

**PENGGUNAAN WORD EMBEDDING WORD2VEC DALAM
PENGEMBANGAN MODEL CNN STUDY KASUS ANALISIS
SENTIMEN TEMPAT WISATA MAKASSAR**

SKRIPSI

Diajukan Sebagai Salah Satu Syarat Untuk Mendapatkan Gelar
Sarjana Komputer (S.Kom) Program Studi Informatika



ARVIANDA 105841102520

PRODI INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS MUHAMMADIYAH MAKASSAR

2024

**PENGGUNAAN WORD EMBEDDING WORD2VEC DALAM
PENGEMBANGAN MODEL CNN STUDY KASUS ANALISIS
SENTIMEN TEMPAT WISATA MAKASSAR**

Diajukan Sebagai Salah Satu Syarat Untuk Mendapatkan Gelar
Sarjana Komputer (S.Kom) Program Studi Informatika

Disusun Dan Diajukan Oleh :

ARVIANDA

105841102520

**PRODI INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS MUHAMMADIYAH MAKASSAR**

2024



UNIVERSITAS MUHAMMADIYAH MAKASSAR
FAKULTAS TEKNIK

GEDUNG MENARA IQRA LT. 3

Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221
Website: www.unismuh.ac.id, e-mail: unismuh@gmail.com
Website: <http://teknik.unismuh.makassar.ac.id>



PENGESAHAN

Skripsi atas nama ARVIANDA dengan nomor induk Mahasiswa 105 84 11025 20, dinyatakan diterima dan disahkan oleh Panitia Ujian Tugas Akhir/Skripsi sesuai dengan Surat Keputusan Dekan Fakultas Teknik Universitas Muhammadiyah Makassar Nomor : 0008/SK-Y/55202/091004/2024, sebagai salah satu syarat guna memperoleh gelar Sarjana Komputer pada Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar pada hari Sabtu tanggal 26 Agustus 2024.

Panitia Ujian : _____ Makassar, 21 Safar 1446 H
26 Agustus 2024 M

1. Pengawas Umum

- a. Rektor Universitas Muhammadiyah Makassar
Dr. Ir. H. Abd. Rahim Nafda, ST, MT, IPU
- b. Dekan Fakultas Teknik Universitas Hasanuddin
Prof. Dr. Eng. Muhammad Israh Ramli, ST, MT

2. Penguji

- a. Ketua : Dr. Ir. Zahr Zainuddin, M.Sc.
- b. Sekretaris : Rizki Yuliana Bekti, ST, MT

3. Anggota

- 1. Lukman Anas, S.Kom., MT.
- 2. Lukman, S.Kom., MT.
- 3. Muhyiddin A.M Hayat, S.Kom., MT.

Mengetahui :

Pembimbing I

Titin Wahyuni, S.Pd., MT.

Pembimbing II

Fahrim Irhamja Rahman, S.Kom., MT.





UNIVERSITAS MUHAMMADIYAH MAKASSAR
FAKULTAS TEKNIK

GEDUNG MENARA IDRA L.T. 3
Jl. Sultan Alauddin No. 259 Telp. (0411) 866 972 Fax (0411) 865 588 Makassar 90221
Website: www.unismuh.ac.id, e_mail: unismuh@gmail.com
Website: <http://teknik.unismuh.makassar.ac.id>

وَسَبِّحْ بِحَمْدِ رَبِّكَ اللَّيْلَ وَالنَّهَارَ

HALAMAN PENGESAHAN

Tugas Akhir ini diajukan untuk memenuhi syarat ujian guna memperoleh gelar Sarjana Komputer (S.Kom) Program Studi Informatika Fakultas Teknik Universitas Muhammadiyah Makassar.

Judul Skripsi : **PENGGUNAAN WORD EMBEDDING WORD2VEC DALAM PENGEMBANGAN MODEL CNN STUDY KASUS ANALISIS SENTIMEN TEMPAT WISATA MAKASSAR**

Nama : **ARVIANDA**
Stambuk : **105 84 11025 20**

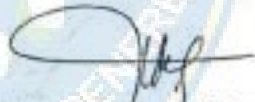
Makassar, 26 Agustus 2024

Telah Diperiksa dan Disetujui
Oleh Dosen Pembimbing:

Pembimbing I

Pembimbing II


Titin Wahyuni, S.Pd., MT.


Fahrin Irtamna Rahman, S.Kom., MT.

Mengetahui,

Ketua Program Studi Informatika


Mulyandani M. Hidayat, S.Kom., MT.

NPM. 1504 577

MOTTO DAN PERSEMBAHAN

MOTTO

"Setiap langkah yang kita ambil dalam perjuangan adalah investasi untuk masa depan yang lebih baik. Kerja keras bukan hanya tentang mencapai tujuan, tetapi juga tentang proses belajar dan tumbuh yang membawa kita menuju kesuksesan."

~ARVIANDA~

PERSEMBAHAN

Ku persembahkan skripsi ini kepada orang tua tercinta, yang selalu menjadi sumber inspirasi dan kekuatan dalam setiap langkah perjalanan hidupku. Sejak kecil, kalian telah mengajarkan arti kerja keras, ketekunan, dan kejujuran, nilai-nilai yang selalu saya bawa dalam menghadapi setiap tantangan. Terima kasih atas cinta yang tulus, dukungan tanpa henti, dan pengorbanan yang telah kalian berikan untuk mewujudkan impian ini. Setiap doa yang kalian panjatkan dan setiap nasihat yang kalian berikan telah membentuk diriku menjadi pribadi yang lebih baik, dan tanpa kalian, perjalanan ini tidak mungkin terwujud. Skripsi ini bukan hanya hasil dari usaha dan kerja keras, tetapi juga merupakan lambang dari harapan dan impian kita bersama. Saya berharap dapat membuat kalian bangga, dan semoga hasil karya ini bisa memberikan manfaat serta menginspirasi orang lain, seperti kalian telah menginspirasi diriku. Dengan sepenuh hati, saya dedikasikan pencapaian ini untuk kalian berdua.

ABSTRAK

ARVIANDA. PENGGUNAAN WORD EMBEDDING WORD2VEC DALAM PENGEMBANGAN MODEL CNN STUDY KASUS ANALISIS SENTIMEN TEMPAT WISATA MAKASSAR (dibimbing oleh Fachrim Irhamna Rahman S.Kom., M.T dan ibu Titin Wahyuni, S.Pd., M.T)

Penelitian ini bertujuan untuk mengevaluasi pengaruh penerapan teknik Word Embedding Word2Vec terhadap akurasi model Convolutional Neural Network (CNN) dalam analisis sentimen ulasan tempat wisata di Makassar. Analisis sentimen adalah proses mengidentifikasi dan mengklasifikasikan emosi atau opini yang terkandung dalam teks, apakah positif, negatif, atau netral. Dataset penelitian terdiri dari 4500 ulasan wisata yang diambil dari Google Maps. Data ini kemudian diolah menggunakan teknik Word2Vec untuk menghasilkan representasi vektor dari kata-kata dalam ulasan. Vektor ini digunakan sebagai input ke dalam model CNN untuk klasifikasi sentimen. Menggunakan tiga skenario pembagian data yaitu 90:10, 80:20, dan 70:30 untuk melatih dan menguji model. Hasil penelitian menunjukkan bahwa penerapan Word2Vec pada model CNN memberikan peningkatan akurasi dalam prediksi sentimen. Model CNN dengan Word2Vec berhasil mencapai akurasi 79%, sementara model CNN tanpa Word2Vec hanya mencapai akurasi 74%. Hal ini menunjukkan bahwa penggunaan Word2Vec dapat meningkatkan performa model dalam mengklasifikasikan sentimen ulasan tempat wisata.

Kata Kunci : Analisis Sentimen, Word2Vec, Convolutional Neural Network, Tempat Wisata, Deep Learning

ABSTRACT

ARVIANDA. THE USE OF WORD EMBEDDING WORD2VEC IN THE DEVELOPMENT OF CNN MODELS: A CASE STUDY OF SENTIMENT ANALYSIS ON TOURIST ATTRACTIONS IN MAKASSAR (supervised by Fachrim Irhamna Rahman S.Kom., M.T and Titin Wahyuni, S.Pd., M.T)

This research aims to evaluate the effect of applying the Word Embedding Word2Vec technique on the accuracy of the Convolutional Neural Network (CNN) model in sentiment analysis of tourist attraction reviews in Makassar. Sentiment analysis is the process of identifying and classifying emotions or opinions contained in text, whether positive, negative, or neutral. The research dataset consists of 4500 tourist attraction reviews taken from Google Maps. The data was then processed using the Word2Vec technique to generate vector representations of the words in the reviews. These vectors were used as input to the CNN model for sentiment classification. The study employed three data splitting scenarios, namely 90:10, 80:20, and 70:30, for training and testing the model. The results showed that the application of Word2Vec in the CNN model improved sentiment prediction accuracy. The CNN model with Word2Vec achieved an accuracy of 79%, while the CNN model without Word2Vec only reached an accuracy of 74%. This indicates that the use of Word2Vec can enhance the performance of the model in classifying sentiment in tourist attraction reviews.

Keywords : *Sentiment Analysis, Word2Vec, Convolutional Neural Network, Tourist Attractions, Deep Learning*

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Dengan mengucapkan bismillahirrahmanirrahim, saya memulai penulisan kata pengantar ini sebagai ungkapan rasa syukur kepada Allah SWT, yang telah memberikan kemudahan dan kekuatan sehingga saya dapat menyelesaikan skripsi ini dengan judul “**PENGGUNAAN WORD EMBEDDING WORD2VEC DALAM PENGEMBANGAN MODEL CNN STUDY KASUS ANALISIS SENTIMEN TEMPAT WISATA MAKASSAR**”.

Skripsi ini merupakan buah dari proses pembelajaran yang panjang dan penuh tantangan di **UNIVERSITAS MUHAMMADIYAH MAKASSAR**, yang tidak mungkin terwujud tanpa dukungan dari berbagai pihak. Oleh karena itu, izinkan saya untuk menyampaikan terima kasih kepada:

1. Ayahanda **Ancu** , beliau yang menjadi inti tulang punggung keluarga. Meskipun beliau tidak sempat merasakan pendidikan hingga bangku perkuliahan, namun beliau mampu mendidik penulis menjadi laki-laki yang kuat dan tegar dalam segala rintangan, hingga penulis mampu menyelesaikan tugas akhir ini.
2. Ibunda **Melda**, pintu surgaku. Beliau sangat berperan penting dalam menyelesaikan program studi penulis. Beliau juga memang tidak sempat merasakan pendidikan hingga bangku perkuliahan, namun gigih dalam memanjatkan doa yang selalu beliau berikan yang tiada henti meminta kepada Tuhan Yang Maha Esa, hingga penulis mampu menyelesaikan akhir ini.
3. Terima kasih yang tulus kepada bapak **Fachrim Irhamna Rahman S.Kom., M.T** dan ibu **Titin Wahyuni, S.Pd., M.T** yang telah menjadi pembimbing yang luar biasa selama proses penulisan tugas akhir ini. Bimbingan, kesabaran, dan wawasan yang diberikan telah menjadi penerang dalam menuntun saya menyelesaikan penelitian ini. Tanpa arahan yang berharga beliau, skripsi ini tidak akan dapat terwujud.

Saya berharap skripsi ini dapat memberikan kontribusi yang berarti bagi pengembangan ilmu INFORMATIKA dan menjadi sumber referensi yang berguna bagi peneliti lainnya.

Akhir kata, semoga skripsi ini dapat diterima dan memenuhi salah satu syarat untuk mencapai gelar sarjana di FAKULTAS TEKNIK

Makassar, 18 Mei 2024

ARVIANDA



DAFTAR ISI

HALAMAN JUDUL	ii
HALAMAN PENGESAHAN.....	iii
HALAMAN PERSETUJUAN	iv
HALAMAN PERSEMBAHAN	v
ABSTRAK	vi
ABSTRACT.....	vii
KATA PENGANTAR.....	viii
DAFTAR ISI.....	x
DAFTAR GAMBAR.....	xii
DAFTAR TABEL.....	xiii
DAFTAR LAMPIRAN	xiv
DAFTAR ISTILAH.....	xv
BAB I PENDAHULUAN.....	1
A. Latar Belakang.....	1
B. Rumusan Masalah	3
C. Tujuan Penelitian	3
D. Manfaat Penelitian.....	3
E. Ruang Lingkup Penelitian.....	4
F. Sistematika Penulisan	4
BAB II TINJAUAN PUSTAKA.....	6
A. Landasan Teori.....	6
B. Penelitian Terkait	11
C. Kerangka Berpikir.....	17
BAB III METODE PENELITIAN	18
A. Tempat dan Waktu Penelitian.....	18
B. Alat dan Bahan.....	18
C. Perancangan Sistem.....	18
D. Pengujian Sitem	21
E. Teknik Analisi Data.....	23
BAB IV HASIL DAN PEMBAHASAN.....	25
A. Pengumpulan Data.....	25

B. Pelabelan Data	26
C. Reprocessing Data	27
D. Penerapan Metode	29
E. Pengujian dan Hasil Metode	54
BAB V PENUTUP	87
A. Kesimpulan	87
B. Saran	87
DAFTAR PUSTAKA	88

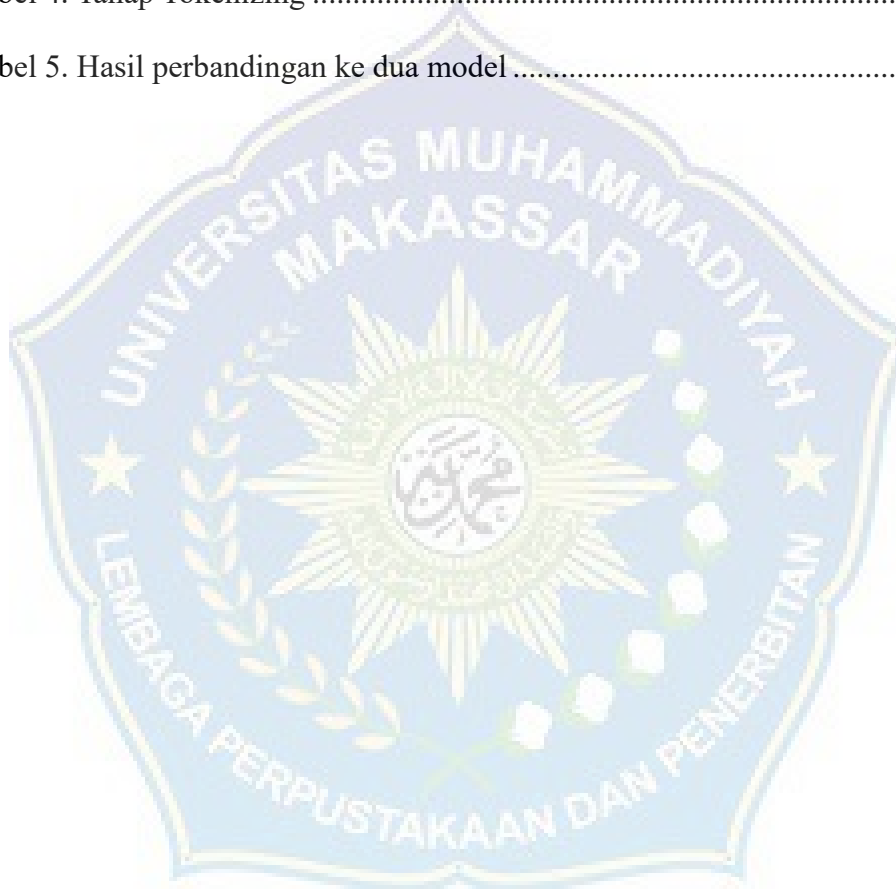


DAFTAR GAMBAR

Gambar 1. Arsitektur CNN	9
Gambar 2. Kerangka berpikir.....	17
Gambar 3. Perancangan sistem	19
Gambar 4. Perancangan sistem training.....	20
Gambar 5. Perancangan sistem testing.....	21
Gambar 6. Proses pengambilan data ulasan dengan Instant Data Scraper.....	25
Gambar 7. Dataset ulasan.....	26
Gambar 8. Proses epoch 90:10.....	71
Gambar 9. Grafik accuracy dan loss 90:10	72
Gambar 10. Hasil prediksi 90:10	73
Gambar 11. Hasil klasifikasi label 90:10	75
Gambar 12. Proses epoch 80:20.....	76
Gambar 13. Grafik accuracy dan loss 80:20	77
Gambar 14. Hasil Prediksi 80:20	78
Gambar 15. Hasil klasifikasi label 80:20	79
Gambar 16. Proses epoch 70:30.....	80
Gambar 17. Grafik accuracy dan loss 70:30	80
Gambar 18. Hasil prediksi 70:30	82
Gambar 19. Hasil klasifikasi label 70:30	83
Gambar 20. Proses epoch 90:10 hanya menggunakan cnn.....	84
Gambar 21. Proses epoch 80:20 hanya menggunakan cnn	84
Gambar 22. Proses epoch 70:30 menggunakan cnn saja	85

DAFTAR TABEL

Tabel 1. Tahap pelabelan data	27
Tabel 2. Tahap cleaning atau pembersihan.....	27
Tabel 3. Tahap Transform Case.....	28
Tabel 4. Tahap Tokenizing	29
Tabel 5. Hasil perbandingan ke dua model	85



DAFTAR LAMPIRAN

Lampiran 1. Source code.....	93
Lampiran 2. Dataset ulasan positif.....	98
Lampiran 3. Dataset ulasan negatif.....	99
Lampiran 4. Dataset ulasan netral.....	100
Lampiran 5. Dataset hasil tokenizing.....	101
Lampiran 6. Proses epoch.....	102
Lampiran 7. Grafik accuracy dan loss.....	104
Lampiran 8. Hasil prediksi.....	105
Lampiran 9. Hasil klasifikasi.....	106
Lampiran 10. Dataset uji hasil prediksi.....	107



DAFTAR ISTILAH

CNN	CNN adalah singkatan dari Convolutional Neural Network. CNN adalah jenis arsitektur jaringan saraf tiruan yang banyak digunakan dalam pengenalan gambar dan pengolahan data berbasis grid.
Word embedding	Teknik dalam pemrosesan bahasa alami (NLP) yang mengubah kata-kata menjadi representasi vektor numerik. Tujuan utamanya adalah untuk menangkap makna semantik dari kata-kata dan hubungan antar kata dalam bentuk vektor yang dapat diproses oleh model pembelajaran mesin.
Layer	Dalam konteks jaringan saraf tiruan, layer atau lapisan adalah komponen fundamental yang terdiri dari sejumlah neuron (juga disebut unit atau node) yang melakukan operasi pada data masukan dan meneruskannya ke lapisan berikutnya. Jaringan saraf tiruan biasanya terdiri dari beberapa jenis lapisan yang bekerja bersama untuk memproses data dan melakukan tugas tertentu, seperti klasifikasi atau prediksi.
Fitur visual	Fitur visual merujuk pada karakteristik atau atribut yang dapat diekstraksi dari gambar atau video dan digunakan untuk mengenali, mengklasifikasi, atau menganalisis konten visual tersebut. Dalam konteks pengolahan citra dan visi komputer, fitur visual sangat penting karena mereka memberikan representasi yang lebih terstruktur dan bermakna dari data mentah (piksel).
Softmax	Softmax adalah fungsi aktivasi yang digunakan dalam jaringan saraf tiruan, terutama dalam konteks klasifikasi

multi-kelas. Fungsi ini mengubah keluaran jaringan menjadi distribusi probabilitas, di mana setiap kelas mendapatkan nilai probabilitas antara 0 dan 1, dan jumlah dari semua probabilitas kelas adalah 1.

Epoch	Epoch dalam konteks pembelajaran mesin, khususnya dalam pelatihan jaringan saraf tiruan, mengacu pada satu siklus penuh melalui seluruh dataset pelatihan.
Python	Python adalah bahasa pemrograman tingkat tinggi yang dikembangkan oleh Guido van Rossum dan pertama kali dirilis pada tahun 1991. Python dikenal karena sintaksnya yang sederhana, kemudahan penggunaan, dan fleksibilitasnya.
ReLU	singkatan dari Rectified Linear Unit, adalah fungsi aktivasi yang sangat populer digunakan dalam jaringan saraf tiruan, terutama dalam Convolutional Neural Networks (CNNs) dan Deep Learning. Fungsi ini membantu memperkenalkan non-linearitas dalam model sambil menjaga efisiensi komputasi.
Flowchart	Representasi grafis dari langkah-langkah atau proses dalam bentuk simbol dan panah yang menunjukkan urutan dan hubungan antar langkah. Flowchart sering digunakan untuk memvisualisasikan alur kerja, algoritma, atau proses dalam berbagai konteks, termasuk pengembangan perangkat lunak, manajemen proyek, dan perencanaan bisnis.
GPU	GPU atau Graphics Processing Unit adalah jenis prosesor khusus yang dirancang untuk mempercepat pemrosesan grafik dan visual pada komputer. GPU awalnya dikembangkan untuk menangani tugas-tugas terkait grafis, seperti rendering gambar dan video, tetapi sekarang juga digunakan dalam berbagai aplikasi komputasi non-grafis

karena kemampuannya untuk melakukan banyak operasi secara paralel.

Tensorflow	sebuah framework open-source untuk komputasi numerik dan pembelajaran mesin yang dikembangkan oleh Google. TensorFlow memungkinkan pengguna untuk membangun dan melatih model pembelajaran mesin (machine learning) dan jaringan saraf tiruan (neural networks) dengan efisien.
Polaritas teks	Polaritas teks mengacu pada penilaian atau klasifikasi sentimen dari sebuah teks berdasarkan aspek positif, negatif, atau netral. Ini merupakan bagian dari analisis sentimen, yang bertujuan untuk menentukan bagaimana perasaan atau sikap pengarang terhadap topik tertentu.
ANN	Artificial Neural Network adalah model pembelajaran mesin yang terinspirasi oleh struktur dan fungsi otak manusia. ANN terdiri dari node atau neuron yang diatur dalam lapisan-lapisan dan digunakan untuk mengenali pola, melakukan klasifikasi, dan memprediksi output berdasarkan input yang diberikan.
Jejaring sosial	Platform atau aplikasi yang memungkinkan individu untuk berinteraksi, berbagi informasi, dan membangun hubungan sosial secara online. Jejaring sosial mencakup berbagai jenis layanan dan aplikasi yang memfasilitasi komunikasi dan koneksi antara pengguna di seluruh dunia.
Dord2vec	adalah ekstensi dari Word2Vec yang dirancang untuk menghasilkan representasi vektor tidak hanya untuk kata-kata, tetapi juga untuk dokumen atau kalimat lengkap. Ini memungkinkan pemodelan makna dan konteks yang lebih luas dalam teks, dan dapat digunakan untuk berbagai tugas

pemrosesan bahasa alami (NLP) seperti klasifikasi dokumen, pencarian informasi, dan analisis sentimen.

Pooling

Teknik dalam jaringan saraf tiruan (neural networks), khususnya dalam Convolutional Neural Networks (CNNs), yang digunakan untuk mengurangi dimensi data dan meminimalkan kompleksitas komputasi sambil mempertahankan informasi penting dari data. Pooling biasanya diterapkan setelah lapisan konvolusi untuk mengurangi ukuran fitur dan meningkatkan efisiensi serta kinerja model.



BAB I

PENDAHULUAN

A. Latar Belakang

Istilah sentiment analysis (Analisis sentiment) pertama kali diperkenalkan pada tahun 2003 menurut jurnal. Sentiment analysis adalah salah satu bidang penelitian dalam text mining yang berkolaborasi dengan Natural Language Processing (NLP), yang fokusnya adalah mengekstraksi pola dan menganalisis informasi berupa opini dari teks. Sentiment analysis mulai menjadi topik penting seiring dengan meningkatnya interaksi di media sosial, penggunaan berbagai forum dan blog, serta komentar dan penilaian di berbagai situs e-commerce. (Jihad et al., 2021)

Analisis sentimen merupakan suatu proses otomatis untuk mengekstraksi, mengolah, dan memahami data teks yang tidak terstruktur, dengan tujuan mendapatkan informasi sentimen yang terkandung dalam sebuah kalimat, pendapat, atau opini. Analisis ini bertujuan untuk mengevaluasi opini dan kecenderungan suatu pendapat terhadap sebuah topik, apakah bersifat negatif atau positif. (Arsi & Waluyo, 2021). Analisis sentimen digunakan untuk menemukan informasi berharga dari data yang tidak terstruktur. Oleh karena itu, penelitian ini diharapkan dapat mengidentifikasi sentimen terhadap obyek wisata di Makassar.

Obyek wisata yakni segala sesuatu di suatu daerah tujuan wisata yang menarik orang untuk datang berkunjung. Obyek wisata bisa berupa keindahan alam seperti gunung, danau, sungai, pantai, laut, atau bangunan bersejarah seperti museum, benteng, dan situs peninggalan sejarah. Obyek wisata memiliki potensi besar untuk dikembangkan sebagai sumber pendapatan daerah, karena selain menguntungkan tempat wisata itu sendiri, juga memberikan manfaat bagi infrastruktur pendukungnya. Obyek wisata adalah segala sesuatu yang ada di daerah tujuan wisata yang menarik orang untuk datang berkunjung. (Ningsih et al., 2019). Menurut Undang-Undang No. 10 Tahun 2009, kegiatan wisata mencakup berbagai jenis aktivitas wisata yang

didukung oleh berbagai fasilitas dan layanan yang disediakan oleh masyarakat, pengusaha, pemerintah, dan pemerintah daerah. (Bahits et al., 2020).

Word embedding bisa di artikan sebagai suatu teknik pembelajaran mesin yang menghasilkan representasi kata-kata dalam bentuk distribusi kontinu di dalam ruang dimensi yang lebih rendah. Secara umum, teknik ini menggunakan model pembelajaran berbasis jaringan saraf tiruan (JST). Salah satu teknik word embedding yang terkenal adalah Word2Vec, yang dikembangkan oleh Tomas Mikolov. Word2Vec merupakan terobosan dalam ekstraksi fitur kata-kata karena memperoleh fitur semantik kata-kata dari korpus teks. Dalam Word2Vec, setiap kata unik direpresentasikan oleh sejumlah angka yang membentuk vektor. Pemilihan vektor dilakukan melalui proses matematis yang menggambarkan tingkat kesamaan semantik antara kata-kata yang direpresentasikan oleh vektor tersebut. (Khomsah, 2021).

