (JSON), 4(3), 526. https://doi.org/10.30865/json.v4i3.5695

[15] Nafisah Nurul Hakim. (2020). Implementasi Machine Learning pada Sistem Prediksi Kejadian dan Lokasi Patah Rel Kereta Api di Indonesia. Jurnal Sistem Cerdas, 3(1), 25–35. https://doi.org/10.37396/jsc.v3i1.58

[16] Nanda, S., Elvia, B., Fikry, M., & Pizaini. (2023). Analisis Sentimen Review Aplikasi MyPertamina Menggunakan Word Embedding FastText dan Algoritma K-Nearest Neighbor. INFORMASI (Jurnal Informatika Dan Sistem Informasi), 15(1), 91.

[17] Nugroho, P. A., Fenriana, I., & Arijanto, R. (2020). Implementasi Deep Learning Menggunakan Convolutional Neural Network (CNN) Pada Ekspresi Manusia. Algor, 2(1), 12–21.

[18] Nurdewi, N. (2022). Implementasi Personal Branding Smart Asn Perwujudan Bangga Melayani Di Provinsi Maluku Utara. SENTRI: Jurnal Riset Ilmiah, 1(2), 297–303. https://doi.org/10.55681/sentri.v1i2.235

[19] Nurdin, A., Anggo Seno Aji, B., Bustamin, A., & Abidin, Z. (2020). Perbandingan Kinerja Word Embedding Word2Vec, Glove, Dan FastText Pada Klasifikasi Teks. Jurnal Tekno Kompak, 14(2), 74. https://doi.org/10.33365/jtk.v14i2.732

[20] Purnasiwi, R. G., Kusrini, & Hanafi, M. (2023). Analisis Sentimen Pada Review Produk Skincare Menggunakan Word Embedding dan Metode Long Short-Term Memory (LSTM). Innovative: Journal Of Social Science Research, 3(2), 11433–11448.

[21] Putu Sawitra Danda Prasetia, I., Adi Sastra Wijaya, K., Putu Dharmanu Yudartha, I., & Savitri, R. (2024). Analisis Persepsi Masyarakat Terhadap Aplikasi JAKI (Jakarta Kini) di Provinsi DKI Jakarta. Jurnal Ilmu Sosial Dan Humaniora, 7(1), 1–12. https://jayapanguspress.penerbit.org/index.php/ganaya

[22] Rahman, M. D., Djunaidy, A., & Mahananto, F. (2021). Penerapan Weighted Word Embedding pada Pengklasifikasian Teks Berbasis Recurrent Neural Network untuk Layanan Pengaduan Perusahaan Transportasi. Jurnal Sains Dan Seni ITS, 10(1). https://doi.org/10.12962/j23373520.v10i1.56145

[23] Riza, M. A., & Charibaldi, N. (2021). Emotion Detection in Twitter Social Media Using Long Short-Term Memory (LSTM) and Fast Text. International Journal of Artificial Intelligence & Robotics (IJAIR), 3(1), 15–26. https://doi.org/10.25139/ijair.v3i1.3827

[24] Rodhi, M., Fadhlurrahman, R., & Aknuranda, I. (2022). Evaluasi Usability pada Aplikasi JAKI menggunakan Pengujian Usability. Jurnal Pengembangan Teknologi Informasi Dan Ilmu Komputer, 6(8), 3857–3863. https://j-ptiik.ub.ac.id/index.php/j-ptiik/article/view/11464

[25] Sofiana, A. (2023). Analisis Implementasi Open Government Melalui Aplikasi Jakarta Kini (Jaki) Pada Fitur Jak-Respon Dalam Penyebarluasan Informasi Publik. In Journal of Social Contemplativa (Vol. 1, Issue 1, pp. 45–53). https://doi.org/10.61183/jsc.v1i1.10

[26] Tuasamu, Z., M. Lewaru, N. A. I., Idris, M. R., Syafaat, A. B. N., Faradilla, F., Fadlan, M., Nadiva, P., & Efendi, R. (2023). Analisis Sistem Informasi Akuntansi Siklus Pendapatan Menggunakan DFD Dan Flowchart Pada Bisnis Porobico. Jurnal Bisnis Manajemen, 1(2), 495–510.

[27] Vidya Chandradev, I Made Agus Dwi Suarjaya, & I Putu Agung Bayupati. (2023). Analisis Sentimen Review Hotel Menggunakan Metode Deep Learning BERT. Jurnal Buana Informatika, 14(02), 107–116. https://doi.org/10.24002/jbi.v14i02.7244

[28] Yulianeu, A., & Oktamala, R. (2022). Sistem Informasi Geografis Trayek Angkutan Umum Di Kota Tasikmalaya Berbasis Web. JUTEKIN (Jurnal Teknik Informatika), 10(2). https://doi.org/10.51530/jutekin.v10i2.669

[29] Yuliska, Y., Qudsi, D. H., Lubis, J. H., Syaliman, K. U., & Najwa, N. F. (2021). Analisis Sentimen pada Data Saran Mahasiswa Terhadap Kinerja Departemen di Perguruan Tinggi Menggunakan Convolutional Neural Network. Jurnal Teknologi Informasi Dan Ilmu Komputer, 8(5), 1067. https://doi.org/10.25126/jtiik.2021854842

[30] Yuniarossy, B. A., Hindrayani, K. M., & Damaliana, A. T. (2024). Analisis Sentimen Terhadap Isu Feminisme Di Twitter Menggunakan Model Convolutional Neural Network (CNN). Jurnal Lebesgue : Jurnal Ilmiah Pendidikan Matematika, Matematika Dan Statistika, 5(1), 477–491. https://doi.org/10.46306/lb.v5i1.585