Convolutional Neural Network (CNN) merupakan metode yang berhasil dalam pengenalan pola, yang dikembangkan oleh Lecun pada tahun 1988. CNN memiliki arsitektur yang terdiri dari puluhan hingga ratusan lapisan. Setiap lapisan melakukan pembelajaran dengan menggunakan output dari lapisan sebelumnya sebagai input. CNN menggunakan tahap pelatihan filter untuk mengekstrak fitur visual dari gambar. Ukuran peta fitur kemudian dikurangi melalui operasi pooling. Proses ini diulang hingga fitur terdalam diekstraksi. Struktur umum dari CNN melibatkan lapisan konvolusional, lapisan pooling, dan jaringan fully connected. (Agung, 2023).

Deep Learning diperkenalkan oleh Geoffrey Hinton pada tahun 2006. Teknologi ini hadir untuk mengatasi keterbatasan metode machine learning konvensional. Salah satu keunggulannya adalah kemampuan untuk melakukan feature engineering secara otomatis. Deep learning mampu menghasilkan performa yang lebih baik dan semakin meningkat seiring dengan bertambahnya jumlah data. (Rachman & Santoso, 2021)

Analisis sentimen berbasis aspek menggunakan Deep Learning sudah banyak dilakukan oleh peneliti sebelumnya, namun hingga saat ini belum ada yang fokus pada objek wisata khususnya di wilayah Makassar dan Gowa. Ada

berbagai metode untuk menganalisis sentimen suatu topik. Dalam penelitian ini, deep learning dipilih sebagai metodenya. (Naquitasia, 2021).

Berdasarkan penelitian yang akan diangkat terhadap masalah di atas, maka di buat analisis sentimen menggunakan metode Convolutional Neural Network (CNN) dengan model Word2vec untuk mengetahui bagaimana cara mengimplementasikan tingkat akurasi dalam ulasan text pada obyek wisata Makassar.

B. Rumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan di atas, maka rumusan masalah yang diangkat pada penelitian ini yaitu, bagaimana pengaruh penerapan Word Embedding terhadap akurasi model Convolutional Neural Network (CNN) dalam memprediksi sentiment terhadap ulasan tempat wisata Makassar.

C. Tujuan Penelitian

Berdasarkan Rumusan Masalah di atas dapat disimpulkan bahwa tujuan penelitian ini adalah untuk mengevaluasi pengaruh penerapan word embedding terhadap akurasi model Convolutional Neural Network (CNN) dalam memprediksi sentimen terhadap tempat wisata di Makassar, dengan membandingkan kinerja model CNN yang menggunakan word embedding dengan yang tidak menggunakan.

D. Manfaat Penelitian

Harapannya, penelitian ini akan memberikan manfaat yang signifikan baik secara teoritis maupun praktis:

1. Secara Teoritis:
 - a. Untuk pengembangan ilmu pengetahuan, terutama pada bidang teknik informatika
 - b. Memberikan wawasan yang lebih dalam tentang penggunaan word embedding, khususnya Word2Vec, dalam konteks analisis sentimen
2. Secara Praktis
 - a. Bagi Peneliti:

- 1) Mendapatkan pengalaman praktis dalam penggunaan teknik word embedding Word2Vec dan pengembangan model Convolutional Neural Network (CNN) dalam konteks analisis sentimen.
 - 2) Sebagai portofolio yang berguna bagi peneliti di masa yang akan datang.
- b. Bagi Universitas
- 1) Sebagai bahan referensi untuk penelitian selanjutnya
 - 2) Sebagai bahan evaluasi bagi Universitas dalam mengembangkan keilmuan, dalam hal ini yang berkaitan dengan model Word2vec dan Metode CNN (Convolutional Neural Network) analisis sentimen text.

E. Ruang Lingkup Penelitian

Dari rumusan masalah di atas, dapat di rumuskan beberapa batasan masalah yaitu:

1. Penelitian ini terbatas pada beberapa tempat wisata di kota Makassar dan Gowa. Hasilnya mungkin tidak bisa langsung diterapkan pada tempat wisata di lokasi lain, kecuali untuk Wisata Kebun Gowa, Pantai Bosowa, Akkarena, Tanjung Bayang, dan Bugis Waterpark.
2. Terbatas pada penerapan hanya menggunakan model Word2vec saja tanpa mempertimbangkan model lain seperti glove dan fasttext.
3. Terbatas hanya menggunakan metode CNN saja tanpa mempertimbangkan metode analisis sentimen lainnya.

F. Sistematika Penulisan

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

BAB I PENDAHULUAN

Bab ini menjelaskan tentang latar belakang masalah, rumusan masalah, batasan masalah, tujuan, manfaat dan sistematika penulisan.

BAB II TINJAUAN PUSTAKA

Bab ini menjelaskan tentang teori-teori yang melandasi penulisandalam melaksanakan Skripsi.

BAB III METODE PENELITIAN

Membahas tentang metode penelitian dan alat yang digunakan untuk pembuatan sistem.

BAB IV ANALISA DAN PENGUJIAN

Bab ini berisikan hasil desain sistem serta pembahasan terhadap desain tersebut.

BAB V KESIMPULAN DAN SARAN

Bab terakhir ini akan memuat kesimpulan isi dari keseluruhan uraian bab-bab sebelumnya dan saran-saran dari hasil yang telah di peroleh serta yang di harapkan dapat berharap dalam pengembangan selanjutnya.



BAB II

TINJAUAN PUSTAKA

A. Landasan Teori

1. Analisis Sentimen

Sentimen dapat diartikan sebagai pandangan atau opini yang didasarkan pada perasaan yang kuat terhadap sesuatu. Sentimen biasanya terdapat dalam pernyataan dan kalimat yang mengandung opini. Sentimen juga berguna untuk memahami perasaan seseorang terhadap topik atau objek tertentu.

Analisis sentimen merupakan proses untuk mengidentifikasi sentimen dan mengklasifikasikan polaritas teks dalam dokumen atau kalimat, sehingga dapat ditentukan apakah sentimen tersebut positif, negatif, atau netral. (Adityarini et al., 2021). Saat ini, analisis sentimen secara luas digunakan oleh para peneliti sebagai salah satu bidang penelitian dalam ilmu komputer. Jejaring sosial seperti Twitter sering digunakan dalam analisis sentimen untuk mengukur persepsi publik. (Samsir et al., 2021)

2. Obyek wisata

Obyek wisata memiliki arti yaitu tempat yang dikunjungi wisatawan karena memiliki daya tarik, baik alami maupun buatan manusia, seperti keindahan alam, pegunungan, pantai, flora dan fauna, kebun binatang, bangunan bersejarah, monumen, candi, tarian, atraksi, dan budaya khas lainnya. Sementara itu, objek wisata alam adalah objek wisata yang daya tarik utamanya berasal dari keindahan sumber daya alam dan tata lingkungannya. (R. Manalu & Fikri, 2021)

Wisata dapat di katakan aktivitas perjalanan, atau sebagian dari aktivitas tersebut, yang dilakukan secara sukarela dan bersifat sementara untuk menikmati objek dan daya tarik wisata. Dari definisi ini, dapat disimpulkan bahwa wisata merupakan kegiatan perjalanan yang dilakukan oleh seseorang atau sekelompok orang dengan mengunjungi tempat tertentu

untuk tujuan rekreasi, pengembangan pribadi, atau mempelajari keunikan daya tarik wisata dalam jangka waktu sementara. Sedangkan wisatawan adalah orang yang melakukan kegiatan wisata tersebut.(Akib, 2020)

3. *Word Embedding*

Word embedding memiliki makna sebagai model pembelajaran yang menghasilkan representasi kata dalam bentuk distribusi kontinu di ruang dimensi rendah. Secara umum, model pembelajaran yang digunakan adalah jaringan saraf tiruan (JST). Salah satu metode word embedding terkenal adalah Word2Vec, yang dikembangkan oleh Tomas Mikolov.

Word2Vec merupakan terobosan dalam ekstraksi fitur kata karena memanfaatkan semantik kata dari korpus. Dalam Word2Vec, setiap kata unik direpresentasikan oleh serangkaian angka yang disebut vektor.(Khomsah, 2021)

Banyak penelitian telah dilakukan dalam analisis sentimen. Dalam analisis sentimen, terdapat berbagai teknik word embedding yang dapat digunakan, seperti metode Word2vec, CCA, Word2Vec, Doc2vec, dan lainnya.(Subowo et al., 2021)

4. *Doc2vec*

Doc2Vec adalah pengembangan dari metode word embedding Word2Vec yang bertujuan untuk merepresentasikan dokumen dalam bentuk vektor. Doc2Vec dapat mengekstraksi fitur dengan menggunakan semua kata dalam dokumen, karena setiap kata digunakan dalam proses pembelajaran. Doc2Vec menghasilkan vektor dokumen dan vektor kata dari data pelatihan. Setiap dokumen dalam data pelatihan direpresentasikan dalam bentuk word set dan tag. Word set adalah kumpulan semua token dalam setiap dokumen, sementara tag adalah pengidentifikasi unik untuk setiap dokumen.(Widyaningtyas et al., 2019)

5. *Deep Learning*

Deep Learning di artikan sebagai salah satu cabang dari machine learning. Model deep learning mampu mempelajari komputasi secara mandiri menggunakan jaringan sarafnya. Deep learning dirancang untuk menganalisis data secara berkelanjutan, mirip dengan cara otak manusia membuat keputusan. Untuk meningkatkan kemampuannya, deep learning menggunakan algoritma artificial neural network (ANN), yang terinspirasi dari jaringan biologis otak manusia.(Peryanto et al., 2020).

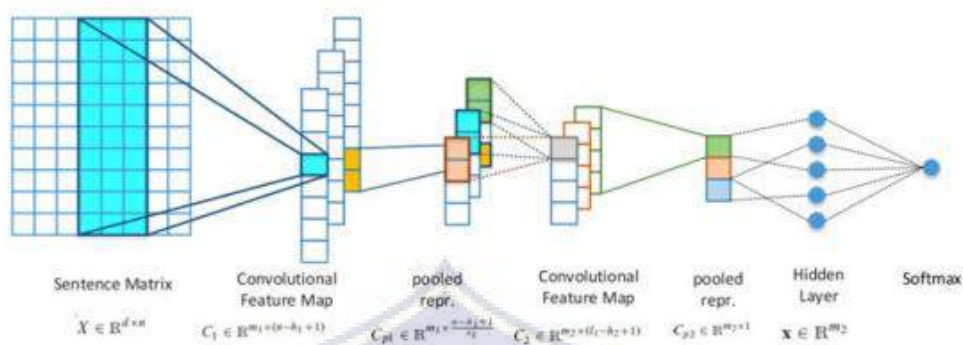
Deep Learning (DL) ialah teknik berbasis jaringan saraf tiruan yang telah banyak digunakan dalam beberapa tahun terakhir sebagai salah satu metode penerapan Machine Learning (ML). Beberapa artikel menyebutkan bahwa DL tidak terbatas pada bidang tertentu saja, tetapi telah didefinisikan sebagai bentuk pembelajaran umum yang mampu menyelesaikan berbagai macam masalah di berbagai bidang.(Diponegoro et al., 2021)

6. *Convolutional Neural Network*

CNN pertama kali dikembangkan oleh Kuniko Fukushima, seorang peneliti dari NHK Broadcasting Science Research Laboratories di Kinuta, Setagaya, Tokyo, Jepang, dengan nama awal NeoCognitron. Konsep CNN kemudian disempurnakan oleh Yann LeCun, seorang peneliti dari AT&T Bell Laboratories di Holmdel, New Jersey, Amerika Serikat. LeCun berhasil menerapkan model CNN yang dikenal sebagai LeNet dalam penelitiannya mengenai pengenalan angka dan tulisan tangan. Pada tahun 2012, Alex Krizhevsky mengaplikasikan model CNN-nya dan memenangkan kompetisi ImageNet Large Scale Visual Recognition Challenge 2012.(Tilasefana & Putra, 2023)

Convolutional Neural Network (CNN) salah satu jenis jaringan saraf tiruan yang banyak digunakan dalam tugas-tugas visi komputer. CNN adalah struktur matematika yang biasanya terdiri dari tiga jenis lapisan: convolution, pooling, dan lapisan yang sepenuhnya terhubung. Dua lapisan pertama, yaitu convolution dan pooling, melakukan ekstraksi fitur, sementara lapisan ketiga menggabungkan fitur yang diekstraksi untuk

menghasilkan keluaran akhir, seperti klasifikasi. (Hidayatullah & Nayoan, 2019)



Gambar 1. Arsitektur CNN

7. *Supervised Learning*

Supervised learning adalah salah satu cabang utama dari machine learning yang membangun fungsi atau model dari data pelatihan yang sudah dilabeli. Data pelatihan ini terdiri dari pasangan input dan output (label). Algoritma yang digunakan adalah Classification Algorithm dengan metode binary classification untuk menciptakan fungsi atau model yang dapat melakukan klasifikasi. (Kristiawan & Widjaja, 2021)

Pada algoritma Supervised Learning, sistem diberikan data pelatihan yang berisi informasi input dan output yang diinginkan. Dengan menggunakan data tersebut, sistem akan mempelajari pola dari data yang ada. Pola ini kemudian digunakan sebagai acuan untuk menganalisis kumpulan data selanjutnya. (R.H.Zer et al., 2022)

8. *Tensorflow*

TensorFlow dapat di katakana sebagai perpustakaan perangkat lunak yang dikembangkan oleh Tim Google Brain dari Google Research, yang bertujuan untuk pembelajaran mesin dan jaringan saraf dalam penelitian mereka. TensorFlow menggabungkan aljabar komputasi dengan teknik optimasi kompilasi, memfasilitasi perhitungan banyak ekspresi matematika. Fitur utama TensorFlow meliputi: (Economics et al., 2020)

1. Mendefinisikan, mengoptimalkan, dan menghitung ekspresi matematika yang melibatkan array multidimensi (tensors).
2. Mendukung pemrograman jaringan saraf dalam dan teknik machine learning.
3. Menggunakan GPU (Graphics Processing Unit) secara efisien, dengan otomatisasi manajemen dan optimalisasi memori terhadap data yang digunakan. TensorFlow memungkinkan penulisan kode yang sama untuk dijalankan di CPU atau GPU, serta mengidentifikasi bagian mana yang harus dipindahkan ke GPU.
4. Memiliki skalabilitas komputasi yang tinggi untuk keseluruhan mesin terhadap kumpulan data besar.

9. *Flowchart*

Flowchart menjelaskan alur logika dalam suatu masalah menggunakan simbol khusus dalam bentuk gambar, sedangkan pseudocode menggunakan kata-kata. Meskipun cara penyajiannya berbeda, keduanya bertujuan untuk membantu menjelaskan alur logika atau masalah guna memudahkan pembuatan program.

Flowchart dapat diartikan sebagai serangkaian langkah penyelesaian masalah yang dituliskan menggunakan simbol-simbol tertentu. Diagram alir ini akan menunjukkan alur logika dalam sebuah program. (Khesya, 2021)

10. *Scikit - Learn*

Scikit-learn sebagai salah satu modul Python yang mengintegrasikan berbagai algoritma pembelajaran mesin untuk masalah yang diawasi dan tidak diawasi dalam skala menengah. Modul ini sangat efisien untuk data mining dan analisis data. (Silitonga, 2019). Scikit-learn merupakan pustaka analisis data open source yang dianggap sebagai standar emas untuk Machine Learning (ML) dalam ekosistem Python. Pustaka ini mencakup berbagai metode algoritma data mining, termasuk klasifikasi, regresi, dan clustering. (D. A. Manalu & Gunadi, 2022)

Scikit-learn menawarkan berbagai metrik evaluasi dan teknik validasi silang untuk mengukur performa model dengan akurat, sehingga

membantu mencegah overfitting dan memastikan model yang dihasilkan dapat diandalkan.

B. Penelitian Terkait

1. NADIA RISTYA DEWI (2022)

Pada penelitian yang berjudul “PERBANDINGAN AKURASI WORD EMBEDDING TF-IDF DAN WORD2VEC MENGGUNAKAN RECURRENT NEURAL NETWORK UNTUK ANALISIS SENTIMEN TWEET VEKSINISASI COVID-19”, Penelitian ini bertujuan untuk mengetahui perbandingan akurasi antara dua metode word embedding yang populer, yaitu TF-IDF dan Word2Vec, ketika diterapkan pada algoritma Recurrent Neural Network (RNN) untuk analisis sentimen tweet tentang vaksinasi COVID-19. Dalam penelitian ini, peneliti menggunakan dataset yang terdiri dari 6490 tweet tentang vaksinasi COVID-19. Dataset ini dibagi dengan perbandingan 7:3 untuk data pelatihan (training) dan data pengujian (testing). Penelitian ini menguji dua pendekatan berbeda: RNN dengan Word2Vec sebagai metode word embedding dan RNN dengan TF-IDF sebagai metode word embedding. Hasil penelitian menunjukkan bahwa RNN dengan Word2Vec menghasilkan akurasi sebesar 51.71%, sedangkan RNN dengan TF-IDF menghasilkan akurasi yang sedikit lebih rendah yaitu 50.73%. Dari hasil ini, dapat disimpulkan bahwa penggunaan Word2Vec memberikan performa yang sedikit lebih baik dibandingkan dengan TF-IDF dalam konteks analisis sentimen tweet vaksinasi COVID-19. Penelitian ini memberikan wawasan bahwa meskipun perbedaan akurasi tidak signifikan, pemilihan metode word embedding yang tepat dapat memberikan dampak pada hasil analisis sentimen. Word2Vec, yang mengambil konteks kata dalam pembentukan vektornya, mungkin lebih mampu menangkap nuansa semantik yang lebih kaya dibandingkan dengan TF-IDF yang lebih berfokus pada frekuensi kata.

2. Dwi Intas Af'idah, Dairoh, Sharfina Febbi Handayani, Rizki Wijayatun Pratiwi (2021)

Pada Penelitian yang berjudul “Pengaruh Parameter Word2vec terhadap Performa Deep Learning pada Klasifikasi Sentimen”, Penelitian ini bertujuan untuk mengevaluasi pengaruh parameter Word2Vec terhadap performa model deep learning dalam klasifikasi sentimen. Word2Vec adalah metode populer untuk ekstraksi fitur kata dalam bentuk vektor yang digunakan dalam pra-pelatihan klasifikasi sentimen. Metode ini mampu menangkap makna semantik teks dengan merepresentasikan vektor yang mirip untuk setiap kata yang memiliki kedekatan makna. Parameter Word2Vec yang dievaluasi dalam penelitian ini meliputi arsitektur, metode evaluasi, dan dimensi. Hasil penelitian menunjukkan bahwa ketiga parameter tersebut memiliki pengaruh signifikan terhadap performa model deep learning dalam klasifikasi sentimen. Kombinasi parameter yang menghasilkan rata-rata akurasi tertinggi adalah arsitektur CBOW (Continuous Bag of Word), metode evaluasi Hierarchical Softmax, dan dimensi bernilai 100. Arsitektur CBOW memberikan performa yang lebih baik karena memiliki akurasi yang sedikit lebih tinggi untuk kata-kata yang sering muncul. Metode evaluasi Hierarchical Softmax menunjukkan hasil yang lebih baik karena menggunakan model pohon biner yang membuat kata-kata yang jarang muncul akan mewarisi representasi vektor di atasnya. Dimensi dengan nilai 100 menghasilkan akurasi yang lebih baik karena sesuai dengan jumlah dataset yang terdiri dari 10.000 ulasan.

3. Dedi Tri Hermanto, Arief Setyanto, Emha Taufiq Luthfi (2021)

Pada penelitian yang berjudul “Algoritma LSRM-CNN untuk Sentimen Klasifikasi dengan Word2vec Pada Media Online”. Penelitian ini mengeksplorasi penggunaan metode deep learning, khususnya Long Short-Term Memory (LSTM) dan Convolutional Neural Network (CNN), untuk klasifikasi sentimen pada judul berita berbahasa Indonesia yang diambil dari situs Detik Finance. Tujuan utama dari penelitian ini adalah untuk mengklasifikasikan judul berita berdasarkan sentimen positif dan negatif. Peneliti menggunakan Word2Vec untuk mengubah kata-kata dalam judul berita menjadi vektor yang kemudian dapat diproses oleh

model LSTM dan CNN. Word2Vec sangat berguna dalam konteks ini karena mampu menangkap makna semantik dari kata-kata dalam bahasa yang sangat kontekstual seperti Bahasa Indonesia.

Model LSTM digunakan karena kemampuannya yang baik dalam mengingat informasi jangka panjang, yang penting dalam memahami konteks dan sentimen dalam teks. Sementara itu, CNN digunakan untuk mengekstrak fitur lokal melalui operasi konvolusi yang dapat menangkap aspek penting dari struktur data teks. Dalam penelitian ini, peneliti melakukan eksperimen dengan tiga pendekatan berbeda: LSTM murni, LSTM diikuti oleh CNN (LSTM-CNN), dan CNN diikuti oleh LSTM (CNN-LSTM). Mereka menemukan bahwa kombinasi LSTM-CNN memberikan hasil yang lebih baik dibandingkan dengan LSTM murni atau CNN-LSTM, dengan akurasi sebesar 65%. Namun, pendekatan CNN-LSTM menunjukkan akurasi tertinggi yaitu 74%, menunjukkan bahwa urutan penerapan model ini memiliki dampak signifikan terhadap hasil klasifikasi sentimen. Penelitian ini memberikan wawasan baru tentang bagaimana kombinasi model deep learning dapat digunakan untuk analisis sentimen dalam Bahasa Indonesia, yang memiliki tantangan tersendiri karena struktur dan konteks bahasa yang kompleks. Hasil dari penelitian ini dapat diaplikasikan dalam berbagai bidang, seperti pemantauan opini publik, analisis pasar, dan pemahaman terhadap reaksi masyarakat terhadap berita ekonomi.

4. Muhammad Fikri Heldiansyah (2021)

Pada penelitian yang berjudul “DETEKSI EMOSI PADA TWEET DENGAN MENGGABUNGKAN CONTEXTUALIZED WORD EMBEDDING DAN CONVOLUTIONAL NEURAL NETWORK (CNN)”, Penelitian ini berfokus pada deteksi emosi dalam tweet berbahasa Indonesia dengan menggunakan kombinasi model Convolutional Neural Network (CNN) dan word embedding kontekstual seperti BERT dan ELMo, serta word embedding tradisional seperti Word2Vec. Tujuan utama dari penelitian ini adalah untuk mengidentifikasi lima emosi yang berbeda

dalam tweet, yaitu marah (anger), cinta (love), takut (fear), bahagia (happy), dan sedih (sadness). Pendekatan ini penting karena emosi yang diekspresikan dalam media sosial seperti Twitter dapat memberikan wawasan yang berharga tentang persepsi dan reaksi publik terhadap berbagai topik dan peristiwa. Dalam penelitian ini, model BERT-CNN menunjukkan hasil terbaik dengan nilai macro-averaged precision sebesar 75,40, macro-average recall sebesar 71,62, dan macro-averaged f1-score sebesar 72,83. Ini menunjukkan bahwa kombinasi BERT, yang merupakan model word embedding kontekstual, dengan CNN dapat meningkatkan kemampuan deteksi emosi pada tweet berbahasa Indonesia. Penelitian ini juga menemukan bahwa model Word2Vec-CNN dan BERT-CNN tidak menunjukkan peningkatan performa pada data yang telah di-stemming, sedangkan model ELMo-CNN menunjukkan peningkatan pada macro-averaged f1-score pada data dengan stemming. Hal ini menunjukkan bahwa pemrosesan awal data dan pemilihan model word embedding memiliki pengaruh signifikan terhadap hasil deteksi emosi. Hasil dari penelitian ini dapat diaplikasikan dalam berbagai bidang, termasuk pemasaran digital, manajemen krisis, dan analisis opini publik. Dengan memahami emosi yang diekspresikan dalam tweet, organisasi dan perusahaan dapat merespons dengan lebih efektif terhadap kebutuhan dan kekhawatiran masyarakat. Penelitian ini memberikan kontribusi penting dalam pengembangan teknologi analisis sentimen dan deteksi emosi, khususnya dalam konteks bahasa Indonesia, yang memiliki tantangan tersendiri karena kekayaan dan keragaman bahasa.

5. Putri Rizki Amalia(2021)

Pada penelitian yang berjudul “ANALISIS SENTIMEN BERDASARKAN ASPEK PADA ULASAN RESTORAN BERBAHASA INDONESIA MENGGUNAKAN CONVOLUTIONAL NEURAL NETWORK (CNN) DAN CONTEXTUALIZED WORD EMBEDDING”, Penelitian ini bertujuan untuk membandingkan kinerja dari tiga metode word embedding yang populer: Word2Vec, GloVe, dan

FastText dalam konteks klasifikasi teks menggunakan algoritma Convolutional Neural Network (CNN). Dalam penelitian ini, ketiga metode word embedding tersebut dipilih karena kemampuan mereka untuk menangkap makna semantik, sintatik, dan konteks di sekitar kata, yang merupakan aspek penting dalam pemrosesan teks. Metode ini dianggap lebih unggul dibandingkan dengan feature engineering tradisional seperti Bag of Words, yang hanya merepresentasikan teks dalam bentuk frekuensi kemunculan kata tanpa mempertimbangkan konteksnya. Peneliti melakukan eksperimen pada dua dataset berita yang berbeda: 20 Newsgroup dan Reuters Newswire. Kinerja dari masing-masing metode word embedding diukur menggunakan metrik F-measure. Hasil penelitian menunjukkan bahwa FastText memberikan performa terbaik dengan nilai F-Measure sebesar 0.979 untuk dataset 20 Newsgroup dan 0.715 untuk Reuters. Meskipun demikian, perbedaan kinerja antara ketiga metode word embedding tidak signifikan, yang menunjukkan bahwa ketiganya memiliki kinerja yang kompetitif.

6. Hans Juwianto, Ester Irawati Setiawan, Joan Santoso, Mauridhi Hery Purnomo (2020)

Pada penelitian yang berjudul “SENTIMEN ANALYSIS TWITTER BERBAHASA INDONESIA BERBASIS WORD2VEC MENGGUNAKAN DEEP CONVOLUTIONAL NEURAL NETWORK”, pada penelitian ini, model Word2Vec digunakan untuk mengubah kata-kata dalam tweet menjadi vektor numerik yang dapat diproses oleh komputer. Model ini penting karena memungkinkan mesin untuk memahami konteks kata dalam kalimat, yang sangat berguna dalam analisis sentimen. Word2Vec juga membantu mempercepat proses pelatihan dan meningkatkan akurasi model. Algoritme Deep Convolutional Neural Network (CNN) dipilih karena kemampuannya yang unggul dalam mengenali pola dalam data teks. CNN melakukan operasi konvolusi menggunakan filter yang disesuaikan dengan ukuran jendela kata untuk mengekstrak fitur penting dari urutan kata dalam tweet.

Peneliti menggunakan dataset yang terdiri dari 999 tweet Bahasa Indonesia. Dengan menggunakan kombinasi Word2Vec dan CNN, mereka berhasil mencapai akurasi terbaik sebesar 76,40%. Ini menunjukkan bahwa model yang mereka kembangkan efektif dalam mengklasifikasikan sentimen tweet sebagai positif atau negatif. Penelitian ini memberikan kontribusi penting dalam bidang analisis sentimen, terutama dalam konteks Bahasa Indonesia, dan menawarkan wawasan baru tentang bagaimana model pembelajaran mendalam dapat digunakan untuk memahami opini publik di media sosial. Hasil dari penelitian ini dapat digunakan oleh perusahaan untuk memantau dan meningkatkan citra merek mereka berdasarkan umpan balik pelanggan yang dinyatakan melalui media social.

7. Aliyanti Nurdin, Bernadus Anggo Seno Aji, Anugrayani Bustamin, Zaenal Abidin (2020)

Pada penelitian yang berjudul “PERBANDINGAN KINERJA WORD EMBEDDING WORD2VEC, GLOVE DAN FASTTEXT PADA KLASIFIKASI TEKS”, Penelitian ini bertujuan untuk membandingkan kinerja dari tiga metode word embedding yang populer: Word2Vec, GloVe, dan FastText dalam konteks klasifikasi teks menggunakan algoritma Convolutional Neural Network (CNN). Dalam penelitian ini, ketiga metode word embedding tersebut dipilih karena kemampuan mereka untuk menangkap makna semantik, sintatik, dan konteks di sekitar kata, yang merupakan aspek penting dalam pemrosesan teks. Metode ini dianggap lebih unggul dibandingkan dengan feature engineering tradisional seperti Bag of Words, yang hanya merepresentasikan teks dalam bentuk frekuensi kemunculan kata tanpa mempertimbangkan konteksnya. Peneliti melakukan eksperimen pada dua dataset berita yang berbeda: 20 Newsgroup dan Reuters Newswire. Kinerja dari masing-masing metode word embedding diukur menggunakan metrik F-measure. Hasil penelitian menunjukkan bahwa FastText memberikan performa terbaik dengan nilai F-Measure sebesar 0.979 untuk dataset 20 Newsgroup dan 0.715 untuk

Reuters. Meskipun demikian, perbedaan kinerja antara ketiga metode word embedding tidak signifikan, yang menunjukkan bahwa ketiganya memiliki kinerja yang kompetitif.

C. Kerangka Berpikir



Gambar 2. Kerangka berpikir

BAB III

METODE PENELITIAN

A. Tempat dan Waktu Penelitian

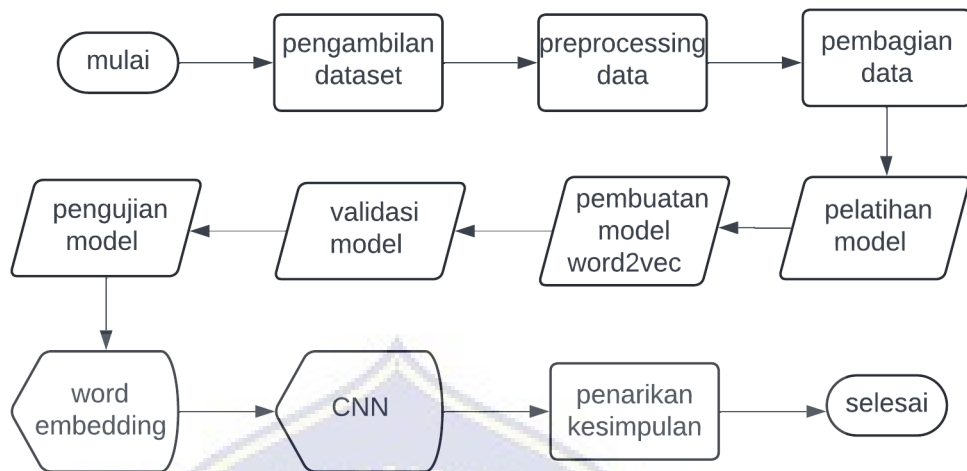
Penelitian ini dilakukan secara online dengan mengumpulkan dataset dari beberapa ulasan wisata yang tersedia di Google Maps. Penelitian ini dilaksanakan mulai Januari 2024 dan akan berlangsung hingga seluruh proses pengumpulan data selesai.

B. Alat dan Bahan

1. Kebutuhan Hardware
 - a. Laptop ASUS E402Y
2. Kebutuhan Software
 - a. Visual Studio Code
 - b. Excel
 - c. Python
 - d. Scikit-learn dan TensorFlow (digunakan untuk mengimplementasikan model machine learning dan deep learning.)

C. Perancangan Sistem

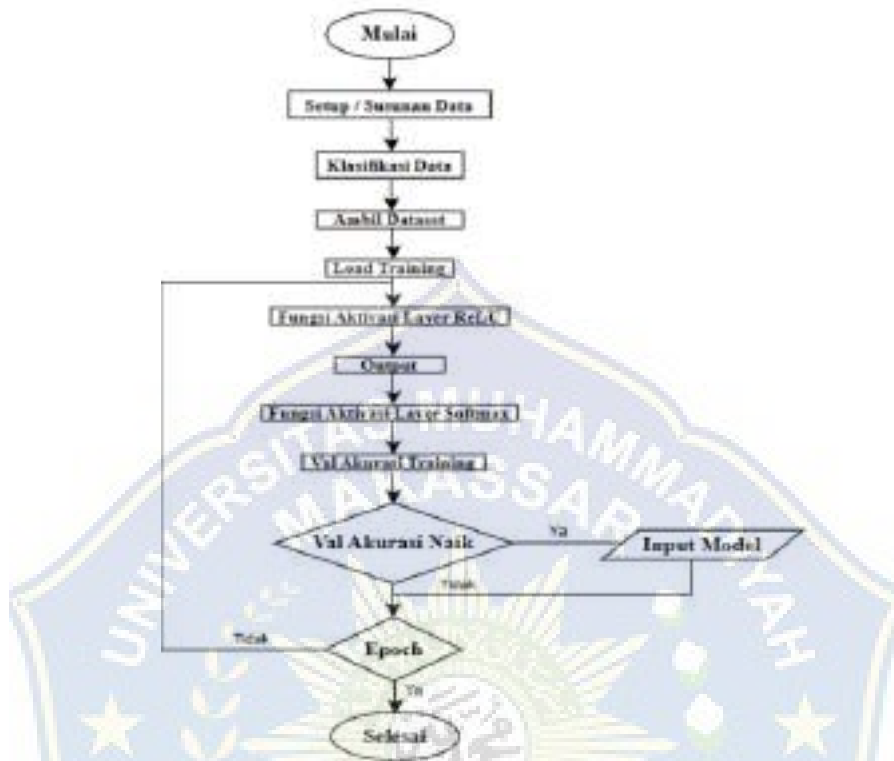
Perancangan sistem merupakan aspek krusial dalam pembangunan suatu sistem karena menjelaskan bagaimana sistem tersebut dikembangkan dari tahap perencanaan hingga pembuatan fungsi-fungsi yang dibutuhkan untuk pengoperasian. Tujuan utama dari perancangan sistem adalah untuk menentukan apakah sistem yang akan dikembangkan dapat menghasilkan hasil yang diinginkan.



Gambar 3. Perancangan sistem

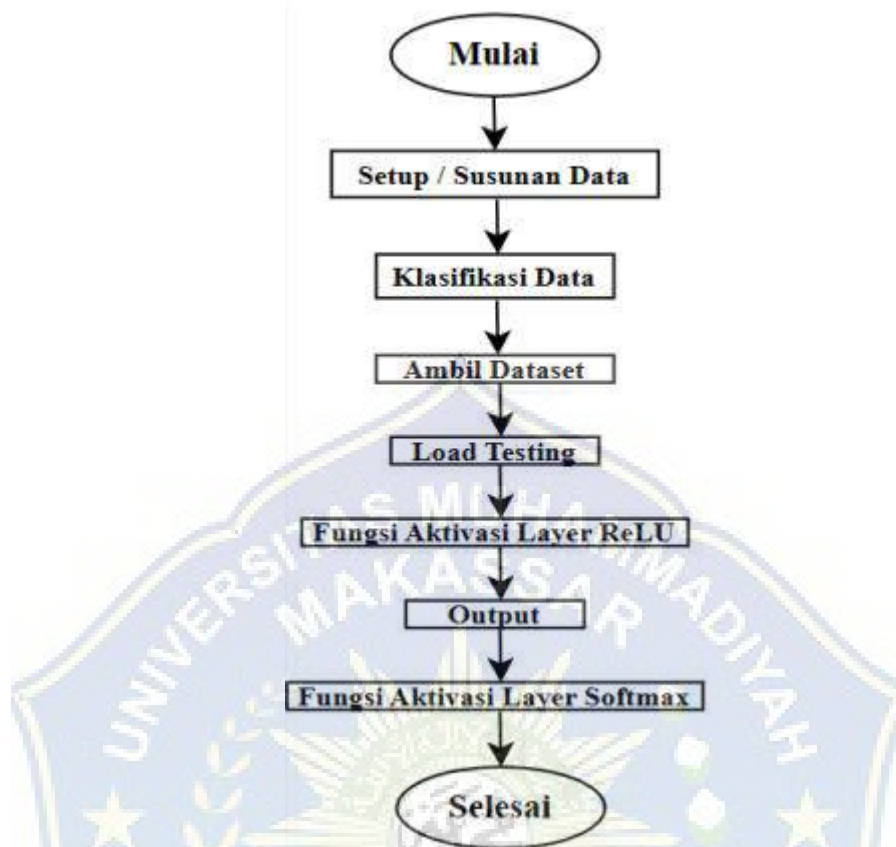
Dalam diagram di atas dapat di jelaskan bahwa untuk mengevaluasi bagaimana Word2Vec dapat meningkatkan kinerja model CNN dalam analisis sentimen terhadap ulasan tempat wisata di Makassar. Ruang lingkup penelitian dibatasi pada ulasan tempat wisata yang tersedia di Google Maps. Selanjutnya, data ulasan dikumpulkan dari Google Maps menggunakan teknik web scraping, diikuti oleh proses pra-pemrosesan seperti pembersihan, tokenisasi, dan normalisasi teks. Setelah data siap, model Word2Vec dilatih pada korpus data ulasan untuk menghasilkan representasi vektor kata. Vektor ini kemudian digunakan sebagai input untuk model CNN yang dirancang. Implementasi model dilakukan menggunakan framework seperti TensorFlow, di mana vektor Word2Vec diintegrasikan sebagai input ke dalam model CNN. Pelatihan model dilakukan dengan membagi data ulasan menjadi set pelatihan, validasi, dan pengujian. Model CNN dilatih dengan data pelatihan dan dievaluasi menggunakan metrik seperti akurasi, precision, recall, dan F1-score. Hasil analisis menunjukkan kinerja model dalam mengklasifikasikan sentimen ulasan. Setelah proses pelatihan dan evaluasi, setiap langkah penelitian didokumentasikan dengan baik, diikuti oleh penyusunan laporan skripsi yang mencakup latar belakang, metodologi, hasil, dan kesimpulan.

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



Gambar 4. Perancangan sistem training

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, akurasi data pelatihan dievaluasi secara berkala, dan jika akurasi 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.



Gambar 5. Perancangan sistem testing

Sistem yang dirancang memiliki dua alur utama: perancangan sistem pelatihan dan perancangan sistem pengujian. Pada perancangan sistem pelatihan, fokus utamanya adalah memantau kurva akurasi 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.

D. Teknik Pengujian Sistem

Teknik pengujian sistem yang akan digunakan dalam pengujian ini melibatkan pemisahan data menjadi dua bagian, yaitu data training dan data testing. Langkah ini diambil untuk memastikan bahwa model yang dikembangkan dapat secara efektif mempelajari pola dari data pelatihan dan kemudian menerapkan pengetahuan tersebut pada data yang belum pernah

dilihat sebelumnya, yaitu data pengujian. Dengan demikian, teknik pengujian ini memungkinkan evaluasi yang akurat terhadap kinerja model dalam mengklasifikasikan sentimen dari teks ulasan tempat wisata di Makassar.

Teknik ini bertujuan untuk menguji keakuratan dan efektivitas metode Convolutional Neural Network (CNN) dalam menganalisis sentimen teks yang berkaitan dengan tempat wisata di Makassar. Pengujian melibatkan pengumpulan data teks tentang tempat wisata, pelabelan sentimen (positif, negatif, atau netral), dan pengujian model CNN untuk menilai seberapa baik model dapat mengklasifikasikan sentimen tersebut.

Pengujian akurasi bertujuan untuk mengukur keberhasilan model dalam mengklasifikasikan sentimen dengan tepat. Akurasi 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.

Rumus untuk perhitungan confusion matrix untuk menghitung precision, recall, dan nilai accuracy dapat dijelaskan di bawah ini (Dinata et al., 2020):

a. Precision

Bermanfaat untuk mengukur sejauh mana ketepatan antara informasi yang diminta oleh pengguna dengan jawaban yang diberikan oleh sistem

$$\text{Precision} = \frac{\text{TP}}{(\text{TP} + \text{FP})} \times 100$$

b. Recall

Berguna untuk mengukur tingkat keberhasilan sistem dalam menemukan kembali sebuah informasi, pada persamaan berikut :

$$\text{Recall} = \frac{\text{TP}}{(\text{TP} + \text{FN})} \times 100$$

c. *Accuracy*

Berguna untuk mengukur suatu kinerja sebuah metode

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{(\text{TP} + \text{TN} + \text{FP} + \text{FN})} \times 100$$

Keterangan :

TP = True positif

TN= True negatif

FP = False positif

FN= False negative

E. Teknik Analisi Data

Analisis data adalah metode yang digunakan untuk memahami cara mendeskripsikan data, hubungan antar data, semantik data, serta batasan data dalam suatu sistem informasi (Pelham, 2023). Proses analisis data dilakukan pada penelitian ini adalah sebagai berikut :

1. Reduksi Data (*Data Reduction*)

Reduksi data adalah langkah penting dalam penanganan data lapangan yang berjumlah sangat besar. Proses ini mengharuskan peneliti mencatat setiap detail dengan cermat. Tujuan reduksi data adalah untuk memberikan gambaran yang lebih jelas dan mempermudah peneliti dalam pengumpulan dan pencarian data lebih lanjut sesuai kebutuhan penelitian. Fokus utama peneliti kualitatif adalah pada hasil, sehingga mereduksi data menjadi kunci untuk mencapai pemahaman yang mendalam. Namun, peneliti harus berhati-hati saat menghadapi outliers, data yang tidak diketahui, dan pola yang tidak teratur.

2. Penyajian Data (*Display Data*)

Hasil produksi akan ditampilkan secara eksklusif untuk setiap pola, kategori, fokus, dan tema yang ingin dipahami oleh peneliti. Penggunaan

display data membantu peneliti melihat gambaran keseluruhan atau bagian tertentu berdasarkan output penelitian. Dalam penelitian kualitatif, penyajian data dapat berupa uraian singkat, bagan, interaksi antar kategori, dan sejenisnya. Deskripsi tekstual sering digunakan untuk menyajikan data dalam konteks penelitian kualitatif.

3. Penarikan Kesimpulan (*Concluding Drawing Verivication*)

Langkah ketiga dalam analisis data adalah penarikan kesimpulan dan verifikasi. Kesimpulan awal yang diajukan bersifat sementara dan dapat berubah jika diperlukan bukti tambahan dari pengumpulan data berikutnya. Oleh karena itu, kesimpulan dalam penelitian dapat menjawab rumusan masalah yang telah ditetapkan sejak awal. Hal ini disebabkan oleh sifat sementara dari perumusan masalah dalam penelitian kualitatif, yang akan berkembang setelah melakukan penelitian di lapangan.



BAB IV

HASIL DAN PEMBAHASAN

A. Pengambilan Data

Data ulasan diambil menggunakan Instant Data Scraper, yang mengumpulkan ulasan dari Google Maps untuk beberapa objek wisata, yaitu Pantai Akkarena, Wisata Kebun, Tanjung Bayang, Pantai Bosowa, dan Bugis Waterpark Adventure. Jumlah total ulasan yang berhasil dikumpulkan dari kelima tempat wisata ini adalah 4500. Berikut proses pengambilan data di google maps :



Gambar 6. Proses pengambilan data ulasan dengan Instant Data Scraper

1. Data Ulasan

Setelah mengumpulkan data ulasan dari Google Maps untuk tempat wisata menggunakan Instant Data, hasilnya disimpan dalam format Excel. Berikut adalah hasil pengambilan dataset ulasan dan kemudian di simpan dalam format excel :

1. Tempatnya sejuk. Lebih cocok untuk rekreasi anak-anak. Tiket weekdays 15.000 / orang. Tempat duduk banyak. Ada pemancingan, wahana bermain anak, sewa sepeda listrik atau ATV, juga kolam renang
2. Banyak pohon-pohon yang berbuah juga
3. Kebun wisata yang sangat dekat dari jalan besar sehingga mudah dijangkau berbagai jenis kendaraan, dengan perjalanan sekitar 25-35 menit dari kota Makassar. Tiket pada hari biasa 15 k, dan pada Sabtu-Minggu 25 k.
4. Pas masuk cukup sejuk karena musim hujan, disambut kolam dan penginapan. Ada 3 kolam besar yang terbagi dengan sekat, kolam dalam, sedang dan untuk anak-anak. Di lokasi ada tempat beli cemilan makanan ringan dan sewa alat renang.
5. Rekomended juga krna banyak disediakan tempat duduk dan berteduh di pinggir kolam. Di sini juga disediakan aula. Sangat cocok buat liburan bareng keluarga
6. Karena sekolah anak sy adakan outbond di wisata kebun jd setelah sekian purnama akhirnya bisa lg ke wisata kebun rame² bareng dgn orgtua siswa yg lainnya,, pas masuk ke area parkir kesan pertama ya , area parkirnya luas , trs ke bagian loket u/ beli karcis masuk , karyawannya ramah bnget .
7. Begitu udah masuk ke dlm area bener² kereeen banget , semua tempat dlm area wiskeb cucok banget buat foto² , pokoknya memori hp bakalan full 🥰
8. Anak sy betah dgn kegiatan outbondnya setelah outbond berenang deh,,
9. Di wisata kebun itu harga tiket terjangkau, fasilitas oke , area bermain anak banyak , kantin jg ada , kolam pancing ada , gasebo² gratis bersih juga 👍👍👍👍
10. Tiket masuknya perorang Rp.15.000, ada kebun durian, rambutan. Yang mau berenang juga disediakan 4 kolam renang untuk anak-anak & dewasa, ada penginapan, kantin, tempat mancing, banyak spot foto-foto yang bagus juga. Kemarin nyobain kereta keliling wisata kebun 2x hanya Rp.5.000
11. T4 ini bagus buat berwisata keluarga nyaman adem fasilitas bagus tersedia aneka permainan utk anak2
12. Salah satu tempat wisata yang nyaman untuk didatangi, ada banyak yang bisa dieksplor di sini, bahkan disediakan banyak tempat untuk istirahat.
13. Saran tolong di pisahkan kolam yg bs digunakan khusus berenang dan memakai ban apalagi ban besar yg peruntukanx buat wahana seluncuran.
14. Tempat wisata keluarga yang cukup ramai waktu berkunjung di awal pekan. Fasilitas yang tersedia kolam renang, tempat makan, aula pertemuan dll.
15. Aman,nyaman,bersih,ada tmpat nginap,kolam renangnya banyak,pengawasan untuk anak² yg ikut berenang juga ada 👍👍

Gambar 7. Dataset ulasan

B. Pelabelan Data

Pelabelan sentimen dilakukan secara manual untuk mengidentifikasi pola dan karakteristik dalam teks yang menunjukkan sentimen positif, negatif, atau netral. Ulasan yang dikumpulkan dari Google Maps disimpan

dalam atribut ulasan, sedangkan nilai klasifikasi seperti positif, negatif, atau netral disimpan dalam atribut label. Tabel pelabelan data dapat dilihat di bawah ini:

Tabel 1. Tahap pelabelan data

ULASAN	LABEL
Bersih rapih lengkap murah menyenangkan untuk untuk acara liburan keluarga,,,	Positif
Tidak ada pemandangan yang bagus. Karcis masuk juga mahal. Pokoknya jelek sejelek-jeleknya.	Negatif
Datang dan rasakan sendiri sensasi yang belum pernah Anda dapatkan sebelumnya.	Netral

C. Reprocessing Data

Reprocessing data adalah proses pengolahan ulang data yang sudah ada untuk tujuan tertentu. Hal ini dapat mencakup berbagai langkah seperti membersihkan, mengubah format, memperbarui, atau menganalisis data dengan cara baru. Berikut beberapa tahap-tahp untuk reprocessing data:

1. Cleaning

Langkah pertama dalam preprocessing data adalah data cleaning atau membersihkan data. Ini berarti data mentah yang sudah dikumpulkan harus diseleksi ulang. Langkah ini penting karena karakter-karakter tertentu umumnya tidak memberikan kontribusi signifikan dalam analisis data atau pemrosesan teks. Misalnya, dalam analisis teks, tanda baca sering dianggap sebagai noise yang tidak relevan dan dihilangkan untuk memfokuskan pada informasi utama yang ada dalam teks tersebut.

Tabel 2. Tahap cleaning atau pembersihan

SEBELUM	SESUDAH
Bersih rapih lengkap murah menyenangkan untuk untuk acara liburan keluarga,,,	Bersih rapih lengkap murah menyenangkan untuk untuk acara liburan keluarga
Tidak ada pemandangan yang bagus. Karcis masuk juga mahal. Pokoknya jelek sejelek-jeleknya.	Tidak ada pemandangan yang bagus Karcis masuk juga mahal Pokoknya jelek sejelekjeleknya
Datang dan rasakan sendiri sensasi yang belum pernah Anda dapatkan sebelumnya.	Datang dan rasakan sendiri sensasi yang belum pernah Anda dapatkan sebelumnya

2. *Transform Case*

Transform case adalah proses mengubah huruf dalam teks dari satu bentuk ke bentuk lainnya, seperti mengubah semua huruf menjadi huruf besar (*uppercase*) atau huruf kecil (*lowercase*).

Tabel 3. Tahap Transform Case

SEBELUM	SESUDAH
Bersih rapih lengkap murah menyenangkan untuk untuk acara liburan keluarga,,,	bersih rapih lengkap murah menyenangkan untuk untuk acara liburan keluarga
Tidak ada pemandangan yang bagus. Karcis masuk juga mahal. Pokoknya jelek sejelek-jeleknya.	tidak ada pemandangan yang bagus. karcis masuk juga mahal pokoknya jelek sejelek jeleknya
Datang dan rasakan sendiri sensasi yang belum pernah Anda dapatkan sebelumnya.	datang dan rasakan sendiri sensasi yang belum pernah anda dapatkan sebelumnya

3. *Tokenizing*

Tokenizing adalah proses dalam pemrosesan bahasa alami (NLP) di mana teks dipecah menjadi unit-unit yang lebih kecil, yang disebut token. Token bisa berupa kata, frasa, karakter, atau simbol, tergantung pada tujuan analisis.

Tabel 4. Tahap Tokenizing

SEBELUM	SESUDAH
Bersih rapih lengkap murah menyenangkan untuk untuk acara liburan keluarga,,,	['bersih', 'rapih', 'lengkap', 'murah', 'menyenangkan', 'untuk', 'untuk', 'acara', 'liburan', 'keluarga']
Tidak ada pemandangan yang bagus. Karcis masuk juga mahal. Pokoknya jelek sejelek-jeleknya.	['tidak', 'ada', 'pemandangan', 'yang', 'bagus', 'karcis', 'masuk', 'juga', 'mahal', 'pokoknya', 'jelek', 'sejelekjeleknya']
Datang dan rasakan sendiri sensasi yang belum pernah Anda dapatkan sebelumnya.	['datang', 'dan', 'rasakan', 'sendiri', 'sensasi', 'yang', 'belum', 'pernah', 'anda', 'dapatkan', 'sebelumnya']

D. Penerapan Metode

```
import pandas as pd
import numpy as np
from tqdm import tqdm
from keras.preprocessing.text import Tokenizer
tqdm.pandas(desc="progress-bar")
from gensim.models import Doc2Vec
from sklearn import utils
from sklearn.model_selection import train_test_split
from keras.preprocessing.sequence import pad_sequences
import gensim
from sklearn.linear_model import LogisticRegression
from gensim.models.doc2vec import TaggedDocument
import re
import seaborn as sns
import matplotlib.pyplot as plt
```

Kodingan yang diberikan mengimpor berbagai pustaka yang diperlukan untuk pemrosesan data, pemodelan, dan visualisasi dalam analisis data. Pertama, `pandas` dan `numpy` diimpor untuk manipulasi data dan operasi numerik, masing-masing. `pandas` sangat berguna untuk bekerja dengan data berformat tabel, sedangkan `numpy` memungkinkan operasi efisien pada array dan matriks multidimensi.

Selanjutnya, pustaka `tqdm` diimpor untuk menampilkan progress bar, yang membantu dalam melacak kemajuan loop yang berjalan lama. Integrasi `tqdm` dengan `pandas` memungkinkan progress bar ditampilkan selama operasi `pandas`. Kemudian, `Tokenizer` dari Keras diimpor untuk memproses teks, memisahkan teks menjadi token, dan mengubahnya menjadi format yang dapat digunakan oleh model pembelajaran mesin.

Selain itu, model `Doc2Vec` dari Gensim diimpor bersama dengan `TaggedDocument` untuk menghasilkan representasi vektor dari dokumen, yang penting untuk berbagai aplikasi NLP. Modul `utils` dari scikit-learn menyediakan berbagai fungsi utilitas, sementara `train_test_split` digunakan untuk membagi dataset menjadi set pelatihan dan pengujian. Fungsi `pad_sequences` dari Keras memastikan bahwa semua urutan teks memiliki panjang yang sama dengan menambahkan padding. Untuk pemodelan, `LogisticRegression` dari scikit-learn diimpor untuk membangun model klasifikasi. Modul `re` digunakan untuk menangani ekspresi reguler, yang berguna dalam pencarian dan manipulasi teks. Terakhir, pustaka `seaborn` dan `matplotlib` diimpor untuk membuat visualisasi data, dengan `seaborn` menyediakan antarmuka yang lebih mudah digunakan untuk visualisasi statistik yang kompleks di atas `matplotlib`.

Dengan mengimpor dan menggunakan pustaka ini, kodingan tersebut mempersiapkan lingkungan untuk melakukan pemrosesan teks, pembuatan model, evaluasi, dan visualisasi data secara efektif.


```
df =
pd.read_excel('/content/Preprocessing.xlsx', sheet_name="Sheet1")
df = df[['ULASAN', 'LABEL']]
df = df[pd.notnull(df['ULASAN'])]
df.rename(columns={'ULASAN': 'ULASAN'}, inplace=True)
```

Kode tersebut memuat dan memproses data dari file Excel menggunakan pustaka 'pandas'. Pertama, data diimpor dari file Excel bernama "Preprocessing.xlsx" yang berada di direktori '/content/' dan diambil dari sheet bernama "Sheet1". Data yang diambil kemudian disimpan dalam sebuah dataframe 'df'. Hanya kolom "ULASAN" dan "LABEL" yang dipilih untuk digunakan dari dataframe tersebut. Langkah berikutnya, baris-baris yang memiliki nilai null pada kolom "ULASAN" dihapus untuk memastikan tidak ada data yang hilang dalam analisis selanjutnya. Terakhir, kolom "ULASAN" diberi nama ulang meskipun namanya tetap sama, kemungkinan sebagai langkah untuk memastikan konsistensi atau mempermudah pemrosesan lebih lanjut.

```
df.index = range(len(df))
total_words = df['ULASAN'].apply(lambda x: len(x.split('
'))).sum()
print("Total jumlah kata dalam semua ulasan:", total_words)
```

Melanjutkan pemrosesan data pada dataframe `df` yang sebelumnya telah diimpor dan diproses. Pertama, indeks dataframe diatur ulang sehingga menjadi urutan bilangan bulat dari 0 hingga panjang dataframe `(len(df))`, memastikan bahwa indeksnya berurutan dan konsisten. Langkah berikutnya adalah menghitung total jumlah kata dalam kolom "ULASAN". Ini dilakukan dengan menerapkan fungsi `lambda` pada setiap entri di kolom "ULASAN", yang membagi teks ulasan menjadi kata-kata terpisah (berdasarkan spasi) dan menghitung jumlah kata dalam setiap ulasan. Hasil dari semua penghitungan ini kemudian dijumlahkan menggunakan metode `sum()`. Akhirnya, total jumlah kata dalam semua ulasan dicetak dengan menggunakan pernyataan

`print`, memberikan informasi berapa banyak kata yang ada dalam semua ulasan di dataframe.

```
cnt_pro = df['LABEL'].value_counts()

# Menggambar diagram batang menggunakan Seaborn
plt.figure(figsize=(12, 4))
sns.barplot(x=cnt_pro.index, y=cnt_pro.values, alpha=0.8)
plt.ylabel('Jumlah Kemunculan', fontsize=12)
plt.xlabel('LABEL', fontsize=12)
plt.xticks(rotation=90)
plt.show()
```

Kode di atas melakukan visualisasi distribusi nilai dalam kolom "LABEL" pada dataframe `df` menggunakan plot batang. Pertama, kode menghitung jumlah kemunculan setiap nilai unik dalam kolom "LABEL" dengan menggunakan fungsi `value_counts()`, yang menghasilkan sebuah Series di mana indeksnya adalah nilai-nilai unik dari kolom tersebut dan nilainya adalah jumlah kemunculan dari setiap nilai. Data hasil perhitungan ini kemudian disimpan dalam variabel `cnt_pro`.

Selanjutnya, sebuah figure baru dibuat dengan ukuran 12x4 inci menggunakan `matplotlib.pyplot` untuk mempersiapkan kanvas tempat plot akan ditampilkan. Kemudian, `seaborn` digunakan untuk membuat plot batang, dengan parameter `x` diisi dengan indeks dari `cnt_pro` (nilai-nilai unik dari kolom "LABEL") dan `y` diisi dengan nilai-nilai dari `cnt_pro` (jumlah kemunculan dari setiap label). Transparansi batang diatur dengan `alpha=0.8`.

Label untuk sumbu y ditambahkan dengan teks "Jumlah Kemunculan" dan ukuran font 12, dan label untuk sumbu x ditambahkan dengan teks "LABEL" dan ukuran font 12. Teks pada sumbu x diputar sebesar 90 derajat untuk memastikan keterbacaan jika teks tersebut panjang. Akhirnya, plot tersebut ditampilkan dengan `plt.show()`. Plot ini membantu dalam memahami distribusi data label dalam dataset dengan cara yang mudah dibaca dan informatif.

```
def print_message(index):
    example = df.iloc[index][['ULASAN', 'LABEL']].values
    if len(example) > 0:
        print('ULASAN:', example[0])
        print('LABEL:', example[1])

# Menggunakan fungsi print_message() dengan indeks
tertentu
print_message(12)
```

Kode di atas mendefinisikan sebuah fungsi bernama `print_message` yang digunakan untuk mencetak ulasan dan label dari dataframe `df` berdasarkan indeks yang diberikan. Fungsi ini mengambil satu parameter, yaitu `index`, yang menentukan baris mana dari dataframe yang akan diambil datanya. Di dalam fungsi, baris dataframe pada indeks yang ditentukan diambil menggunakan `df.iloc[index]`, dan hanya kolom "ULASAN" dan "LABEL" yang dipilih. Nilai dari kolom-kolom tersebut kemudian disimpan dalam variabel `example` dalam bentuk array.

Jika panjang array `example` lebih dari 0 (artinya data ulasan dan label ada dan valid), maka fungsi akan mencetak ulasan dan label tersebut. `example[0]` mengacu pada nilai dari kolom "ULASAN" dan `example[1]` mengacu pada nilai dari kolom "LABEL". Selanjutnya, fungsi ini digunakan untuk mencetak ulasan dan label pada baris ke-12 dari dataframe `df` dengan memanggil `print_message(12)`. Dengan demikian, kode ini memudahkan untuk melihat secara spesifik ulasan dan label dari baris tertentu dalam dataframe.

```
import string
def remove_punctuation(text):
    return text.translate(str.maketrans('', '',
string.punctuation))

# Menghapus tanda baca dari kolom ULASAN
df['ULASAN'] = df['ULASAN'].apply(remove_punctuation)
```

Mendefinisikan sebuah fungsi bernama `remove_punctuation` yang digunakan untuk menghapus tanda baca dari teks. Fungsi ini menggunakan

modul `string` dari Python, khususnya `string.punctuation`, yang berisi semua karakter tanda baca standar. Fungsi `remove_punctuation` menerima parameter `text` yang merupakan teks yang akan diproses. Di dalam fungsi, metode `translate` digunakan bersama dengan `str.maketrans` untuk membuat peta terjemahan yang menghapus semua karakter tanda baca dari teks.

Setelah mendefinisikan fungsi, kode ini mengaplikasikannya pada kolom "ULASAN" dalam dataframe `df`. Hal ini dilakukan dengan menggunakan metode `apply` dari pandas, yang menerapkan fungsi `remove_punctuation` pada setiap entri dalam kolom "ULASAN". Sebagai hasilnya, semua tanda baca dalam setiap ulasan dihapus, yang dapat membantu dalam pemrosesan teks lebih lanjut, seperti tokenisasi atau analisis sentimen, dengan mengurangi gangguan dari karakter-karakter yang tidak relevan.

```
import nltk
# Download the 'punkt' resource
nltk.download('punkt')

# Tokenisasi teks menggunakan nltk
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

# Memisahkan data menjadi train dan test
train, test = train_test_split(df, test_size=0.01,
                                random_state=42)

# TaggedDocument untuk train dan test set
train_tagged = train.apply(
    lambda r:
    TaggedDocument(words=tokenize_text(r['ULASAN']),
                    tags=[r.LABEL]), axis=1)
test_tagged = test.apply(
```

```

lambda r:
    TaggedDocument(words=tokenize_text(r['ULASAN']),
    tags=[r.LABEL]), axis=1)

# Pengaturan tokenizer
max_features = 500000 # Jumlah maksimum kata yang akan
digunakan
max_sequence_length = 50 # Panjang maksimum setiap teks

tokenizer = Tokenizer(num_words=max_features, split=' ',
filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~', lower=True)
tokenizer.fit_on_texts(df['ULASAN'].values)

# Konversi teks ke dalam urutan angka (sequences)
X_train =
tokenizer.texts_to_sequences(train['ULASAN'].values)
X_train = pad_sequences(X_train,
maxlen=max_sequence_length)

X_test =
tokenizer.texts_to_sequences(test['ULASAN'].values)
X_test = pad_sequences(X_test, maxlen=max_sequence_length)

print('Found %s unique tokens.' %
len(tokenizer.word_index))

```

Melakukan serangkaian langkah untuk mempersiapkan data teks ulasan untuk pemodelan pembelajaran mesin. Berikut adalah penjelasannya secara menyeluruh:

Pertama, modul nltk diimpor dan pustaka 'punkt' diunduh, yang diperlukan untuk tokenisasi kalimat dan kata. Fungsi `tokenize_text` kemudian didefinisikan untuk memecah teks menjadi token. Dalam fungsi ini, teks dipecah menjadi kalimat menggunakan `nltk.sent_tokenize`, kemudian setiap kalimat dipecah menjadi kata-kata menggunakan `nltk.word_tokenize`. Setiap kata yang dihasilkan kemudian diubah menjadi huruf kecil dan ditambahkan ke dalam daftar tokens, yang akhirnya dikembalikan oleh fungsi.

Selanjutnya, data ulasan dibagi menjadi dua bagian, yaitu set pelatihan (training set) dan set pengujian (test set), menggunakan fungsi `train_test_split`

dari scikit-learn dengan proporsi 99% untuk pelatihan dan 1% untuk pengujian. Pembagian ini dilakukan secara acak namun konsisten dengan memberikan parameter `random_state` sebesar 42.

Setelah itu, untuk setiap baris dalam set pelatihan dan pengujian, kolom "ULASAN" di-tokenisasi menggunakan fungsi `tokenize_text`, dan setiap teks yang telah di-tokenisasi dibungkus dalam objek `TaggedDocument` bersama dengan labelnya. Langkah ini dilakukan dengan menggunakan `apply` dari `pandas` yang menerapkan fungsi `lambda` pada setiap baris.

Untuk mempersiapkan teks agar dapat digunakan oleh model pembelajaran mesin, sebuah tokenizer dari Keras diatur dengan parameter `max_features` sebesar 500,000 untuk membatasi jumlah kata yang digunakan dan `max_sequence_length` sebesar 50 untuk membatasi panjang maksimum setiap teks. Tokenizer kemudian dilatih (`fit_on_texts`) pada data ulasan dalam dataframe `df` untuk membuat indeks kata.

Teks dalam set pelatihan dan pengujian diubah menjadi urutan angka menggunakan tokenizer, yang merepresentasikan setiap kata dengan angka berdasarkan indeks kata yang telah dibuat. Urutan angka ini kemudian dipadatkan (`pad_sequences`) sehingga setiap urutan memiliki panjang yang sama, yaitu `max_sequence_length`.

Terakhir, jumlah token unik yang ditemukan oleh tokenizer dicetak, memberikan informasi tentang ukuran kosakata yang dihasilkan dari data ulasan.

Secara keseluruhan, kode ini mempersiapkan data teks ulasan dengan membersihkannya, mengubahnya menjadi bentuk numerik, dan memastikan konsistensi panjang urutan untuk dimasukkan ke dalam model pembelajaran mesin.

```
# Konversi teks ke dalam urutan angka (sequences)
```

```
X = tokenizer.texts_to_sequences(df['ULASAN'].values)
X = pad_sequences(X, maxlen=max_sequence_length)

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

Kode ini bertujuan untuk mengubah teks ulasan dalam dataframe `df` menjadi format yang dapat digunakan oleh model pembelajaran mesin. Berikut adalah penjelasannya dalam bentuk paragraf:

Pertama, teks ulasan dalam kolom "ULASAN" dataframe `df` diubah menjadi urutan angka menggunakan `tokenizer` yang telah dilatih sebelumnya. Proses ini dilakukan dengan metode `texts_to_sequences`, yang mengonversi setiap kata dalam teks ulasan menjadi angka berdasarkan indeks kata yang telah dibuat oleh tokenizer. Hasil konversi ini adalah daftar urutan angka, di mana setiap urutan mewakili sebuah ulasan.

Selanjutnya, urutan angka ini dipadatkan menggunakan fungsi `pad_sequences` dari Keras. Fungsi ini memastikan bahwa semua urutan memiliki panjang yang sama, yaitu `max_sequence_length`, dengan menambahkan padding pada urutan yang lebih pendek dari panjang maksimum. Padding ini penting untuk memastikan konsistensi panjang input data saat dimasukkan ke dalam model pembelajaran mesin.

Setelah proses padding selesai, bentuk (shape) dari tensor data yang dihasilkan dicetak menggunakan pernyataan `print`. Bentuk dari tensor ini memberikan informasi tentang dimensi data, di mana `X.shape` menunjukkan jumlah ulasan (jumlah baris) dan panjang maksimum setiap ulasan setelah padding (jumlah kolom). Hasil ini membantu memverifikasi bahwa data telah diproses dan diformat dengan benar sebelum digunakan dalam pelatihan model.

Kode ini mengubah teks ulasan menjadi urutan angka dan memastikan setiap urutan memiliki panjang yang sama, yang merupakan langkah penting dalam pra-pemrosesan data teks untuk analisis dan pemodelan pembelajaran mesin.

```

from gensim.models.doc2vec import Doc2Vec, TaggedDocument

# Ubah ukuran vektor (vector_size) sesuai kebutuhan Anda
vector_size = 20

# Inisialisasi model Doc2Vec
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)

# Membangun kosakata dari tagged documents pada data
pelatihan
train_tagged =
[TaggedDocument(words=tokenize_text(row['ULASAN']),
tags=[row['LABEL']]) for index, row in train.iterrows()]
d2v_model.build_vocab(train_tagged)

```

Kode di atas menggunakan pustaka Gensim untuk membangun dan melatih model Doc2Vec, yang digunakan untuk mengubah teks ulasan menjadi representasi vektor yang dapat digunakan dalam analisis atau pemodelan lanjutan. Berikut adalah penjelasan secara detail:

Pertama, ukuran vektor (`vector_size`) yang diinginkan ditentukan dan disimpan dalam variabel `vector_size`, di mana dalam contoh ini ukurannya adalah 20.

Selanjutnya, model Doc2Vec diinisialisasi dengan parameter sebagai berikut:

1. `dm=1`: Parameter ini menunjukkan penggunaan metode Distributed Memory (DM) untuk melatih model.
2. `dm_mean=1`: Ini menetapkan bahwa vektor yang dihasilkan dari konteks ulasan akan dirata-ratakan.
3. `vector_size`: Parameter ini menentukan panjang dari vektor fitur yang dihasilkan untuk mewakili setiap dokumen (dalam hal ini, setiap ulasan).
4. `window=8`: Ini mengindikasikan jendela konteks yang digunakan saat melatih model, yaitu jumlah kata yang diperhatikan sebelum dan sesudah kata saat ini dalam setiap ulasan.

5. ``min_count=1``: Parameter ini menyaring kata-kata yang muncul kurang dari satu kali dalam seluruh dataset. Kata-kata yang jarang muncul cenderung tidak memberikan informasi yang signifikan.
6. ``workers=1``: Menentukan jumlah thread yang akan digunakan dalam pelatihan model. Pada contoh ini, hanya satu thread yang digunakan.
7. ``alpha=0.065`` dan ``min_alpha=0.065``: Ini adalah laju pembelajaran awal dan minimum yang digunakan dalam proses pelatihan. Laju pembelajaran ini mengatur seberapa cepat model akan mengubah vektor representasi selama pelatihan.

Setelah model Doc2Vec diinisialisasi, langkah berikutnya adalah membangun kosakata (vocabulary) dari tagged documents yang diberikan dalam ``train_tagged``. List ``train_tagged`` dibangun dengan iterasi melalui setiap baris dalam dataframe ``train``. Setiap baris diambil sebagai objek ``TaggedDocument``, di mana ``words`` diisi dengan hasil tokenisasi teks ulasan menggunakan fungsi ``tokenize_text(row['ULASAN'])``, dan ``tags`` diisi dengan label dari baris tersebut (``[row['LABEL']]``). Proses ini dilakukan untuk mempersiapkan data pelatihan yang akan digunakan dalam melatih model Doc2Vec.

Dengan demikian, kode ini mempersiapkan dan membangun model Doc2Vec untuk dapat mengubah teks ulasan menjadi representasi vektor dalam ruang fitur yang numerik, yang dapat digunakan untuk berbagai tujuan analisis atau pemodelan lanjutan seperti klasifikasi atau clustering.

```
d2v_model.build_vocab(train_tagged)

# Latih model
for epoch in range(30):
    d2v_model.train(utils.shuffle(train_tagged),
total_examples=len(train_tagged), epochs=1)
    d2v_model.alpha -= 0.002 # Reduksi alpha setiap epoch
    d2v_model.min_alpha = d2v_model.alpha # Tetapkan
min_alpha sesuai alpha saat ini
```

Digunakan untuk melatih model Doc2Vec yang telah dibangun sebelumnya dengan menggunakan data yang telah dipersiapkan (`train_tagged`). Proses pelatihan dilakukan dalam beberapa tahap, sebagai berikut:

Pertama, perintah `d2v_model.build_vocab(train_tagged)` digunakan untuk membangun kosakata model Doc2Vec dari data pelatihan yang telah ditandai (`train_tagged`). Langkah ini penting karena mempersiapkan model untuk memahami dan merepresentasikan kata-kata serta dokumen-dokumen yang terkandung dalam data pelatihan.

Setelah kosakata dibangun, model dilatih dalam beberapa iterasi (`epoch`). Dalam setiap iterasi, data pelatihan (`train_tagged`) diacak (`utils.shuffle(train_tagged)`) untuk memperkenalkan variasi yang lebih baik dalam pelatihan. Metode `train()` dari model Doc2Vec kemudian dipanggil dengan memberikan parameter berupa data pelatihan yang telah diacak, total contoh yang ada (`total_examples=len(train_tagged)`), dan `epochs=1` untuk menunjukkan bahwa setiap iterasi adalah satu epoch.

Selama proses pelatihan, laju pembelajaran (`alpha`) dari model diperbarui secara berurutan dengan mengurangi nilainya sebesar 0.002 setiap epoch (`d2v_model.alpha -= 0.002`). Proses ini membantu dalam menyesuaikan model agar lebih akurat dalam merepresentasikan dokumen-dokumen berdasarkan konteks yang diberikan.

Terakhir, `d2v_model.min_alpha` diatur kembali ke nilai `d2v_model.alpha`, memastikan bahwa laju pembelajaran minimum tidak lebih tinggi dari laju pembelajaran saat ini setiap kali iterasi epoch selesai.

Kode ini mengimplementasikan proses pelatihan bertahap untuk model Doc2Vec, di mana model secara iteratif memperbarui representasi vektor untuk kata-kata dan dokumen berdasarkan data pelatihan yang diberikan.

Proses ini bertujuan untuk meningkatkan kemampuan model dalam menangkap hubungan antar kata dan dokumen dalam ruang vektor.

```
num_words = len(d2v_model.wv.key_to_index)
print("Jumlah kata dalam kosakata:", num_words)

# Mengakses kata-kata dalam kosakata
words_in_vocab = list(d2v_model.wv.key_to_index.keys())
print("Kata-kata dalam kosakata:", words_in_vocab)
```

Kode di atas bertujuan untuk mengeksplorasi kosakata yang telah dibangun oleh model Doc2Vec setelah proses pelatihan selesai. Pertama, pernyataan `num_words = len(d2v_model.wv.key_to_index)` menghitung jumlah kata unik dalam kosakata model. Ini dilakukan dengan mengakses atribut `wv` dari model Doc2Vec (`d2v_model.wv`), yang berisi representasi kata-kata dalam bentuk vektor, dan kemudian menggunakan `key_to_index` untuk menghitung panjangnya.

Selanjutnya, perintah `words_in_vocab = list(d2v_model.wv.key_to_index.keys())` digunakan untuk mendapatkan daftar kata-kata yang ada dalam kosakata. Proses ini mengambil kunci-kunci (kata-kata) dari `key_to_index` dan mengonversinya menjadi sebuah list.

Hasil dari kedua pernyataan tersebut kemudian dicetak. Pernyataan pertama mencetak jumlah total kata yang terdapat dalam kosakata, sementara pernyataan kedua mencetak daftar kata-kata itu sendiri.

Secara keseluruhan, kode ini memberikan wawasan tentang isi dari kosakata yang telah dibangun oleh model Doc2Vec setelah melalui proses pelatihan. Ini membantu untuk memahami seberapa besar kosakata model yang dihasilkan, serta kata-kata spesifik yang terdapat di dalamnya.

```
# Inisialisasi matriks embedding kosong
embedding_matrix = np.zeros((len(d2v_model.dv.vectors),
                             d2v_model.vector_size))
```

```

# Mengisi matriks embedding dengan vektor-vektor dokumen
dari model Doc2Vec
for i in range(len(d2v_model.dv.vectors)):
    embedding_matrix[i] = d2v_model.dv.vectors[i]

# Contoh penggunaan matriks embedding
print("Ukuran matriks embedding:", embedding_matrix.shape)
print("Contoh vektor untuk dokumen pertama:",
embedding_matrix[0])

```

Bertujuan untuk membuat matriks embedding dari vektor representasi dokumen yang telah dihasilkan oleh model Doc2Vec setelah proses pelatihan selesai. Proses dimulai dengan inisialisasi matriks kosong `embedding_matrix` menggunakan NumPy, dengan dimensi `(len(d2v_model.dv.vectors), d2v_model.vector_size)`. Di sini, `len(d2v_model.dv.vectors)` mengacu pada jumlah dokumen yang telah direpresentasikan dalam model, sedangkan `d2v_model.vector_size` menunjukkan panjang vektor untuk setiap dokumen.

Selanjutnya, dilakukan iterasi sepanjang `d2v_model.dv.vectors`, yang mengandung vektor representasi untuk setiap dokumen dalam model Doc2Vec. Pada setiap iterasi, vektor representasi dokumen tersebut disalin ke baris yang sesuai dalam `embedding_matrix`.

Setelah proses penyalinan selesai, hasilnya dicetak untuk memberikan informasi tentang matriks embedding yang telah dibuat. Pernyataan pertama mencetak ukuran matriks embedding (`embedding_matrix.shape`), yang menunjukkan jumlah baris dan kolomnya. Pernyataan kedua mencetak contoh vektor untuk dokumen pertama dalam matriks embedding (`embedding_matrix[0]`), yang memberikan representasi numerik dari dokumen tersebut dalam ruang vektor.

Kode ini digunakan untuk menghasilkan matriks embedding yang berisi vektor representasi numerik untuk setiap dokumen dalam model Doc2Vec. Matriks embedding ini dapat digunakan sebagai input untuk model

pembelajaran mesin lainnya, seperti model klasifikasi atau clustering, untuk mengekstraksi dan memanfaatkan informasi semantik yang terdapat dalam teks dokumen.

```
from keras.models import Sequential
from keras.layers import Conv1D, GlobalMaxPooling1D,
Dense, Embedding, Dropout

# Definisikan panjang maksimum urutan
MAX_SEQUENCE_LENGTH = 50

# Definisikan jumlah kata unik
num_unique_words = len(tokenizer.word_index) + 1

# Pastikan bahwa embedding_matrix memiliki bentuk yang
sesuai
embedding_matrix = np.random.rand(num_unique_words, 20)

model = Sequential()

# Menambahkan lapisan Embedding dengan bobot yang sesuai
model.add(Embedding(num_unique_words, 20,
input_length=MAX_SEQUENCE_LENGTH,
weights=[embedding_matrix], trainable=True))

# Menambahkan lapisan Conv1D
model.add(Conv1D(50, 3, activation='relu'))

model.add(Dropout(0.5))

# Menambahkan lapisan GlobalMaxPooling1D
model.add(GlobalMaxPooling1D())

# Menambahkan lapisan Dense untuk output
model.add(Dense(3, activation="softmax"))

# Menampilkan ringkasan model
model.summary()

# Kompilasi model
model.compile(optimizer="adam",
loss="categorical_crossentropy", metrics=['acc'])
```

```

# Contoh pemanggilan fungsi split_input
def split_input(sequence):
    return sequence[:-1], sequence[1:]

# Contoh penggunaan split_input
sequence_example = np.array([1, 2, 3, 4, 5])
x, y = split_input(sequence_example)
print("Input:", x)
print("Output:", y)

```

Kode di atas mendefinisikan dan menginisialisasi model jaringan saraf konvolusi (Convolutional Neural Network/CNN) untuk analisis teks menggunakan pustaka Keras. Model ini dibuat untuk mengklasifikasikan teks ke dalam tiga kategori, memanfaatkan embedding yang telah diinisialisasi secara acak.

Pertama, beberapa parameter penting ditentukan, seperti `MAX_SEQUENCE_LENGTH` yang diatur ke 50 dan `num_unique_words` yang dihitung dari tokenizer untuk menentukan jumlah kata unik dalam kosakata ditambah satu. Matriks embedding `embedding_matrix` kemudian diinisialisasi secara acak dengan dimensi `(num_unique_words, 20)`.

Selanjutnya, model Sequential didefinisikan dengan beberapa lapisan:

1. Lapisan Embedding yang menerima `num_unique_words` sebagai input dan memetakan setiap kata ke dalam vektor berdimensi 20. Matriks embedding diinisialisasi dengan `embedding_matrix` dan diatur untuk dapat dilatih.
2. Lapisan Conv1D dengan 50 filter dan ukuran kernel 3, yang menerapkan operasi konvolusi pada input dan menggunakan fungsi aktivasi ReLU.
3. Lapisan Dropout dengan rasio dropout 0.5 untuk mencegah overfitting dengan secara acak menonaktifkan setengah dari neuron selama pelatihan.

4. Lapisan GlobalMaxPooling1D yang mereduksi dimensi output dari lapisan konvolusi dengan mengambil nilai maksimum dari setiap filter.
5. Lapisan Dense dengan 3 neuron dan fungsi aktivasi softmax untuk mengklasifikasikan input ke dalam salah satu dari tiga kategori.

Model tersebut kemudian dirangkum menggunakan `model.summary()`, dan dikompilasi dengan optimizer Adam, menggunakan fungsi loss categorical_crossentropy dan metrik akurasi (`acc`).

Terakhir, fungsi `split_input(sequence)` didefinisikan untuk memisahkan input sequence menjadi dua bagian: input `x` yang terdiri dari semua elemen kecuali elemen terakhir, dan output `y` yang terdiri dari semua elemen kecuali elemen pertama. Contoh penggunaan fungsi ini ditunjukkan dengan `sequence_example`, menghasilkan input `[1, 2, 3, 4]` dan output `[2, 3, 4, 5]`, yang kemudian dicetak.

Secara keseluruhan, kode ini mendemonstrasikan langkah-langkah untuk membangun, menginisialisasi, dan mengkompilasi model CNN untuk analisis teks, serta menyertakan contoh fungsi untuk memproses sequence data.

```
Y = pd.get_dummies(df['LABEL']).values
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
                                                    test_size=0.1, random_state=42)

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)
```

Digunakan untuk mempersiapkan data untuk pelatihan dan pengujian model pembelajaran mesin. Pertama, label dalam kolom 'LABEL' dari DataFrame `df` dikonversi menjadi representasi one-hot encoding menggunakan `pd.get_dummies()`, dan hasilnya disimpan dalam variabel

`Y`. One-hot encoding mengubah kategori label menjadi format biner yang dapat digunakan oleh model pembelajaran mesin.

Kemudian, data dibagi menjadi dua set: data pelatihan dan data pengujian. Fungsi `train_test_split` dari pustaka Scikit-learn digunakan untuk melakukan pembagian ini, dengan 10% dari data disisihkan untuk pengujian (`test_size=0.1`). Parameter `random_state=42` memastikan bahwa pembagian data selalu konsisten setiap kali kode dijalankan.

Setelah pembagian, variabel `X_train` dan `X_test` berisi fitur (data ulasan yang telah diubah menjadi urutan angka), sementara `Y_train` dan `Y_test` berisi label dalam format one-hot encoding. Ukuran dari masing-masing set data kemudian dicetak untuk memastikan bahwa pembagian telah dilakukan dengan benar.

Keseluruhannya, kode ini mempersiapkan data fitur dan label untuk pelatihan dan pengujian model pembelajaran mesin, memastikan bahwa data dibagi dengan proporsi yang sesuai dan dalam format yang tepat untuk digunakan dalam pelatihan model.

```
history = model.fit(X_train, Y_train, epochs=50,
                    batch_size=16, verbose=2, validation_data=(X_test,
                                                                Y_test))

# Mendapatkan histori pelatihan
print(history.history.keys())

# Menampilkan val_loss dan val_accuracy
val_loss = history.history['val_loss']
val_acc = history.history['val_acc']
print("Validation Loss:", val_loss)
print("Validation Accuracy:", val_acc)
```

Digunakan untuk melatih model jaringan saraf konvolusi (CNN) yang telah didefinisikan sebelumnya menggunakan data pelatihan (`X_train` dan `Y_train`) dan memvalidasi kinerjanya menggunakan data pengujian

(`X_test` dan `Y_test`). Proses pelatihan dilakukan selama 50 epoch dengan ukuran batch 16, di mana model akan diperbarui dan dievaluasi setiap 16 sampel data. Parameter `verbose=2` digunakan untuk memberikan output yang lebih rinci selama proses pelatihan.

Setelah pelatihan selesai, objek `history` yang dikembalikan oleh metode `fit` berisi informasi tentang metrik pelatihan dan validasi untuk setiap epoch. Pernyataan `print(history.history.keys())` digunakan untuk menampilkan kunci-kunci dalam dictionary `history.history`, yang mencakup metrik seperti `loss`, `accuracy`, `val_loss`, dan `val_acc`.

Selanjutnya, nilai `val_loss` (kerugian validasi) dan `val_acc` (akurasi validasi) diekstraksi dari objek `history` dan dicetak. Nilai `val_loss` menunjukkan seberapa baik model memprediksi data pengujian dalam hal kerugian, sementara `val_acc` menunjukkan akurasi model dalam memprediksi label yang benar pada data pengujian.

Secara keseluruhan, kode ini melatih model CNN dan mengevaluasi kinerjanya menggunakan data validasi, memberikan wawasan tentang seberapa baik model tersebut dalam generalisasi terhadap data baru yang tidak terlihat selama pelatihan.

```
history_dict = history.history

# Ekstrak nilai untuk setiap metrik
loss_values = history_dict['loss']
val_loss_values = history_dict['val_loss']
acc_values = history_dict['acc']
val_acc_values = history_dict['val_acc']

# Buat range untuk jumlah epoch
epochs = range(1, len(loss_values) + 1)

# Plot Loss
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()

# Plot Accuracy
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()

```

Bertujuan untuk memvisualisasikan performa model CNN selama proses pelatihan dan validasi, dengan menggambarkan kurva kerugian (loss) dan akurasi (accuracy) dari setiap epoch. Pertama, data historis dari proses pelatihan disimpan dalam `history_dict`, yang berisi nilai-nilai untuk kerugian dan akurasi baik untuk pelatihan maupun validasi.

Nilai-nilai untuk kerugian pelatihan dan validasi diekstraksi dari `history_dict` dan disimpan dalam `loss_values` dan `val_loss_values`, sementara nilai-nilai untuk akurasi pelatihan dan validasi disimpan dalam `acc_values` dan `val_acc_values`. Rentang epoch dihitung sebagai `range(1, len(loss_values) + 1)` untuk menggambarkan jumlah epoch selama proses pelatihan.

Kemudian, dua subplots dibuat dalam sebuah figur berukuran 12x4:

1. Subplot pertama menggambarkan kurva kerugian. `plt.plot` digunakan untuk menggambar kurva kerugian pelatihan dengan tanda 'bo' (bulatan biru) dan kurva kerugian validasi dengan tanda 'b' (garis biru). Judul subplot ini adalah "Training and Validation Loss", dengan label sumbu-

x "Epochs" dan sumbu-y "Loss". Sebuah legenda ditambahkan untuk membedakan antara kurva pelatihan dan validasi.

2. Subplot kedua menggambarkan kurva akurasi. `plt.plot`` digunakan untuk menggambar kurva akurasi pelatihan dengan tanda 'bo' (bulatan biru) dan kurva akurasi validasi dengan tanda 'b' (garis biru). Judul subplot ini adalah "Training and Validation Accuracy", dengan label sumbu-x "Epochs" dan sumbu-y "Accuracy". Sebuah legenda juga ditambahkan untuk membedakan antara kurva pelatihan dan validasi.

Secara keseluruhan, kode ini menghasilkan dua grafik yang memvisualisasikan bagaimana kerugian dan akurasi model berubah selama epoch pelatihan, memberikan wawasan tentang apakah model mengalami overfitting atau underfitting serta seberapa baik model tersebut dalam memprediksi data validasi.

```
from sklearn.metrics import classification_report,
f1_score, precision_score, recall_score

# Melakukan prediksi
predictions = model.predict(X_test)
predicted_labels = (predictions > 0.5).astype(int) #
Konversi probabilitas menjadi label biner (0 atau 1)
true_labels = Y_test

# Menghitung metrik evaluasi tambahan
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')
print(f"F1 Score: {f1}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(classification_report(true_labels,
predicted_labels))

# Tampilkan hasil prediksi dalam array
print("Array hasil prediksi:")
```

```
print(true_labels)
print(predicted_labels)
```

Digunakan untuk mengevaluasi performa model CNN pada data pengujian menggunakan metrik-metrik klasifikasi seperti F1 score, precision, recall, dan classification report dari Scikit-learn. Proses ini melibatkan beberapa langkah utama:

1. **Prediksi Probabilitas:** Model digunakan untuk memprediksi probabilitas kelas untuk setiap sampel dalam `X_test` menggunakan `model.predict(X_test)`.
2. **Konversi ke Label Biner:** Probabilitas yang dihasilkan dikonversi menjadi label biner (0 atau 1) menggunakan `(predictions > 0.5).astype(int)`. Jika probabilitas lebih besar dari 0.5, nilai tersebut dikonversi menjadi 1; jika tidak, menjadi 0.
3. **Metrik Evaluasi:** Nilai true labels dari data pengujian (`Y_test`) disimpan dalam `true_labels`.

Metrik evaluasi dihitung sebagai berikut:

1. **F1 Score:** Menggunakan `f1_score(true_labels, predicted_labels, average='weighted')`, yang mengukur keseimbangan antara precision dan recall.
2. **Precision:** Menggunakan `precision_score(true_labels, predicted_labels, average='weighted')`, yang mengukur ketepatan prediksi positif.
3. **Recall:** Menggunakan `recall_score(true_labels, predicted_labels, average='weighted')`, yang mengukur kemampuan model dalam mendeteksi semua contoh positif.
4. **Classification Report:** Classification report yang lebih lengkap dihasilkan menggunakan `classification_report(true_labels, predicted_labels)`, yang memberikan metrik precision, recall, F1 score, dan support untuk setiap kelas.

5. Cetak Hasil: F1 score, precision, recall, dan classification report dicetak untuk memberikan gambaran lengkap tentang performa model. Selain itu, array hasil prediksi (`true_labels` dan `predicted_labels`) juga dicetak untuk analisis lebih lanjut.

Kode ini melakukan evaluasi menyeluruh terhadap performa model CNN pada data pengujian, memberikan berbagai metrik yang penting untuk memahami seberapa baik model tersebut dalam melakukan klasifikasi.

```
print("Panjang Tes Ulasan:", len(test['ULASAN']))
print("Panjang X_test:", len(X_test))
print("Panjang Y_test:", len(Y_test))
print("Panjang true_labels:", len(true_labels))
print("Panjang predicted_labels:", len(predicted_labels))
```

Kode tersebut digunakan untuk mencetak panjang (jumlah elemen) dari beberapa variabel yang terlibat dalam evaluasi model pada data pengujian. Ini membantu memverifikasi bahwa semua variabel memiliki ukuran yang konsisten dan sesuai dengan harapan. Berikut adalah penjelasan langkah demi langkah:

1. `print("Panjang Tes Ulasan:", len(test['ULASAN']))`: Mencetak jumlah ulasan dalam set data pengujian `test`.
2. `print("Panjang X_test:", len(X_test))`: Mencetak panjang dari `X_test`, yang merupakan array atau tensor dari data ulasan yang telah diproses untuk pengujian. Ini harus sesuai dengan jumlah ulasan dalam `test`.
3. `print("Panjang Y_test:", len(Y_test))`: Mencetak panjang dari `Y_test`, yang merupakan array dari label yang sebenarnya (true labels) untuk data pengujian. Ini juga harus sesuai dengan jumlah ulasan dalam `test`.
4. `print("Panjang true_labels:", len(true_labels))`: Mencetak panjang dari `true_labels`, yang merupakan referensi lain dari `Y_test` yang digunakan untuk evaluasi. Panjangnya harus sama dengan `Y_test`.

5. `print("Panjang predicted_labels:", len(predicted_labels))`: Mencetak panjang dari `predicted_labels`, yang merupakan array dari label prediksi yang dihasilkan oleh model. Panjangnya harus sama dengan `true_labels` dan `Y_test`.

Dengan mencetak panjang dari setiap variabel ini, kode memastikan bahwa jumlah data ulasan dan label yang digunakan untuk evaluasi model konsisten dan sesuai, membantu mengidentifikasi potensi kesalahan atau inkonsistensi dalam data.

```
import pandas as pd
from sklearn.metrics import classification_report,
f1_score, precision_score, recall_score

# Melakukan prediksi dengan softmax (misalnya, jika
menggunakan TensorFlow/Keras)
predictions = model.predict(X_test)
predicted_labels = predictions.argmax(axis=1) # Mengambil
kelas dengan probabilitas tertinggi sebagai prediksi

# Pastikan true_labels adalah dalam bentuk indeks kelas
yang sama dengan predicted_labels
true_labels = Y_test.argmax(axis=1) # Jika Y_test adalah
dalam bentuk one-hot encoded, konversi ke indeks kelas

# Menghitung metrik evaluasi tambahan
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')

print(f"F1 Score: {f1}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(classification_report(true_labels,
predicted_labels))

# Ambil ulasan, label sebenarnya, dan prediksi label
# Pastikan panjang semua array sama
```

```

min_length = min(len(test['ULASAN']), len(true_labels),
len(predicted_labels))
test_results = pd.DataFrame({
    'Ulasan': test['ULASAN'].values[:min_length], #
Gunakan min_length untuk memastikan panjang yang sama
    'Label Sebenarnya': true_labels[:min_length],
    'Prediksi': predicted_labels[:min_length]})

# Klasifikasi label 'Negatif', 'Positif', dan 'Netral'
berdasarkan nilai
def classify_label(label):
    if label == 0:
        return 'Negatif'
    elif label == 1:
        return 'Positif'
    else:
        return 'Netral'

# Menambahkan kolom klasifikasi label
test_results['Label Sebenarnya'] = test_results['Label
Sebenarnya'].apply(classify_label)
test_results['Prediksi'] =
test_results['Prediksi'].apply(classify_label)

# Export ke Excel

test_results.to_excel('/content/drive/MyDrive/Skripsi/2020
/Arvianda - Word2Vec CNN/hasil_prediksi2.xlsx',
index=False)

# Tampilkan hasil
print("\nHasil Prediksi:\n", test_results)
print("Data berhasil diekspor ke 'hasil_prediksi2.xlsx'.")

```

Melakukan beberapa langkah untuk mengevaluasi model klasifikasi dan menyimpan hasil prediksi ke dalam file Excel. Berikut adalah penjelasan setiap langkah dalam bentuk paragraf:

Pertama, kode ini mengimpor modul `pandas` dan metrik evaluasi dari Scikit-learn. Selanjutnya, model yang telah dilatih digunakan untuk memprediksi data pengujian (`X_test`) dengan menghasilkan probabilitas untuk setiap kelas. Probabilitas ini kemudian dikonversi menjadi prediksi label dengan

mengambil indeks kelas yang memiliki probabilitas tertinggi. Jika label sebenarnya (`Y_test`) dalam bentuk one-hot encoded, mereka juga dikonversi menjadi indeks kelas untuk memastikan kesesuaian dengan prediksi.

Kemudian, metrik evaluasi seperti F1 score, precision, dan recall dihitung menggunakan `f1_score`, `precision_score`, dan `recall_score` dengan rata-rata berbobot untuk mempertimbangkan ketidakseimbangan kelas. Hasil evaluasi ini dicetak bersama dengan laporan klasifikasi yang lebih rinci yang dihasilkan oleh `classification_report`.

Setelah evaluasi, kode memastikan bahwa panjang array ulasan, label sebenarnya, dan prediksi sama dengan mengambil panjang minimum dari ketiganya. Data ini kemudian disusun ke dalam DataFrame `test_results`, yang mencakup kolom untuk ulasan, label sebenarnya, dan prediksi. Fungsi `classify_label` digunakan untuk mengubah label numerik menjadi label kategorikal seperti 'Negatif', 'Positif', dan 'Netral'.

DataFrame yang dihasilkan kemudian diekspor ke file Excel dengan nama `hasil_prediksi2.xlsx` di Google Drive. Terakhir, kode mencetak hasil prediksi dan mengonfirmasi bahwa data telah berhasil diekspor ke file Excel yang ditentukan.

Secara keseluruhan, kode ini tidak hanya melakukan evaluasi performa model tetapi juga menyusun dan menyimpan hasil prediksi dalam format yang mudah diakses untuk analisis lebih lanjut.

E. Pengujian dan hasil metode

1. Word embedding

```
from gensim.models.doc2vec import Doc2Vec, TaggedDocument  
  
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)

train_tagged =
[TaggedDocument(words=tokenize_text(row[ 'ULASAN' ]),
tags=[row[ 'LABEL' ]]) for index, row in train.iterrows()]
d2v_model.build_vocab(train_tagged)

```

from gensim.models.doc2vec import Doc2Vec, TaggedDocument

Mengimpor kelas Doc2Vec dan TaggedDocument dari pustaka Gensim. Doc2Vec adalah model yang digunakan untuk menghasilkan representasi vektor dari dokumen, sedangkan TaggedDocument digunakan untuk menandai setiap dokumen dengan kata-kata dan tag.

vector_size = 20

Di sini, kita menentukan ukuran vektor (jumlah fitur) yang akan digunakan oleh model Doc2Vec. Dalam hal ini, ukuran vektor ditetapkan menjadi 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)

Baris ini menginisialisasi model Doc2Vec dengan parameter tertentu:

- dm=1: Menggunakan Distributed Memory (DM) mode.
- dm_mean=1: Menggunakan rata-rata dari vektor konteks.
- vector_size=20: Ukuran vektor fitur untuk setiap dokumen.
- window=8: Ukuran jendela konteks di sekitar target kata.
- min_count=1: Mempertimbangkan semua kata yang muncul setidaknya satu kali.
- workers=1: Jumlah thread yang digunakan untuk pelatihan.
- alpha=0.065: Laju pembelajaran awal.
- min_alpha=0.065: Laju pembelajaran minimum

```
train_tagged = [TaggedDocument(words=tokenize_text(row['ULASAN']),  
tags=[row['LABEL']]) for index, row in train.iterrows()]
```

Kode ini membuat daftar TaggedDocument dari data pelatihan. Setiap dokumen dalam data pelatihan ditandai dengan kata-kata (words) dan label (tags). Fungsi tokenize_text digunakan untuk memecah teks ulasan menjadi token.

```
d2v_model.build_vocab(train_tagged)  
for epoch in range(30):  
    d2v_model.train(utils.shuffle(train_tagged),  
total_examples=len(train_tagged), epochs=1)  
    d2v_model.alpha -=  
    d2v_model.min_alpha = d2v_model.alpha
```

d2v_model.build_vocab(train_tagged)

Baris ini membangun kosakata model Doc2Vec dari daftar TaggedDocument yang telah dibuat. Proses ini mencakup mengidentifikasi semua kata unik dalam dokumen yang akan digunakan untuk pelatihan model.

d2v_model.build_vocab(train_tagged)

membangun kosakata (vocabulary) untuk model Doc2Vec dari daftar dokumen yang sudah ditandai (train_tagged). Ini adalah langkah penting sebelum memulai pelatihan model, karena model perlu mengetahui semua kata yang ada di dokumen untuk membuat representasi vektornya.

for epoch in range(30):

memulai loop pelatihan yang akan dijalankan sebanyak 30 kali (30 epoch). Setiap epoch adalah satu iterasi penuh melalui seluruh data pelatihan.

d2v_model.train(utils.shuffle(train_tagged),total_examples=len(train_tagged), epochs=1)

Di dalam loop, model Doc2Vec dilatih menggunakan data yang diacak (utils.shuffle(train_tagged)) untuk memastikan bahwa model tidak

mengingat urutan data yang tetap. `total_examples=len(train_tagged)` menunjukkan jumlah total dokumen yang digunakan untuk pelatihan, dan `epochs=1` berarti model dilatih untuk satu epoch pada setiap iterasi loop.

`d2v_model.alpha -= 0.002`

Mengurangi nilai `alpha` (learning rate) model sebesar 0.002 setelah setiap epoch. Mengurangi learning rate selama pelatihan membantu model untuk menyesuaikan parameter dengan lebih hati-hati seiring waktu, yang dapat meningkatkan kualitas pelatihan.

`d2v_model.min_alpha = d2v_model.alpha`

Menetapkan nilai `min_alpha` model menjadi sama dengan `alpha` saat ini. `min_alpha` adalah nilai minimum dari learning rate yang akan digunakan oleh model, memastikan bahwa learning rate tidak turun di bawah nilai ini selama pelatihan.

```
print(d2v_model)
```

Baris ini mencetak representasi string dari objek `d2v_model`

```
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)
```

`num_words = len(d2v_model.wv.key_to_index)`

Menghitung jumlah pasangan kata-indeks dalam kamus ini, yang mewakili jumlah kata unik dalam kosakata model. Kemudian hasil perhitungan tersebut di simpan dalam variable `num_words`

`print("Jumlah kata dalam kosakata:", num_words)`

Mencetak string "Jumlah kata dalam kosakata:" diikuti dengan nilai dari `num_words`, yang menunjukkan jumlah kata unik dalam kosakata model, ke konsol.

`words_in_vocab = list(d2v_model.wv.key_to_index.keys())`

- `d2v_model.wv.key_to_index.keys()` menghasilkan daftar semua kata (keys) dalam kosakata model.
- `list(d2v_model.wv.key_to_index.keys())` mengubah daftar ini menjadi list Python.
- Hasilnya disimpan dalam variabel `words_in_vocab`.

`print("Kata-kata dalam kosakata:", words_in_vocab)`

Mencetak string "Kata-kata dalam kosakata:" diikuti dengan nilai dari `words_in_vocab`, yang merupakan daftar semua kata unik dalam kosakata model, ke konsol.

`embedding_matrix = np.zeros((len(d2v_model.dv.vectors), d2v_model.vector_size))`

- `len(d2v_model.dv.vectors)` menghitung jumlah dokumen dalam data pelatihan, karena `d2v_model.dv.vectors` adalah array yang menyimpan vektor dokumen.
- `d2v_model.vector_size` memberikan ukuran vektor (jumlah dimensi) yang digunakan untuk merepresentasikan setiap dokumen.
- `np.zeros((len(d2v_model.dv.vectors), d2v_model.vector_size))` menciptakan sebuah matriks dengan bentuk (jumlah_dokumen, ukuran_vektor) yang diisi dengan nilai nol. Matriks ini akan menyimpan vektor-vektor embedding untuk setiap dokumen.
- Hasilnya disimpan dalam variabel `embedding_matrix`.

`for i in range(len(d2v_model.dv.vectors)): embedding_matrix[i] = d2v_model.dv.vectors[i]`

- `for i in range(len(d2v_model.dv.vectors))` membuat loop yang iterasi dari 0 sampai jumlah dokumen dalam data pelatihan.
- `embedding_matrix[i] = d2v_model.dv.vectors[i]` mengisi baris ke-`i` dari `embedding_matrix` dengan vektor embedding dari dokumen ke-`i` yang ada di `d2v_model.dv.vectors`.

```
print("Ukuran matriks embedding:", embedding_matrix.shape)
```

`embedding_matrix.shape` memberikan bentuk dari matriks `embedding_matrix` sebagai tuple (jumlah_dokumen, ukuran_vektor). Mencetak string "Ukuran matriks embedding:" diikuti dengan bentuk dari `embedding_matrix` ke konsol.

```
print("Contoh vektor untuk dokumen pertama:", embedding_matrix[0])
```

`embedding_matrix[0]` mengambil vektor embedding untuk dokumen pertama (baris pertama dari matriks `embedding_matrix`). Mencetak string "Contoh vektor untuk dokumen pertama:" diikuti dengan vektor embedding untuk dokumen pertama ke konsol.

2. Word Embedding ke CNN

```
from keras.models import Sequential
from keras.layers import Conv1D, GlobalMaxPooling1D,
Dense, Embedding, Dropout

MAX_SEQUENCE_LENGTH = 50

num_unique_words = len(tokenizer.word_index) + 1
embedding_matrix = np.random.rand(num_unique_words, 20)

model = Sequential()

model.add(Embedding(num_unique_words, 20,
input_length=MAX_SEQUENCE_LENGTH,
weights=[embedding_matrix], trainable=True))

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

model.add(Dropout(0.5))

model.add(GlobalMaxPooling1D())
model.add(Dense(3, activation="softmax"))

model.summary()

model.compile(optimizer="adam",
loss="categorical_crossentropy", metrics=['acc'])
```

```

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)

```

from keras.models import Sequential

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

Mengimpor kelas-kelas dari pustaka keras yang diperlukan untuk membangun model jaringan saraf konvolusional. Sequential adalah model linier dari Keras, Conv1D adalah lapisan konvolusi 1D, GlobalMaxPooling1D adalah lapisan pemrosesan pooling global, Dense adalah lapisan penuh (fully connected), Embedding adalah lapisan embedding, dan Dropout adalah lapisan dropout.

MAX_SEQUENCE_LENGTH = 50

Mendefinisikan panjang maksimum dari urutan teks yang akan diproses oleh model. Dalam hal ini, panjang maksimum urutan adalah 50 token.

num_unique_words = len(tokenizer.word_index) + 1

Menghitung jumlah kata unik dalam kosakata yang dihasilkan oleh tokenizer, ditambah satu untuk menangani token padding yang biasanya digunakan dalam pemrosesan teks.

embedding_matrix = np.random.rand(num_unique_words, 20)

Membuat matriks embedding acak dengan ukuran (num_unique_words, 20), di mana num_unique_words adalah jumlah kata unik dalam kosakata dan 20 adalah dimensi dari vektor embedding untuk setiap kata.

model = Sequential()

Membuat sebuah model jaringan saraf Sequential yang memungkinkan penambahan lapisan secara berurutan.

model.add(Embedding(num_unique_words, 20, input_length=MAX_SEQUENCE_LENGTH, weights=[embedding_matrix], trainable=True))

Menambahkan lapisan Embedding ke model. Lapisan ini mengubah indeks kata menjadi vektor embedding dengan dimensi 20. num_unique_words adalah ukuran kosakata, 20 adalah dimensi dari vektor embedding, dan input_length=MAX_SEQUENCE_LENGTH adalah panjang maksimum input. weights=[embedding_matrix] menginisialisasi lapisan embedding dengan matriks embedding yang telah didefinisikan, dan trainable=True menunjukkan bahwa bobot embedding dapat diperbarui selama pelatihan.

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

Menambahkan lapisan konvolusi 1D dengan 50 filter, ukuran kernel 3, dan fungsi aktivasi ReLU. Lapisan ini akan mengidentifikasi fitur lokal dalam urutan teks.

model.add(Dropout(0.5))

Menambahkan lapisan dropout dengan rasio dropout 0.5. Dropout adalah teknik regularisasi yang membantu mencegah overfitting dengan mengabaikan 50% neuron secara acak selama pelatihan.

model.add(GlobalMaxPooling1D())

Menambahkan lapisan pooling global maksimum 1D yang mengambil nilai maksimum dari setiap fitur sepanjang dimensi waktu. Ini mengurangi ukuran output dari lapisan konvolusi menjadi satu nilai per fitur.

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

Menambahkan lapisan dense (fully connected) dengan 3 unit dan fungsi aktivasi softmax. Lapisan ini menghasilkan probabilitas untuk 3 kelas, sehingga model dapat mengklasifikasikan input ke dalam salah satu dari tiga kategori.

model.summary()

Menampilkan ringkasan struktur model, termasuk jumlah parameter dan bentuk output setiap lapisan.

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

Mengompilasi model dengan optimizer Adam, fungsi loss categorical_crossentropy (cocok untuk masalah klasifikasi multi-kelas), dan metrik akurasi.

def split_input(sequence):

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

Mendefinisikan fungsi split_input yang membagi urutan input menjadi dua bagian: bagian pertama berisi semua elemen kecuali yang terakhir (sequence[:-1]), dan bagian kedua berisi semua elemen kecuali yang pertama (sequence[1:]). Ini berguna untuk membuat data pelatihan untuk model prediksi urutan.

sequence_example = np.array([1, 2, 3, 4, 5])

x, y = split_input(sequence_example)

print("Input:", x)

print("Output:", y)

Membuat contoh urutan sequence_example dan menggunakan fungsi split_input untuk membaginya menjadi x dan y. x adalah semua elemen kecuali yang terakhir, dan y adalah semua elemen kecuali yang pertama. Mencetak hasilnya untuk memverifikasi pembagian data.

```
Y = pd.get_dummies(df['LABEL']).values
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.1, random_state=42)

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)
```



```
Y = pd.get_dummies(df['LABEL']).values
```

Menggunakan fungsi `pd.get_dummies` dari pustaka `pandas` untuk mengonversi kolom `LABEL` pada `DataFrame` `df` menjadi format one-hot encoded. Ini mengubah label kategori menjadi array biner di mana setiap kategori diwakili oleh kolom terpisah dengan nilai 0 atau 1. `values` mengubah `DataFrame` hasil menjadi array `NumPy`.

```
from sklearn.model_selection import train_test_split
```

Mengimpor fungsi `train_test_split` dari pustaka `scikit-learn`, yang digunakan untuk membagi dataset menjadi set pelatihan dan set pengujian.

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.1,  
random_state=42)
```

Menggunakan `train_test_split` untuk membagi data `X` (fitur) dan `Y` (label) menjadi data pelatihan (`X_train` dan `Y_train`) serta data pengujian (`X_test` dan `Y_test`). Parameter `test_size=0.1` menentukan bahwa 10% dari data akan digunakan sebagai set pengujian, sedangkan `random_state=42` memastikan pembagian yang konsisten setiap kali kode dijalankan dengan menggunakan seed yang sama.

```
print("Shape of X_train:", X_train.shape)
```

Mencetak bentuk (`shape`) dari array `X_train`, yang menunjukkan jumlah sampel dan fitur dalam set pelatihan.

```
print("Shape of Y_train:", Y_train.shape)
```

Mencetak bentuk (`shape`) dari array `Y_train`, yang menunjukkan jumlah sampel dan jumlah label dalam set pelatihan.

```
print("Shape of X_test:", X_test.shape)
```

Mencetak bentuk (`shape`) dari array `X_test`, yang menunjukkan jumlah sampel dan fitur dalam set pengujian.

```
print("Shape of Y_test:", Y_test.shape)
```

Mencetak bentuk (`shape`) dari array `Y_test`, yang menunjukkan jumlah sampel dan jumlah label dalam set pengujian.

```

history = model.fit(X_train, Y_train, epochs=10,
batch_size=16, verbose=2, validation_data=(X_test,
Y_test))

print(history.history.keys())

val_loss = history.history['val_loss']
val_acc = history.history['val_acc']
print("Validation Loss:", val_loss)
print("Validation Accuracy:", val_acc)

```

history = model.fit(X_train, Y_train, epochs=10, batch_size=16, verbose=2, validation_data=(X_test, Y_test))

- **model.fit:** Melatih model menggunakan data pelatihan (X_train dan Y_train). **epochs=10:** Menentukan jumlah epoch atau iterasi penuh melalui seluruh data pelatihan. Model akan dilatih selama 10 epoch.
- **batch_size=16:** Mengatur ukuran batch, yaitu jumlah sampel yang akan diproses sebelum pembaruan bobot dilakukan. Dalam hal ini, ukuran batch adalah 16.
- **verbose=2:** Mengatur tingkat detail output pelatihan. Nilai 2 memberikan output pelatihan yang lebih detail, termasuk kemajuan untuk setiap epoch. **validation_data=(X_test, Y_test):** Menyediakan data pengujian untuk evaluasi model selama pelatihan. Data ini digunakan untuk memantau kinerja model pada set pengujian setelah setiap epoch.
- **history:** Menyimpan objek History yang berisi informasi pelatihan, termasuk nilai kerugian (loss) dan akurasi untuk set pelatihan dan validasi.

print(history.history.keys())

Mencetak kunci dari dictionary history.history, yang menunjukkan metrik-metrik yang tersedia dari pelatihan. Ini biasanya mencakup 'loss', 'accuracy', 'val_loss', dan 'val_accuracy'.

val_loss = history.history['val_loss']

Mengambil nilai-nilai kerugian (loss) pada data validasi untuk setiap epoch dari objek `history`. Ini memberikan gambaran tentang seberapa baik model berkinerja pada set validasi.

```
val_acc = history.history['val_acc']
```

Mengambil nilai-nilai akurasi pada data validasi untuk setiap epoch dari objek `history`. Ini menunjukkan seberapa akurat model dalam membuat prediksi pada set validasi.

```
print("Validation Loss:", val_loss)
```

Mencetak nilai-nilai kerugian (loss) pada data validasi yang telah diambil sebelumnya, memberikan informasi tentang seberapa baik model dalam memprediksi data validasi selama pelatihan.

```
print("Validation Accuracy:", val_acc)
```

Mencetak nilai-nilai akurasi pada data validasi yang telah diambil sebelumnya, memberikan informasi tentang akurasi model pada set validasi selama pelatihan.

```
from sklearn.metrics import classification_report,
f1_score, precision_score, recall_score

predictions = model.predict(X_test)
predicted_labels = (predictions > 0.5).astype(int)
true_labels = Y_test

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')
print(f"F1 Score: {f1}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(classification_report(true_labels,
predicted_labels))

print("Array hasil prediksi:")
print(true_labels)
print(predicted_labels)
```

```
from sklearn.metrics import classification_report, f1_score,  
precision_score, recall_score
```

Mengimpor fungsi-fungsi dari sklearn.metrics yang digunakan untuk evaluasi model. Ini termasuk classification_report, f1_score, precision_score, dan recall_score.

```
predictions = model.predict(X_test)
```

Menggunakan model yang sudah dilatih untuk membuat prediksi pada data pengujian (X_test). Hasilnya adalah array dari probabilitas prediksi untuk setiap kelas.

```
predicted_labels = (predictions > 0.5).astype(int) # Konversi  
probabilitas menjadi label biner (0 atau 1)
```

Mengonversi probabilitas prediksi menjadi label biner. Jika probabilitas lebih besar dari 0.5, maka diklasifikasikan sebagai 1, dan sebaliknya sebagai 0. astype(int) digunakan untuk mengubah tipe data menjadi integer.

```
true_labels = Y_test
```

Menyimpan label sebenarnya dari data pengujian (Y_test) ke dalam variabel true_labels.

```
f1 = f1_score(true_labels, predicted_labels, average='weighted')
```

Menghitung skor F1, yang merupakan harmonisasi rata-rata dari precision dan recall. average='weighted' berarti skor dihitung dengan memberi bobot sesuai dengan jumlah sampel di setiap kelas.

```
precision = precision_score(true_labels, predicted_labels,  
average='weighted')
```

Menghitung recall, yaitu proporsi sampel positif yang benar-benar terdeteksi sebagai positif. average='weighted' berarti recall dihitung dengan memberi bobot sesuai dengan jumlah sampel di setiap kelas.

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

```
print(f"Precision: {precision}")
```

```
print(f"Recall: {recall}")
```

Mencetak nilai F1 Score, Precision, dan Recall yang telah dihitung.

```
print(classification_report(true_labels, predicted_labels))
```

Mencetak laporan klasifikasi yang berisi precision, recall, F1-score, dan support (jumlah sampel) untuk setiap kelas.

```
print("Array hasil prediksi:")
```

```
print(true_labels)
```

```
print(predicted_labels)
```

Mencetak array label sebenarnya (true_labels) dan label prediksi (predicted_labels). Ini memberikan gambaran tentang bagaimana model memprediksi kelas dibandingkan dengan label yang sebenarnya.

```
import pandas as pd
from sklearn.metrics import classification_report,
f1_score, precision_score, recall_score

predictions = model.predict(X_test)
predicted_labels = predictions.argmax(axis=1)

true_labels = Y_test.argmax(axis=1)

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')

print(f"F1 Score: {f1}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(classification_report(true_labels,
predicted_labels))

min_length = min(len(test['ULASAN']), len(true_labels),
len(predicted_labels))
test_results = pd.DataFrame({
    'Ulasan': test['ULASAN'].values[:min_length],
    'Label Sebenarnya': true_labels[:min_length],
    'Prediksi': predicted_labels[:min_length]
})

def classify_label(label):
```

```

if label == 0:
    return 'Negatif'
elif label == 1:
    return 'Positif'
else:
    return 'Netral'

test_results['Label Sebenarnya'] = test_results['Label
Sebenarnya'].apply(classify_label)
test_results['Prediksi'] =
test_results['Prediksi'].apply(classify_label)

test_results.to_excel('/content/drive/MyDrive/Skripsi/202
0/Arvianda - Word2Vec CNN/hasil_prediksi2.xlsx',
index=False)

print("\nHasil Prediksi:\n", test_results)
print("Data berhasil diekspor ke
'hasil_prediksi2.xlsx'.")

```

```

import pandas as pd from sklearn.metrics import classification_report,
f1_score, precision_score, recall_score

```

Mengimpor pustaka pandas untuk manipulasi data dan fungsi-fungsi dari sklearn.metrics untuk evaluasi model, termasuk classification_report, f1_score, precision_score, dan recall_score.

```

predictions = model.predict(X_test)

```

Menggunakan model yang telah dilatih untuk membuat prediksi pada data pengujian (X_test). Hasilnya adalah array probabilitas prediksi untuk setiap kelas.

```

predicted_labels = predictions.argmax(axis=1)

```

Mengonversi probabilitas prediksi menjadi label kelas dengan mengambil indeks dari probabilitas tertinggi di setiap prediksi.

```

true_labels = Y_test.argmax(axis=1)

```

Mengonversi label sebenarnya (Y_test) dari bentuk one-hot encoding menjadi indeks kelas.

```

f1 = f1_score(true_labels, predicted_labels, average='weighted')

```

Menghitung skor F1, yang merupakan harmonisasi rata-rata dari precision dan recall. `average='weighted'` berarti skor dihitung dengan memberi bobot sesuai dengan jumlah sampel di setiap kelas.

```
precision = precision_score(true_labels, predicted_labels,  
average='weighted')
```

Menghitung precision, yaitu proporsi prediksi positif yang benar-benar positif. `average='weighted'` berarti precision dihitung dengan memberi bobot sesuai dengan jumlah sampel di setiap kelas.

```
recall = recall_score(true_labels, predicted_labels, average='weighted')
```

Menghitung recall, yaitu proporsi sampel positif yang benar-benar terdeteksi sebagai positif. `average='weighted'` berarti recall dihitung dengan memberi bobot sesuai dengan jumlah sampel di setiap kelas.

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

```
print(f"Precision: {precision}")
```

```
print(f"Recall: {recall}")
```

```
print(classification_report(true_labels, predicted_labels))
```

Mencetak nilai F1 Score, Precision, dan Recall yang telah dihitung, serta laporan klasifikasi yang berisi precision, recall, F1-score, dan support (jumlah sampel) untuk setiap kelas.

```
min_length = min(len(test['ULASAN']), len(true_labels),  
len(predicted_labels))
```

Menghitung panjang minimum dari kolom 'ULASAN' dalam data pengujian (`test['ULASAN']`), label sebenarnya (`true_labels`), dan label prediksi (`predicted_labels`). Hal ini untuk memastikan semua array memiliki panjang yang sama.

```
test_results = pd.DataFrame({  
    'Ulasan': test['ULASAN'].values[:min_length],  
    'Label Sebenarnya': true_labels[:min_length],  
    'Prediksi': predicted_labels[:min_length]  
})
```

Membuat DataFrame `test_results` yang berisi kolom 'Ulasan', 'Label Sebenarnya', dan 'Prediksi' dengan panjang yang sesuai dengan `min_length`.

```
def classify_label(label):
```

```
    if label == 0:  
        return 'Negatif'  
    elif label == 1:  
        return 'Positif'  
    else:  
        return 'Netral'
```

Mendefinisikan fungsi `classify_label` yang mengklasifikasikan label numerik menjadi label teks ('Negatif', 'Positif', atau 'Netral').

```
test_results['Label Sebenarnya'] = test_results['Label  
Sebenarnya'].apply(classify_label)
```

```
test_results['Prediksi'] = test_results['Prediksi'].apply(classify_label)
```

Mengaplikasikan fungsi `classify_label` pada kolom 'Label Sebenarnya' dan 'Prediksi' dalam DataFrame `test_results` untuk mengubah label numerik menjadi label teks.

```
test_results.to_excel('/content/drive/MyDrive/Skripsi/2020/Arvianda -  
Word2Vec CNN/hasil_prediksi2.xlsx', index=False)
```

Mengekspor DataFrame `test_results` ke file Excel dengan nama 'hasil_prediksi2.xlsx' di Google Drive tanpa menyertakan indeks.

```
print("\nHasil Prediksi:\n", test_results) print("Data berhasil diekspor  
ke 'hasil_prediksi2.xlsx'.")
```

Mencetak DataFrame `test_results` yang berisi ulasan, label sebenarnya, dan prediksi label. Juga mencetak pesan bahwa data berhasil diekspor ke file Excel.

3. Hasil perbandingan

Membandingkan hasil akurasi Word Embedding CNN dan yang hanya menggunakan CNN saja pada epoch yang sama yaitu 10.

- Word Embedding CNN
 1. Pembagian data 90:10
 - a. Epoch

```

Epoch 1/10
254/254 - 3s - loss: 1.0727 - acc: 0.4321 - val_loss: 1.0183 - val_acc: 0.5511 - 3s/epoch - 13ms/step
Epoch 2/10
254/254 - 1s - loss: 0.9874 - acc: 0.5822 - val_loss: 0.9810 - val_acc: 0.6289 - 1s/epoch - 5ms/step
Epoch 3/10
254/254 - 1s - loss: 0.7588 - acc: 0.6647 - val_loss: 0.7949 - val_acc: 0.6778 - 1s/epoch - 4ms/step
Epoch 4/10
254/254 - 1s - loss: 0.6337 - acc: 0.7402 - val_loss: 0.7167 - val_acc: 0.7133 - 1s/epoch - 4ms/step
Epoch 5/10
254/254 - 1s - loss: 0.5468 - acc: 0.7847 - val_loss: 0.6588 - val_acc: 0.7467 - 1s/epoch - 5ms/step
Epoch 6/10
254/254 - 1s - loss: 0.4704 - acc: 0.8217 - val_loss: 0.6040 - val_acc: 0.7489 - 1s/epoch - 4ms/step
Epoch 7/10
254/254 - 1s - loss: 0.4163 - acc: 0.8373 - val_loss: 0.5846 - val_acc: 0.7667 - 1s/epoch - 5ms/step
Epoch 8/10
254/254 - 1s - loss: 0.3639 - acc: 0.8662 - val_loss: 0.5642 - val_acc: 0.7689 - 1s/epoch - 5ms/step
Epoch 9/10
254/254 - 1s - loss: 0.3268 - acc: 0.8805 - val_loss: 0.5449 - val_acc: 0.7756 - 1s/epoch - 4ms/step
Epoch 10/10
254/254 - 1s - loss: 0.2947 - acc: 0.8931 - val_loss: 0.5414 - val_acc: 0.7733 - 1s/epoch - 6ms/step

```

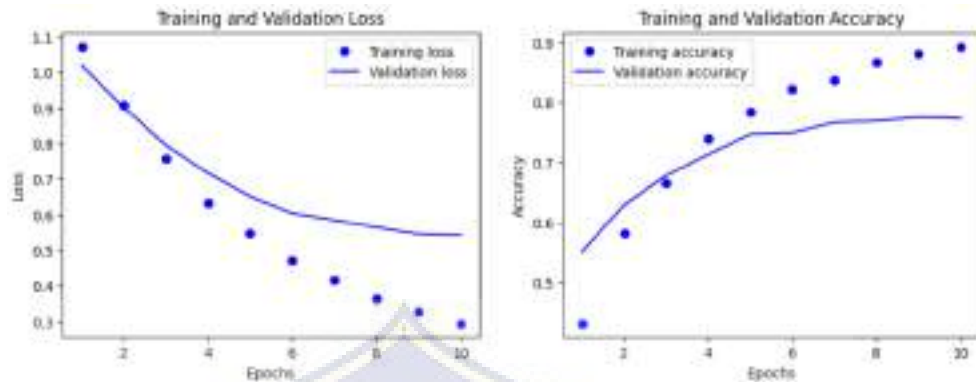
Gambar 8. Proses epoch 90:10

Gambar ini menampilkan serangkaian baris teks yang tampaknya merupakan hasil dari proses pelatihan model pembelajaran mesin. Setiap baris dimulai dengan “Epoch” diikuti oleh pecahan yang menunjukkan kemajuan pelatihan (misalnya, 1/10, 2/10, dst.), dan mencakup berbagai metrik seperti loss, akurasi (acc), val_loss, dan val_acc untuk setiap epoch. Angka-angka yang mengikuti metrik ini mewakili nilai-nilai mereka pada titik tertentu dalam proses pelatihan. Di akhir setiap baris disebutkan jumlah sampel yang diproses per langkah dan total langkah per epoch.

Menarik atau relevan karena memberikan wawasan tentang bagaimana kinerja model pembelajaran mesin berkembang seiring waktu selama pelatihan, yang sangat penting untuk memahami perilaku model dan membuat keputusan tentang penyesuaian untuk meningkatkan kinerjanya.

Pada gambar di atas di dapatkan Validation Accuracy atau hasil training tertinggi yaitu 77%.

b. Grafik



Gambar 9. Grafik accuracy dan loss 90:10

Grafik di atas menggambarkan kinerja model machine learning selama proses pelatihan (training) dan validasi terhadap data. Grafik ini terbagi menjadi dua bagian:

1. Training and Validation Loss: Grafik ini menunjukkan bahwa loss (kesalahan prediksi) dari model menurun seiring dengan bertambahnya jumlah epochs (iterasi pelatihan). Garis biru menunjukkan loss dari data pelatihan yang terus menurun, menandakan model semakin baik dalam memprediksi data yang dilihat selama pelatihan. Garis biru dengan titik-titik menunjukkan loss dari data validasi. Jika garis ini stabil atau menurun, menunjukkan model yang general baik terhadap data baru yang tidak dilihat selama proses pelatihan.
2. Training and Validation Accuracy: Grafik ini menunjukkan akurasi model. Garis biru menunjukkan akurasi pelatihan yang meningkat seiring dengan berjalannya epochs, menandakan model semakin tepat dalam mengklasifikasikan data pelatihan. Sementara itu, garis biru dengan titik-titik menunjukkan akurasi pada data validasi. Akurasi pada data validasi yang meningkat menunjukkan bahwa model tidak hanya belajar "menghafal" data pelatihan tetapi juga generalisasi dengan baik ke data yang tidak terlihat sebelumnya.

Kedua grafik ini sangat penting untuk mengevaluasi model dalam hal overfitting. Overfitting terjadi ketika model terlalu sempurna dalam memprediksi data pelatihan namun performanya buruk pada data baru. Idealnya, kedua garis pada kedua grafik pelatihan dan validasi harus mendekati satu sama lain, yang menunjukkan model yang baik dan generalisasi yang baik.

Pada kasus kasus di atas, tampaknya terdapat indikasi baik dari loss yang menurun dan akurasi yang meningkat baik pada data pelatihan maupun validasi, yang menandakan model yang Anda kembangkan berpotensi baik dalam performanya prediksi.

c. Prediksi

```
F1 Score: 0.7604043715952232
Precision: 0.821946619840725
Recall: 0.7177777777777777
```

	precision	recall	f1-score	support
0	0.88	0.75	0.81	138
1	0.83	0.59	0.69	160
2	0.76	0.82	0.79	152
micro avg	0.82	0.72	0.76	450
macro avg	0.82	0.72	0.76	450
weighted avg	0.82	0.72	0.76	450
samples avg	0.72	0.72	0.72	450

Gambar 10. Hasil prediksi 90:10

Gambar di atas hasil evaluasi dari model klasifikasi yang menggunakan metrik Precision, Recall, dan F1-Score untuk tiga kelas (0, 1, dan 2), serta rata-rata berbagai metrik tersebut.

1. Precision: Menunjukkan proporsi positif yang diprediksi yang benar-benar positif. Misalnya, untuk kelas 0, precision 0.88 berarti 88% dari semua prediksi kelas 0 adalah benar.
2. Recall (Sensitivity): Menunjukkan seberapa baik model dalam menangkap kasus positif. Untuk kelas 2, recall 0.82 berarti model dapat mengidentifikasi 82% dari semua kasus aktual kelas 2.

3. F1-Score: Merupakan harmonic mean dari precision dan recall, memberikan ukuran keseimbangan antara keduanya. Sebagai contoh, F1-Score untuk kelas 0 adalah 0.81, menunjukkan keseimbangan yang baik antara precision dan recall.
4. Support: Menunjukkan jumlah sampel sebenarnya untuk setiap kelas yang digunakan untuk menghitung metrik. Kelas 0 memiliki 138 sampel, kelas 1 memiliki 160 sampel, dan kelas 2 memiliki 152 sampel.
5. Rata-rata:
 - Micro Average: Menghitung total true positives, false negatives, dan false positives secara global sebelum menghitung metrik.
 - Macro Average: Memberikan rata-rata aritmatika dari metrik untuk setiap kelas tanpa mempertimbangkan proporsi masing-masing kelas.
 - Weighted Average: Seperti macro, tapi ini juga mempertimbangkan support (jumlah sampel per kelas), memberikan bobot lebih pada kelas dengan lebih banyak sampel.
 - Samples Average: Metrik yang dihitung untuk setiap sampel secara individual, kemudian dirata-ratakan.

Secara keseluruhan, model ini menunjukkan performa yang relatif baik dengan F1 Score keseluruhan 0.76, Precision 0.82, dan Recall 0.72, mengindikasikan model ini efektif dalam memprediksi kelas-kelas yang ada dengan keseimbangan yang cukup antara kemampuan mengidentifikasi kasus positif dan ketepatan prediksinya.

d. Klasifikasi

```

F1 Score: 0.7712182955373931
Precision: 0.7787719286745893
Recall: 0.7733333333333333

```

	precision	recall	f1-score	support
0	0.84	0.81	0.82	138
1	0.79	0.65	0.71	160
2	0.72	0.87	0.79	152
accuracy			0.77	450
macro avg	0.78	0.78	0.77	450
weighted avg	0.78	0.77	0.77	450

Gambar 11. Hasil klasifikasi label 90:10

Gambar di atas menunjukkan berisi hasil evaluasi klasifikasi untuk model yang dibagi menjadi tiga kelas (0, 1, dan 2), menggunakan metrik Precision, Recall, dan F1-Score, serta nilai keseluruhan untuk akurasi dan rata-rata berbobot.

1. Precision:

- Kelas 0: 0.84 artinya 84% dari prediksi sebagai kelas 0 adalah benar.
- Kelas 1: 0.79 artinya 79% dari prediksi sebagai kelas 1 adalah benar.
- Kelas 2: 0.72 artinya 72% dari prediksi sebagai kelas 2 adalah benar.

2. Recall:

- Kelas 0: 0.81 artinya model dapat mengidentifikasi 81% dari semua kasus aktual kelas 0.
- Kelas 1: 0.65 artinya model dapat mengidentifikasi 65% dari semua kasus aktual kelas 1.
- Kelas 2: 0.87 artinya model dapat mengidentifikasi 87% dari semua kasus aktual kelas 2.

3. F1-Score:

- Kelas 0: 0.82 merupakan rata-rata harmonis dari precision dan recall, menunjukkan keseimbangan yang baik.
- Kelas 1: 0.71 menunjukkan keseimbangan yang lebih rendah antara precision dan recall.
- Kelas 2: 0.79 menunjukkan keseimbangan yang baik.

4. Support: Menunjukkan jumlah sampel sebenarnya untuk masing-masing kelas yang digunakan dalam perhitungan metrik.
5. Akurasi Keseluruhan: 0.77 menunjukkan bahwa 77% dari semua prediksi adalah benar.
6. Rata-rata:
 - Macro Average: Rata-rata aritmatika dari metrik untuk semua kelas tanpa mempertimbangkan jumlah sampel per kelas.
 - Weighted Average: Seperti macro average, tetapi metrik ini memberikan bobot lebih pada kelas dengan lebih banyak sampel, di sini hasilnya sama untuk kedua jenis rata-rata, menunjukkan distribusi sampel yang relatif seimbang.

Secara keseluruhan, tabel ini memberikan gambaran komprehensif mengenai kinerja model klasifikasi, dengan menunjukkan kekuatan dan kelemahan dalam memprediksi setiap kelas, dan efektivitas model secara keseluruhan.

2. Pembagian data 80:20

a. Epoch

```

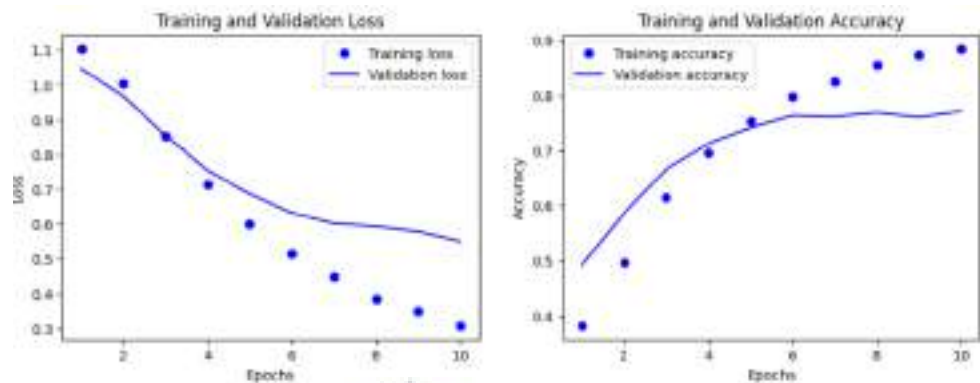
Epoch 1/10
225/225 - 2s - loss: 1.1845 - acc: 0.3819 - val_loss: 1.8467 - val_acc: 0.4911 - 2s/epoch - 3ms/step
Epoch 2/10
225/225 - 1s - loss: 1.8848 - acc: 0.4961 - val_loss: 0.9684 - val_acc: 0.5856 - 1s/epoch - 5ms/step
Epoch 3/10
225/225 - 1s - loss: 0.8547 - acc: 0.6147 - val_loss: 0.8553 - val_acc: 0.6656 - 1s/epoch - 5ms/step
Epoch 4/10
225/225 - 1s - loss: 0.7132 - acc: 0.6953 - val_loss: 0.7547 - val_acc: 0.7122 - 1s/epoch - 5ms/step
Epoch 5/10
225/225 - 1s - loss: 0.6815 - acc: 0.7531 - val_loss: 0.6871 - val_acc: 0.7411 - 1s/epoch - 5ms/step
Epoch 6/10
225/225 - 1s - loss: 0.5143 - acc: 0.7986 - val_loss: 0.6323 - val_acc: 0.7633 - 1s/epoch - 6ms/step
Epoch 7/10
225/225 - 1s - loss: 0.4595 - acc: 0.8258 - val_loss: 0.6024 - val_acc: 0.7611 - 1s/epoch - 8ms/step
Epoch 8/10
225/225 - 1s - loss: 0.3858 - acc: 0.8558 - val_loss: 0.5936 - val_acc: 0.7689 - 1s/epoch - 5ms/step
Epoch 9/10
225/225 - 1s - loss: 0.3581 - acc: 0.8731 - val_loss: 0.5788 - val_acc: 0.7688 - 1s/epoch - 5ms/step
Epoch 10/10
225/225 - 1s - loss: 0.3884 - acc: 0.8847 - val_loss: 0.5507 - val_acc: 0.7711 - 1s/epoch - 5ms/step

```

Gambar 12. Proses epoch 80:20

Pada gambar di atas di dapatkan Validation Accuracy atau hasil training tertinggi yaitu 77%.

b. Grafik



Gambar 13. Grafik accuracy dan loss 80:20

Grafik di atas menggambarkan kinerja model machine learning selama proses pelatihan (training) dan validasi terhadap data. Grafik ini terbagi menjadi dua bagian:

1. Training and Validation Loss: Grafik ini menunjukkan bahwa loss (kesalahan prediksi) dari model menurun seiring dengan bertambahnya jumlah epochs (iterasi pelatihan). Garis biru menunjukkan loss dari data pelatihan yang terus menurun, menandakan model semakin baik dalam memprediksi data yang dilihat selama pelatihan. Garis biru dengan titik-titik menunjukkan loss dari data validasi. Jika garis ini stabil atau menurun, menunjukkan model yang general baik terhadap data baru yang tidak dilihat selama proses pelatihan.
2. Training and Validation Accuracy: Grafik ini menunjukkan akurasi model. Garis biru menunjukkan akurasi pelatihan yang meningkat seiring dengan berjalannya epochs, menandakan model semakin tepat dalam mengklasifikasikan data pelatihan. Sementara itu, garis biru dengan titik-titik menunjukkan akurasi pada data validasi. Akurasi pada data validasi yang meningkat menunjukkan bahwa model tidak hanya belajar "menghafal" data pelatihan tetapi juga generalisasi dengan baik ke data yang tidak terlihat sebelumnya.

Kedua grafik ini sangat penting untuk mengevaluasi model dalam hal overfitting. Overfitting terjadi ketika model terlalu sempurna dalam memprediksi data pelatihan

namun performanya buruk pada data baru. Idealnya, kedua garis pada kedua grafik pelatihan dan validasi harus mendekati satu sama lain, yang menunjukkan model yang baik dan generalisasi yang baik.

Pada kasus kasus di atas, tampaknya terdapat indikasi baik dari loss yang menurun dan akurasi yang meningkat baik pada data pelatihan maupun validasi, yang menandakan model yang Anda kembangkan berpotensi baik dalam performanya prediksi.

c. Prediksi

```

F1 Score: 0.7603766942271739
Precision: 0.800833901673023
Recall: 0.7255555555555555

```

	precision	recall	f1-score	support
0	0.83	0.78	0.80	295
1	0.75	0.62	0.68	294
2	0.82	0.78	0.80	311
micro avg	0.80	0.73	0.76	900
macro avg	0.80	0.72	0.76	900
weighted avg	0.80	0.73	0.76	900
samples avg	0.73	0.73	0.73	900

Gambar 14. Hasil Prediksi 80:20

1. Precision: untuk kelas 0, precision 0.83 berarti 83% dari semua prediksi kelas 0 adalah benar.
2. Recall (Sensitivity): untuk kelas 2, recall 0.78 berarti model dapat mengidentifikasi 78% dari semua kasus aktual kelas 2.
3. F1-Score: F1-Score untuk kelas 0 adalah 0.80, menunjukkan keseimbangan yang baik antara precision dan recall.
4. Support: Menunjukkan jumlah sampel sebenarnya untuk setiap kelas yang digunakan untuk menghitung metrik. Kelas 0 memiliki 295 sampel, kelas 1 memiliki 294 sampel, dan kelas 2 memiliki 311 sampel.

Secara keseluruhan, model ini menunjukkan performa yang relatif baik dengan F1 Score keseluruhan 0.76, Precision 0.80, dan Recall 0.73, mengindikasikan model

ini efektif dalam memprediksi kelas-kelas yang ada dengan keseimbangan yang cukup antara kemampuan mengidentifikasi kasus positif dan ketepatan prediksinya.

d. Klasifikasi

```

F1 Score: 0.7705654957440907
Precision: 0.7702958526513878
Recall: 0.7711111111111111

```

	precision	recall	f1-score	support
0	0.81	0.81	0.81	295
1	0.71	0.69	0.70	294
2	0.79	0.81	0.80	311
accuracy			0.77	900
macro avg	0.77	0.77	0.77	900
weighted avg	0.77	0.77	0.77	900

Gambar 15. Hasil klasifikasi label 80:20

1. Precision:

- Kelas 0: 0.81 artinya 81% dari prediksi sebagai kelas 0 adalah benar.
- Kelas 1: 0.71 artinya 71% dari prediksi sebagai kelas 1 adalah benar.
- Kelas 2: 0.79 artinya 79% dari prediksi sebagai kelas 2 adalah benar.

2. Recall:

- Kelas 0: 0.81 artinya model dapat mengidentifikasi 81% dari semua kasus aktual kelas 0.
- Kelas 1: 0.69 artinya model dapat mengidentifikasi 69% dari semua kasus aktual kelas 1.
- Kelas 2: 0.81 artinya model dapat mengidentifikasi 81% dari semua kasus aktual kelas 2.

3. F1-Score:

- Kelas 0: 0.81 merupakan rata-rata harmonis dari precision dan recall, menunjukkan keseimbangan yang baik.
- Kelas 1: 0.70 menunjukkan keseimbangan yang lebih rendah antara precision dan recall.
- Kelas 2: 0.80 menunjukkan keseimbangan yang baik.

4. Akurasi Keseluruhan: 0.77 menunjukkan bahwa 77% dari semua prediksi adalah benar.

3. Pembagian data 70:30

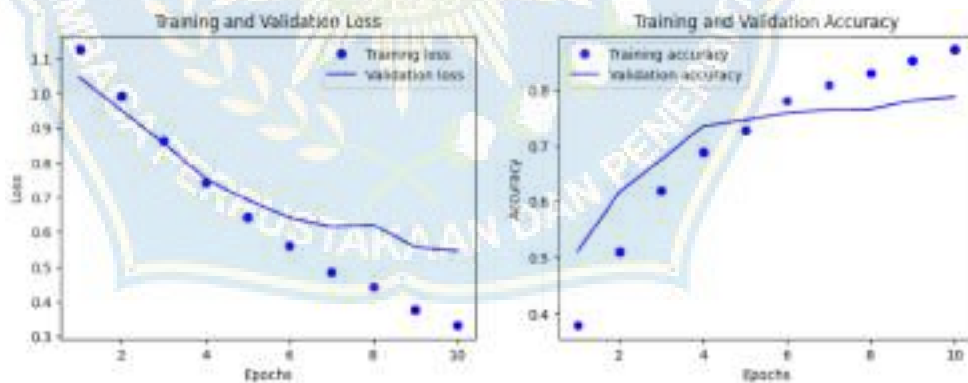
a. Epoch

```
Epoch 1/10  
197/197 - 2s - loss: 1.1254 - acc: 0.3784 - val_loss: 1.0432 - val_acc: 0.5184 - 2s/epoch - 10ms/step  
Epoch 2/10  
197/197 - 1s - loss: 0.9911 - acc: 0.5111 - val_loss: 0.9490 - val_acc: 0.6178 - 1s/epoch - 6ms/step  
Epoch 3/10  
197/197 - 2s - loss: 0.8617 - acc: 0.6218 - val_loss: 0.8563 - val_acc: 0.6756 - 2s/epoch - 8ms/step  
Epoch 4/10  
197/197 - 1s - loss: 0.7434 - acc: 0.6886 - val_loss: 0.7531 - val_acc: 0.7356 - 1s/epoch - 7ms/step  
Epoch 5/10  
197/197 - 1s - loss: 0.6438 - acc: 0.7273 - val_loss: 0.6920 - val_acc: 0.7467 - 972ms/epoch - 5ms/step  
Epoch 6/10  
197/197 - 1s - loss: 0.5683 - acc: 0.7883 - val_loss: 0.6417 - val_acc: 0.7585 - 1s/epoch - 5ms/step  
Epoch 7/10  
197/197 - 1s - loss: 0.4846 - acc: 0.8895 - val_loss: 0.6162 - val_acc: 0.7644 - 1s/epoch - 5ms/step  
Epoch 8/10  
197/197 - 1s - loss: 0.4414 - acc: 0.8111 - val_loss: 0.6287 - val_acc: 0.7652 - 990ms/epoch - 5ms/step  
Epoch 9/10  
197/197 - 1s - loss: 0.3778 - acc: 0.8527 - val_loss: 0.5571 - val_acc: 0.7815 - 978ms/epoch - 5ms/step  
Epoch 10/10  
197/197 - 1s - loss: 0.3329 - acc: 0.8733 - val_loss: 0.5464 - val_acc: 0.7881 - 1s/epoch - 5ms/step
```

Gambar 16. Proses epoch 70:30

Pada gambar di atas di dapatkan Validation Accuracy hasil training tertinggi yaitu 79%.

b. Grafik



Gambar 17. Grafik accuracy dan loss 70:30

Grafik di atas menggambarkan kinerja model machine learning selama proses pelatihan (training) dan validasi terhadap data. Grafik ini terbagi menjadi dua bagian:

1. Training and Validation Loss: Grafik ini menunjukkan bahwa loss (kesalahan prediksi) dari model menurun seiring dengan bertambahnya jumlah epochs (iterasi pelatihan). Garis biru menunjukkan loss dari data pelatihan yang terus menurun, menandakan model semakin baik dalam memprediksi data yang dilihat selama pelatihan. Garis biru dengan titik-titik menunjukkan loss dari data validasi. Jika garis ini stabil atau menurun, menunjukkan model yang general baik terhadap data baru yang tidak dilihat selama proses pelatihan.
2. Training and Validation Accuracy: Grafik ini menunjukkan akurasi model. Garis biru menunjukkan akurasi pelatihan yang meningkat seiring dengan berjalannya epochs, menandakan model semakin tepat dalam mengklasifikasikan data pelatihan. Sementara itu, garis biru dengan titik-titik menunjukkan akurasi pada data validasi. Akurasi pada data validasi yang meningkat menunjukkan bahwa model tidak hanya belajar "menghafal" data pelatihan tetapi juga generalisasi dengan baik ke data yang tidak terlihat sebelumnya.

Kedua grafik ini sangat penting untuk mengevaluasi model dalam hal overfitting. Overfitting terjadi ketika model terlalu sempurna dalam memprediksi data pelatihan namun performanya buruk pada data baru. Idealnya, kedua garis pada kedua grafik pelatihan dan validasi harus mendekati satu sama lain, yang menunjukkan model yang baik dan generalisasi yang baik.

Pada kasus kasus di atas, tampaknya terdapat indikasi baik dari loss yang menurun dan akurasi yang meningkat baik pada data pelatihan maupun validasi, yang menandakan model yang Anda kembangkan berpotensi baik dalam performanya prediksi.

c. Prediksi

```

F1 Score: 0.7639640933244367
Precision: 0.8351513971266455
Recall: 0.7222222222222222

```

	precision	recall	f1-score	support
0	0.82	0.86	0.84	450
1	0.87	0.51	0.64	428
2	0.82	0.79	0.80	472
micro avg	0.83	0.72	0.77	1350
macro avg	0.84	0.72	0.76	1350
weighted avg	0.84	0.72	0.76	1350
samples avg	0.72	0.72	0.72	1350

Gambar 18. Hasil prediksi 70:30

1. Precision: untuk kelas 0, precision 0.82 berarti 82% dari semua prediksi kelas 0 adalah benar.
2. Recall (Sensitivity): untuk kelas 2, recall 0.79 berarti model dapat mengidentifikasi 79% dari semua kasus aktual kelas 2.
3. F1-Score: F1-Score untuk kelas 0 adalah 0.84, menunjukkan keseimbangan yang baik antara precision dan recall.
4. Support: Menunjukkan jumlah sampel sebenarnya untuk setiap kelas yang digunakan untuk menghitung metrik. Kelas 0 memiliki 450 sampel, kelas 1 memiliki 428 sampel, dan kelas 2 memiliki 472 sampel.

Secara keseluruhan, model ini menunjukkan performa yang relatif baik dengan F1 Score keseluruhan 0.77, Precision 0.84, dan Recall 0.72, mengindikasikan model ini efektif dalam memprediksi kelas-kelas yang ada dengan keseimbangan yang cukup antara kemampuan mengidentifikasi kasus positif dan ketepatan prediksinya.

d. Klasifikasi

```
F1 Score: 0.7835929377898251
Precision: 0.7905746005758145
Recall: 0.7881481481481482
```

	precision	recall	f1-score	support
0	0.78	0.89	0.83	450
1	0.81	0.62	0.70	428
2	0.78	0.85	0.81	472
accuracy			0.79	1350
macro avg	0.79	0.78	0.78	1350
weighted avg	0.79	0.79	0.78	1350

Gambar 19. Hasil klasifikasi label 70:30

5. Precision:

- Kelas 0: 0.78 artinya 78% dari prediksi sebagai kelas 0 adalah benar.
- Kelas 1: 0.81 artinya 81% dari prediksi sebagai kelas 1 adalah benar.
- Kelas 2: 0.78 artinya 78% dari prediksi sebagai kelas 2 adalah benar.

6. Recall:

- Kelas 0: 0.89 artinya model dapat mengidentifikasi 89% dari semua kasus aktual kelas 0.
- Kelas 1: 0.62 artinya model dapat mengidentifikasi 62% dari semua kasus aktual kelas 1.
- Kelas 2: 0.85 artinya model dapat mengidentifikasi 85% dari semua kasus aktual kelas 2.

7. F1-Score:

- Kelas 0: 0.83 merupakan rata-rata harmonis dari precision dan recall, menunjukkan keseimbangan yang baik.
- Kelas 1: 0.70 menunjukkan keseimbangan yang lebih rendah antara precision dan recall.
- Kelas 2: 0.81 menunjukkan keseimbangan yang baik.

8. Akurasi Keseluruhan: 0.79 menunjukkan bahwa 79% dari semua prediksi adalah benar.

- CNN

1. Pembagian data 90:10

```
Epoch 1/10 [.....] - 7s 13ms/step - loss: 0.8177 - accuracy: 0.6894 - val_loss: 0.6368 - val_accuracy: 0.7209
Epoch 2/10 [.....] - 7s 10ms/step - loss: 0.4822 - accuracy: 0.8165 - val_loss: 0.6279 - val_accuracy: 0.7356
Epoch 3/10 [.....] - 5s 10ms/step - loss: 0.3165 - accuracy: 0.8886 - val_loss: 0.6814 - val_accuracy: 0.7289
Epoch 4/10 [.....] - 4s 23ms/step - loss: 0.2138 - accuracy: 0.9208 - val_loss: 0.7932 - val_accuracy: 0.7889
Epoch 5/10 [.....] - 5s 10ms/step - loss: 0.1601 - accuracy: 0.9472 - val_loss: 0.8829 - val_accuracy: 0.7111
Epoch 6/10 [.....] - 4s 18ms/step - loss: 0.1188 - accuracy: 0.9687 - val_loss: 1.0978 - val_accuracy: 0.6988
Epoch 7/10 [.....] - 7s 17ms/step - loss: 0.1074 - accuracy: 0.9657 - val_loss: 1.1275 - val_accuracy: 0.6544
Epoch 8/10 [.....] - 4s 28ms/step - loss: 0.0938 - accuracy: 0.9672 - val_loss: 1.1188 - val_accuracy: 0.7111
Epoch 9/10 [.....] - 5s 21ms/step - loss: 0.0868 - accuracy: 0.9726 - val_loss: 1.1492 - val_accuracy: 0.7244
Epoch 10/10 [.....] - 4s 21ms/step - loss: 0.0769 - accuracy: 0.9786 - val_loss: 1.1887 - val_accuracy: 0.7329
```

Gambar 20. Proses epoch 90:10 hanya menggunakan cnn

Pada gambar di atas di dapatkan Validation Accuracy atau hasil training tertinggi yaitu 73%.

2. Pembagian data 80:20

```
Epoch 1/10 [.....] - 8s 13ms/step - loss: 0.6578 - accuracy: 0.6917 - val_loss: 0.6455 - val_accuracy: 0.7200
Epoch 2/10 [.....] - 6s 12ms/step - loss: 0.4649 - accuracy: 0.8176 - val_loss: 0.6888 - val_accuracy: 0.7289
Epoch 3/10 [.....] - 1s 13ms/step - loss: 0.3828 - accuracy: 0.8925 - val_loss: 0.6992 - val_accuracy: 0.7578
Epoch 4/10 [.....] - 8s 26ms/step - loss: 0.3153 - accuracy: 0.9298 - val_loss: 0.7988 - val_accuracy: 0.7178
Epoch 5/10 [.....] - 6s 16ms/step - loss: 0.2544 - accuracy: 0.9439 - val_loss: 0.9492 - val_accuracy: 0.7044
Epoch 6/10 [.....] - 1s 25ms/step - loss: 0.2242 - accuracy: 0.9583 - val_loss: 1.0563 - val_accuracy: 0.6968
Epoch 7/10 [.....] - 5s 18ms/step - loss: 0.2818 - accuracy: 0.9658 - val_loss: 1.1110 - val_accuracy: 0.7089
Epoch 8/10 [.....] - 4s 27ms/step - loss: 0.0852 - accuracy: 0.9786 - val_loss: 1.1559 - val_accuracy: 0.7167
Epoch 9/10 [.....] - 1s 18ms/step - loss: 0.0711 - accuracy: 0.9764 - val_loss: 1.2006 - val_accuracy: 0.7222
Epoch 10/10 [.....] - 4s 16ms/step - loss: 0.0618 - accuracy: 0.9774 - val_loss: 1.2889 - val_accuracy: 0.7111
```

Gambar 21. Proses epoch 80:20 hanya menggunakan cnn

Pada gambar di atas di dapatkan Validation Accuracy atau hasil training tertinggi yaitu 74%.

3. Pembagian data 70:30

```

Epoch 1/30
197/197 [#####] - 5s 22ms/step - loss: 0.8861 - accuracy: 0.5917 - val_loss: 0.8614 - val_accuracy: 0.7253
Epoch 2/30
197/197 [#####] - 6s 39ms/step - loss: 0.4773 - accuracy: 0.8148 - val_loss: 0.8552 - val_accuracy: 0.7181
Epoch 3/30
197/197 [#####] - 6s 21ms/step - loss: 0.2847 - accuracy: 0.8978 - val_loss: 0.7987 - val_accuracy: 0.7213
Epoch 4/30
197/197 [#####] - 4s 22ms/step - loss: 0.1998 - accuracy: 0.9349 - val_loss: 0.7945 - val_accuracy: 0.7181
Epoch 5/30
197/197 [#####] - 6s 29ms/step - loss: 0.1828 - accuracy: 0.8937 - val_loss: 0.8841 - val_accuracy: 0.7079
Epoch 6/30
197/197 [#####] - 4s 22ms/step - loss: 0.1876 - accuracy: 0.9663 - val_loss: 1.0389 - val_accuracy: 0.6954
Epoch 7/30
197/197 [#####] - 6s 27ms/step - loss: 0.8887 - accuracy: 0.8747 - val_loss: 1.1352 - val_accuracy: 0.6928
Epoch 8/30
197/197 [#####] - 4s 22ms/step - loss: 0.8755 - accuracy: 0.9746 - val_loss: 1.1688 - val_accuracy: 0.7148
Epoch 9/30
197/197 [#####] - 6s 28ms/step - loss: 0.8885 - accuracy: 0.9798 - val_loss: 1.2425 - val_accuracy: 0.7222
Epoch 10/30
197/197 [#####] - 5s 23ms/step - loss: 0.8612 - accuracy: 0.9787 - val_loss: 1.3279 - val_accuracy: 0.7181

```

Gambar 22. Proses epoch 70:30 menggunakan cnn saja

Pada gambar di atas di dapatkan Validation Accuracy atau hasil training tertinggi yaitu 73%.

Tabel 5. Hasil perbandingan ke dua model

Pembagian Data	Akurasi	
	Word Embedding CNN	CNN
90:10	77%	73%
80:20	77%	74%
70:30	79%	73%

Dari tabel di atas membandingkan kinerja dua model, yaitu "Word Embedding CNN" dan "CNN", dengan tiga skenario pembagian data yang berbeda: 90:10, 80:20, dan 70:30. Persentase yang ditampilkan merupakan akurasi dari masing-masing model pada setiap skenario pembagian data. Berikut adalah penjelasan dari isi tabel tersebut:

1. Pembagian Data 90:10:

- Word Embedding CNN: Model ini mencapai akurasi sebesar 77%.
- CNN: Model ini mencapai akurasi sebesar 73%.

2. Pembagian Data 80:20:

- Word Embedding CNN: Model ini mencapai akurasi sebesar 77%.

- CNN: Model ini mencapai akurasi sebesar 74%.

3. Pembagian Data 70:30:

- Word Embedding CNN: Model ini mencapai akurasi sebesar 79%.
- CNN: Model ini mencapai akurasi sebesar 73%.

Dari tabel tersebut, dapat dilihat bahwa model "Word Embedding CNN" secara konsisten menunjukkan kinerja yang lebih baik dibandingkan dengan model "CNN" pada semua skenario pembagian data yang diuji. Akurasi tertinggi yang dicapai oleh "Word Embedding CNN" adalah 79% pada pembagian data 70:30, sedangkan akurasi tertinggi untuk model "CNN" adalah 74%, yang juga pada pembagian data 80:20.



BAB V

PENUTUP

A. Kesimpulan

Penelitian ini berhasil mengeksplorasi penggunaan Word Embedding Word2Vec dalam pengembangan model Convolutional Neural Network (CNN) untuk analisis sentimen pada tempat wisata di Makassar dan menunjukkan hasil yang signifikan. Penggunaan Word2Vec meningkatkan performa model CNN, dengan akurasi tertinggi mencapai 79% dibandingkan dengan 74% pada model tanpa Word2Vec. Dengan data sebanyak 4500 sampel, model CNN yang diintegrasikan dengan Word2Vec memperlihatkan kinerja yang lebih baik dalam mengklasifikasikan sentimen, menandakan bahwa representasi kata oleh Word2Vec dapat menangkap makna dan konteks dengan lebih efektif dibandingkan pendekatan tradisional. Studi kasus ini juga menunjukkan bahwa teknologi Word Embedding dan model deep learning seperti CNN dapat diterapkan dalam analisis sentimen di berbagai domain, termasuk pariwisata, memberikan wawasan yang lebih akurat tentang persepsi dan opini publik. Penelitian ini diharapkan dapat menjadi referensi bagi penelitian selanjutnya dalam menggabungkan teknik serupa untuk meningkatkan akurasi analisis sentimen.

B. Saran

Berdasarkan hasil penelitian yang telah dilakukan, terdapat saran yang dapat dipertimbangkan untuk penelitian selanjutnya. Penggunaan Teknik Word Embedding Lain Selain Word2Vec, terdapat berbagai teknik word embedding lain seperti GloVe, FastText, dan BERT yang mungkin dapat memberikan hasil yang lebih baik. Penelitian lebih lanjut dapat membandingkan kinerja berbagai teknik tersebut dalam pengembangan model CNN untuk analisis sentimen.

DAFTAR PUSTAKA

- Adityarini, E., Nur Ayuni, S., & Aminatus Sa'diah, R. (2021). ANALISIS SENTIMEN TERHADAP ULASAN PRODUK PADA SISTEM PENJUALAN TOKO PUTRA ELEKTRONIK. *Journal of Islamic Business Management Studies (JIBMS)*, 2(2), 84–98. <https://doi.org/10.51875/jibms.v2i2.184>
- Afidah, D. I., Dairoh, D., Handayani, S. F., & Pratiwi, R. W. (2021). Pengaruh parameter word2vec terhadap performa deep learning pada klasifikasi sentimen. *Jurnal Informatika: Jurnal Pengembangan IT*, 6(3), 156-161.
- Agung, B. A. I. G. N. (2023). Implementasi Deep Learning untuk ImageClasification menggunakan Arsitektur Convolutional Neural Network (CNN) pada Citra Sampah Hotel (Studi Kasus: Hotel <http://eprints.unram.ac.id/id/eprint/41624>) <http://eprints.unram.ac.id/4162>
- Akib, E. (2020). Pariwisata Dalam Tinjauan Pendidikan: Studi Menuju Era Revolusi Industri. *PUSAKA (Journal of Tourism, Hospitality, Travel and Business Event)*, 2(1), 1–7. <https://doi.org/10.33649/pusaka.v2i1.40>
- Amalia, P. R. (2021). Analisis Sentimen Berdasarkan Aspek Pada Ulasan Restoran Berbahasa Indonesia Menggunakan Kombinasi Convolutional Neural Network (CNN) dan Contextualized Word Embedding (Doctoral dissertation, Universitas Gadjah Mada).

- Arsi, P., & Waluyo, R. (2021). Analisis Sentimen Wacana Pemindahan Ibu Kota Indonesia Menggunakan Algoritma Support Vector Machine (SVM). *Jurnal Teknologi Informasi dan Ilmu Komputer*, 8(1), 147. <https://doi.org/10.25126/jtiik.0813944>
- Bahits, A., Komarudin, M. F., & Afriani, R. I. (2020). STRATEGI PENGEMBANGAN TEMPAT WISATA RELIGI UNTUK MENINGKATKAN PEREKONOMIAN MASYARAKAT DI GUNUNG SANTRI DESA BOJONEGARA KECAMATAN BOJONEGARA KABUPATEN SERANG BANTEN. *Jurnal Manajemen STIE Muhammadiyah Palopo*, 6(2), 55. <https://doi.org/10.35906/jm001.v6i2.593>
- Dinata, R. K., Hasdyna, N., & Azizah, N. (2020). Analisis K-Means Clustering pada Data Sepeda Motor. 5(1).
- Diponegoro, Sri Suning Kusumawardani, & Indriana Hidayah. (2021). Tinjauan Pustaka Sistematis: Implementasi Metode Deep Learning pada Prediksi Kinerja Murid. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi*, 10(2), 131–138. <https://doi.org/10.22146/jnteti.v10i2.1417>
- HELDIANSYAH, M. F. (2022). DETEKSI EMOSI PADA TWEET DENGAN MENGGABUNGAN CONTEXTUALIZED WORD EMBEDDING DAN CONVOLUTIONAL NEURAL NETWORK (CNN) (Doctoral dissertation, Universitas Gadjah Mada).
- Hermanto, D. T., Setyanto, A., & Luthfi, E. T. (2021). Algoritma LSTM-CNN untuk Sentimen Klasifikasi dengan Word2vec pada Media Online. 8(1).

- Jihad, M. A. A., Adiwijaya, A., & Astut, W. (2021). Analisis sentimen terhadap ulasan film menggunakan algoritma random forest. *eProceedings of Engineering*, 8(5).
- Khesya, N. (2021). *MENGENAL FLOWCHART DAN PSEUDOCODE DALAM ALGORITMA DAN PEMROGRAMAN*.
- Khomsah, S. (2021). Sentiment Analysis On YouTube Comments Using Word2Vec and Random Forest. *Telematika*, 18(1), 61. <https://doi.org/10.31315/telematika.v18i1.4493>
- Kristiawan, K., & Widjaja, A. (2021). Perbandingan Algoritma Machine Learning dalam Menilai Sebuah Lokasi Toko Ritel. *Jurnal Teknik Informatika dan Sistem Informasi*, 7(1). <https://doi.org/10.28932/jutisi.v7i1.3182>
- Manalu, D. A., & Gunadi, G. (2022). IMPLEMENTASI METODE DATA MINING K-MEANS CLUSTERING TERHADAP DATA PEMBAYARAN TRANSAKSI MENGGUNAKAN BAHASA PEMROGRAMAN PYTHON PADA CV DIGITAL DIMENSI. *Infotech: Journal of Technology Information*, 8(1), 43–54. <https://doi.org/10.37365/jti.v8i1.131>
- Manalu, R., & Fikri, A. (2021). *INNOVATIVE: Volume 1 Nomor 2 Tahun 2021 Research & Learning in Primary Education*.
- Naquitasia. (2021). *ANALISIS SENTIMEN BERBASIS ASPEK PADA WISATA HALAL DENGAN DEEP LEARNING*.
- Ningsih, S. R., Hartama, D., Wanto, A., & Parlina, I. (2019). *Penerapan Sistem Pendukung Keputusan Pada Pemilihan Objek Wisata di Simalungun*.

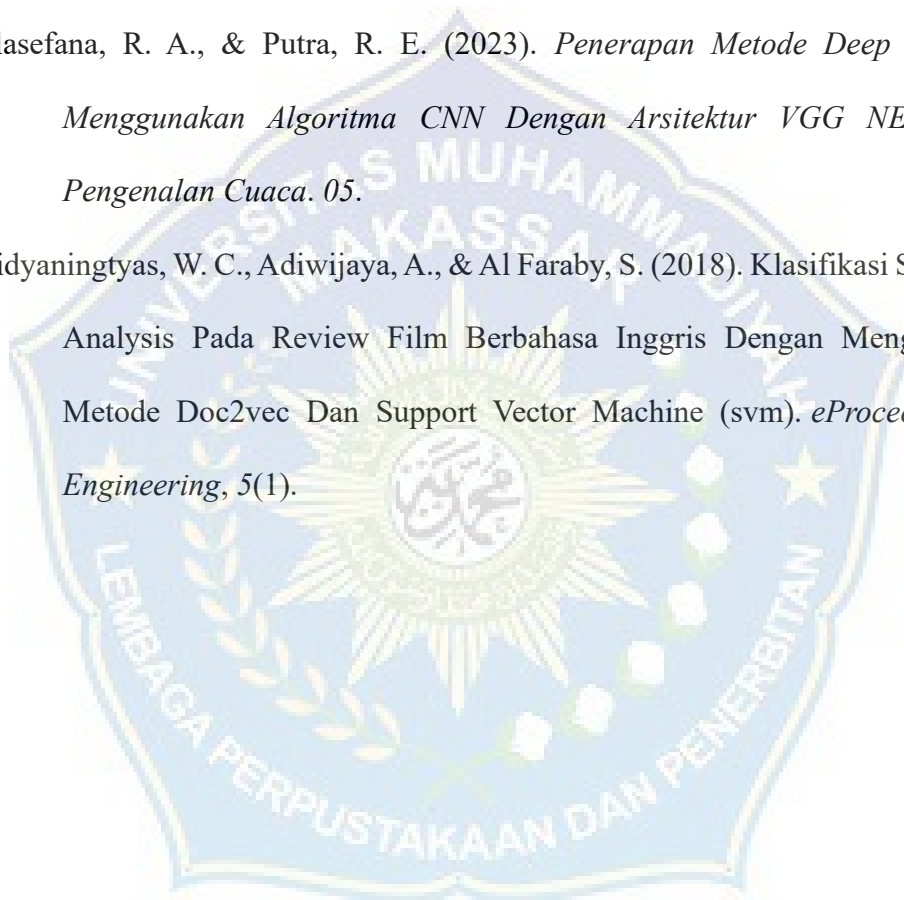
- Nurdin, A., Aji, B. A. S., Bustamin, A., & Abidin, Z. (2020). Perbandingan Kinerja Word Embedding Word2Vec, Glove, Dan Fasttext Pada Klasifikasi Teks. *Jurnal Tekno Kompak*, 14(2), 74-79.
- Pelham, I. (2023). Erd2. Secretary Pathway, 5, 135–135.
<https://doi.org/10.1093/oso/9780198599425.003.0085>
- Peryanto, A., Yudhana, A., & Umar, R. (2020). Rancang Bangun Klasifikasi Citra Dengan Teknologi Deep Learning Berbasis Metode Convolutional Neural Network. *Format: Jurnal Ilmiah Teknik Informatika*, 8(2), 138.
<https://doi.org/10.22441/format.2019.v8.i2.007>
- Rachman, F. P., & Santoso, H. (2021). Perbandingan Model Deep Learning untuk Klasifikasi Sentiment Analysis dengan Teknik Natural Language Processing. *Jurnal Teknologi dan Manajemen Informatika*, 7(2), 113–121.
<https://doi.org/10.26905/jtmi.v7i2.6506>
- R.H. Zer, P. P. P. A. N. W. F. I., Hayadi, B. H., & Damanik, A. R. (2022). PENDEKATAN MACHINE LEARNING MENGGUNAKAN ALGORITMA C4.5 BERBASIS PSO DALAM ANALISA PEMAHAMAN PEMROGRAMAN WEBSITE. *Jurnal Informatika dan Teknik Elektro Terapan*, 10(3). <https://doi.org/10.23960/jitet.v10i3.2700>
- Samsir, S., Ambiyar, A., Verawardina, U., Edi, F., & Watrianthos, R. (2021). Analisis Sentimen Pembelajaran Daring Pada Twitter di Masa Pandemi COVID-19 Menggunakan Metode Naïve Bayes. *JURNAL MEDIA INFORMATIKA BUDIDARMA*, 5(1), 157.
<https://doi.org/10.30865/mib.v5i1.2580>

Silitonga, Y. R. (2019). *SISTEM PENDETEKSI BERITA HOAX DI MEDIA SOSIAL DENGAN TEKNIK DATA MINING SCIKIT LEARN. 4.*

Subowo, E., & Ribowo, T. J. (2021). SENTIMENT EMBEDDINGS WORD2VEC PADA KLASIFIKASI KEPUASAN KARYAWAN PADA MANAJEMEN RTO GROUP. *Jurnal Surya Informatika, 11(1), 35-39.*

Tilasefana, R. A., & Putra, R. E. (2023). *Penerapan Metode Deep Learning Menggunakan Algoritma CNN Dengan Arsitektur VGG NET Untuk Pengenalan Cuaca. 05.*

Widyaningtyas, W. C., Adiwijaya, A., & Al Faraby, S. (2018). Klasifikasi Sentiment Analysis Pada Review Film Berbahasa Inggris Dengan Menggunakan Metode Doc2vec Dan Support Vector Machine (svm). *eProceedings of Engineering, 5(1).*



LAMPIRAN

Lampiran 1. Source code

```
!pip install openpyxl

import pandas as pd
import numpy as np
from tqdm import tqdm
from keras.preprocessing.text import Tokenizer
tqdm.pandas(desc="progress-bar")
from gensim.models import Doc2Vec
from sklearn import utils
from sklearn.model_selection import train_test_split
from keras.preprocessing.sequence import pad_sequences
import gensim
from sklearn.linear_model import LogisticRegression
from gensim.models.doc2vec import TaggedDocument
import re
import seaborn as sns
import matplotlib.pyplot as plt

df = pd.read_excel('/content/DATABARU-1.xlsx', sheet_name="Sheet1")
df = df[['ULASAN', 'LABEL']]
df = df[pd.notnull(df['ULASAN'])]
df.rename(columns={'ULASAN': 'ULASAN'}, inplace=True)

df.head()

df.shape

df.index = range(len(df))
total_words = df['ULASAN'].apply(lambda x: len(x.split(' '))).sum()
print("Total jumlah kata dalam semua ulasan:", total_words)

cnt_pro = df['LABEL'].value_counts()
plt.figure(figsize=(12, 4))
sns.barplot(x=cnt_pro.index, y=cnt_pro.values, alpha=0.8)
plt.ylabel('Jumlah Kemunculan', fontsize=12)
plt.xlabel('LABEL', fontsize=12)
plt.xticks(rotation=90)
plt.show()
```

```

def print_message(index):
    example = df.iloc[index][['ULASAN', 'LABEL']].values
    if len(example) > 0:
        print('ULASAN:', example[0])
        print('LABEL:', example[1])
print_message(12)

import string
def remove_punctuation(text):
    return text.translate(str.maketrans('', '',
string.punctuation))
df['ULASAN'] = df['ULASAN'].apply(remove_punctuation)

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
train, test = train_test_split(df, test_size=0.3,
random_state=42)
train_tagged = train.apply(
    lambda r: TaggedDocument(words=tokenize_text(r['ULASAN']),
tags=[r.LABEL]), axis=1)
test_tagged = test.apply(
    lambda r: TaggedDocument(words=tokenize_text(r['ULASAN']),
tags=[r.LABEL]), axis=1)
max_features = 500000
max_sequence_length = 50
tokenizer = Tokenizer(num_words=max_features, split=' ',
filters='!"#$%&()*+,-./:;<=>@[\\]^_`{|}~', lower=True)
tokenizer.fit_on_texts(df['ULASAN'].values)
X_train = tokenizer.texts_to_sequences(train['ULASAN'].values)
X_train = pad_sequences(X_train, maxlen=max_sequence_length)
X_test = tokenizer.texts_to_sequences(test['ULASAN'].values)
X_test = pad_sequences(X_test, maxlen=max_sequence_length)
print('Found %s unique tokens.' % len(tokenizer.word_index))

X = tokenizer.texts_to_sequences(df['ULASAN'].values)
X = pad_sequences(X, maxlen=max_sequence_length)
print('Shape dari data tensor:', X.shape)

```



```

train_tagged.values

from gensim.models.doc2vec import Doc2Vec, TaggedDocument
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)
train_tagged =
[TaggedDocument(words=tokenize_text(row['ULASAN']),
tags=[row['LABEL']]) for index, row in train.iterrows()]
d2v_model.build_vocab(train_tagged)

d2v_model.build_vocab(train_tagged)
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

print(d2v_model)

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)

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])

from keras.models import Sequential
from keras.layers import Conv1D, GlobalMaxPooling1D, Dense,
Embedding, Dropout
MAX_SEQUENCE_LENGTH = 50
num_unique_words = len(tokenizer.word_index) + 1
embedding_matrix = np.random.rand(num_unique_words, 20)
model = Sequential()
model.add(Embedding(num_unique_words, 20,
input_length=MAX_SEQUENCE_LENGTH, weights=[embedding_matrix],
trainable=True))

```

```

model.add(Conv1D(50, 3, activation='relu'))
model.add(Dropout(0.5))
model.add(GlobalMaxPooling1D())
model.add(Dense(3, activation="softmax"))
model.summary()
model.compile(optimizer="adam",
loss="categorical_crossentropy", metrics=['acc'])
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)

Y = pd.get_dummies(df['LABEL']).values
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.3, random_state=42)
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)

history = model.fit(X_train, Y_train, epochs=10,
batch_size=16, verbose=2, validation_data=(X_test, Y_test))
print(history.history.keys())
val_loss = history.history['val_loss']
val_acc = history.history['val_acc']
print("Validation Loss:", val_loss)
print("Validation Accuracy:", val_acc)

model.save('/content/drive/MyDrive/Skripsi/2020/Arvianda -
Word2Vec CNN/CNN_W2V.h5')

from sklearn.metrics import classification_report, f1_score,
precision_score, recall_score
predictions = model.predict(X_test)
predicted_labels = (predictions > 0.5).astype(int)
true_labels = Y_test
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')

```

```

print(f"F1 Score: {f1}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(classification_report(true_labels, predicted_labels))
print("Array hasil prediksi:")
print(true_labels)
print(predicted_labels)

print("Panjang Tes Ulasan:", len(test['ULASAN']))
print("Panjang X_test:", len(X_test))
print("Panjang Y_test:", len(Y_test))
print("Panjang true_labels:", len(true_labels))
print("Panjang predicted_labels:", len(predicted_labels))

import pandas as pd
from sklearn.metrics import classification_report, f1_score,
precision_score, recall_score
predictions = model.predict(X_test)
predicted_labels = predictions.argmax(axis=1)
true_labels = Y_test.argmax(axis=1)
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')
print(f"F1 Score: {f1}")
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(classification_report(true_labels, predicted_labels))
min_length = min(len(test['ULASAN']), len(true_labels),
len(predicted_labels))
test_results = pd.DataFrame({
    'Ulasan': test['ULASAN'].values[:min_length],
    'Label Sebenarnya': true_labels[:min_length],
    'Prediksi': predicted_labels[:min_length]
})
def classify_label(label):
    if label == 0:
        return 'Negatif'
    elif label == 1:
        return 'Positif'
    else:
        return 'Netral'

```


Lampiran 3. Dataset ulasan negatif

1502 Selama begitu kotor (jumlah dia hotel air panas), air ledh jernih, kawat banyak, cepat masak ggi tidak efektif (juga anak terdudukkan) dalam yg baik cukup tenang	Negatif
1503 Tidak ada yang istimewa.	Negatif
1504 Tidak lagi dihi ke pantai karena	Negatif
1505 sangat menyenangkan ... tidak boleh makan	Negatif
1506 bakunya tidak enak :)	Negatif
1507 Sekadar makan, makanannya ini memang tidak enak...	Negatif
1508 Muchlisah sangat Buruk!	Negatif
1509 Hari Weekend sangat ramai parkir, tempat parkir jadi sempit.	Negatif
1510 Kualitasnya sangat buruk.	Negatif
1511 Menu makan sangat mahal.	Negatif
1512 Harga makanannya buat mangrove dangkal	Negatif
1513 ★ Rasanya (itu makan) sangat enak (RUMAH JEM. Dulu yang mau santai kating keluarga, jadi makan-makan di gazeboya tidak stress mandikan air, mahal banget)	Negatif
1514 Desainnya tidak menarik, pelayanannya sudah garing-garing seperti mau berubah gitu... warnanya putih juga sudah lingsir ★ ★	Negatif
1515 Memang sih enak banget. Aku sangat amat tidak merekomendasikan tempat ini untuk berwisata pantai. Airnya kotor.	Negatif
1516 Halaman sekitar pantai (di luar yang agak sedikit berpasir) sangat berdebu.	Negatif
1517 Tidak ada pemandu/pemandu yang bagus. Karcis masuk juga mahal. Kolombnya jelek seperti jebakan.	Negatif
1518 Tempatnya jadi lebih cozy untuk santai/ nongkrong bersama teman dan keluarga. Sayangnya air lautnya kotor, jadi fungsi rekreasinya berkurang.	Negatif
1519 Berkualitasnya harga tidak murah dibanding apa yang ada di dalam area wisata/kegiatan wisata.	Negatif
1520 Tidak memberikan rasa kurang, dan perawatan garba berkelas. Sangat enak!	Negatif
1521 Like banget hari libur, obrol sopi, tidak ada, mdu. Airnya memang tidak ada sambek karena di bidanya dibikin beronggong peminahan aneh.	Negatif
1522 Tidak terasa dehutan ombaknya. Cocoknya karena sudah cukup baik!	Negatif
2226 Tidak begitu menarik.	Negatif
2240 Selain rasanya kurang bagus dan tidak istimewa.	Negatif
2241 Selain rasanya tidak enak.	Negatif
2242 Tidak terlalu bersih, rimang, gila-jalan.	Negatif
2243 Yang penting enak-enak, jadi memang enak, jadi kapan kapan saja memang enak!	Negatif
2244 Tidak ada daya tarik yang tempat wisata ini.	Negatif
2245 Tidak terlalu menarik.	Negatif
2246 Bona pergi ke sini karena rasanya itu kurang enak.	Negatif
2247 Sarana fasilitas yang sudah tua tidak berkualitas.	Negatif
2248 Harga tidak sesuai dengan yang ada.	Negatif
2249 Tidak menarik untuk Kluar & Zulfawad SDP BCPB/PD USM.	Negatif
2250 Tempat yang tidak bagus rasanya tidak enak.	Negatif
2251 Tidak banyak aktivitas untuk anak-anak yang ada.	Negatif
2252 Tidak memberikan pengalaman yang menyenangkan.	Negatif
2253 Tidak begitu seru bersama keluarga, lebih seru dengan teman-teman yang ada di rumah.	Negatif
2254 Wah ada kebun binatangnya tidak menarik...	Negatif
2255 Rungutan perlintas tidak jelas bunyinya.	Negatif
2256 Tidak begitu menarik dengan suasana yang ada.	Negatif
2257 Berat banget dan banyak yang rusak.	Negatif
2258 Airnya kurang bersih dan agak bau.	Negatif
2259 Tidak enak untuk alamnya.	Negatif
2260 Tidak bagus. Tidak memberikan pengalaman seperti di Makasar.	Negatif
2865 Sa lah satu dekapan, yaitu pantai di Sulawesi Selatan. Ini tidak begitu menarik.	Negatif
2866 Tempat tersebut tidak bersih seperti yang diharapkan.	Negatif
2867 Tidak ada yang spesial, tidak ada yang menarik.	Negatif
2868 Tempat ini luas, pengalihan, dan tidak layak dikunjungi.	Negatif
2869 tempatnya tidak bersih.	Negatif
2870 Tempat ini tidak cocok untuk wisata bersama keluarga.	Negatif
2871 Tidak begitu menarik, kurang menarik perhatian.	Negatif
2872 Tempat pemanduannya itu ini tidak sesuai dengan baik.	Negatif
2873 Ornamennya tidak indah dan tidak menarik.	Negatif
2874 Lokasinya tidak terlalu menarik, sebaiknya ini bisa jadi lebih menarik.	Negatif
2875 Tempat tersebut tidak menjadi tempat yang menarik untuk berkunjung setelah ini.	Negatif
2876 Seragamnya pemanduannya dengan banyak, jadi baik.	Negatif
2877 Pengembangannya masih kurang dan tidak menarik.	Negatif
2878 Tempat wisata ini tidak begitu menarik dan kurang memuaskan.	Negatif
2879 Sangat enak sudah bisa di rumah, pengalaman tidak begitu memuaskan.	Negatif
2880 Tidak cocok untuk wisata, tidak ada spot yang baik.	Negatif
2881 Amanan baik.	Negatif
2882 Wah ada salah satu yang bisa dimakan, ada waktu waktu bertemu waktu bisa berpetting.	Negatif
2883 (selama) membayar di sana.	Negatif
2884 Tempat yang nyaman untuk weekend, jadi pelayanan kurang memuaskan.	Negatif
2885 Wah ada terumbu karang karapasional di sini, tidak bisa digunakan sepenuhnya.	Negatif
2886 Tidak ada bau.	Negatif

Lampiran 4. Dataset ulasan netral

3002 Tempat parkir berbeda dimana kemana bisa jalanin lagi nih parkir.	Netral
3003 Alas jalan yang cukup baik, terletak di jalan peron Malina, dalam Pakanda. Harga tiket 20 ribu untuk hari Minggu dan Sabtu, dan 10 ribu untuk hari.	Netral
3004 Dedaunnya terdapat banyak kolam renang, terdapat teman-teman yang indah.	Netral
3005 Banyak wahana 5rb untuk anak 7 tahun ke atas, 4 ribu untuk gratis.	Netral
3006 Hari ini Sabtu 20 rdt berkunjung ke wisata.	Netral
3007 Dulu sempat diadukan pengantaraan untuk acara2 seperti ini bisa.	Netral
3008 sangat menarik terdapat untuk kawasan wisata.	Netral
3009 Tempat ini salah satu wahana wisata yang banyak dikunjungi warga Makassar dan Gowa saat akhir pekan.	Netral
3010 Benda.	Netral
3011 Sialandi ini keluarnya sangat lancar sehingga kejalanannya sangat tenang.	Netral
3012 Alami jalan baik. Harga tiket terjangkau.	Netral
... sangat nyaman, polikanya sangat bersih, sangat luas diparkir lagi, akses terus wisata keban. Terus sudah telah menyediakan tempat rekreasi untuk keluarga.	Netral
3013	Netral
3014 Sore ...	Netral
3015 overall sangat nyi keren.	Netral
3016 Banyak tempat berteduh. area bermain.	Netral
3017 Sore nihh. Pindah lokasi hari sabtu minggu tp rame. Admin baik pdkt. Kabin bersih. Cai agak dingin lagi yng dingin. Tj tetap seru.	Netral
3018 Pihak pengelola sebaiknya menyediakan jasa bus penumpang disekitar.	Netral
3019 A'Khalid ya.	Netral
3020 air bersih ukuran dalam kolam bisa bisa.	Netral
3021 Ada Wasthouse nya juga keren.	Netral
3022 memantapkan wahana nya.	Netral
3737 Rantai di mekanis, sehingga sudah enak juga juga juga. Tapi tempat yang baik itu juga sudah kering di Pantai.	Netral
3738 Layanan yang kurang, sangat nyi keren juga.	Netral
3739 Rantai untuk bermain dan ke rumah nya tidak seru.	Netral
3740 Rantai yang bagus tapi sayang kabinnya.	Netral
3741 Rantai yang baru diperbaiki kabinnya.	Netral
3742 anak di kabin foto di foto ng foto nya.	Netral
3743 bisa suka, tapi kabinnya sudah rusak.	Netral
3744 Rantai ini sudah siap untuk bermain, dimana mana.	Netral
3745 sudah bisa di mainkan untuk para wisatawan.	Netral
3746 Ada bisa gubuknya di hilangkan.	Netral
3747 Rantai.	Netral
3748 kabin yang rusak nihh.	Netral
3749 Masih kurang pengaman juga.	Netral
3750 Ada beberapa terdapat.	Netral
3751 Layanan yang cukup baik.	Netral
3752 Luas juga.	Netral
3753 Rantai.	Netral
3754 Banyak wahana permainan.	Netral
3755 Sore.	Netral
3756 mesin sudah rusak. bisa dengan mesin lain supaya lancar.	Netral
3757 karena untuk mengantar di tempat ini para wisatawan, tempat ini tidak rusak.	Netral
3758 Tempat ini bagus jika dipelihara dengan baik.	Netral
3759 Bagus tapi kebetulan ke sini.	Netral
4482 Meskipun tempatnya bersih, perlu diperhatikan lagi tempat ini sebagai hiburan karena banyak wahana yang rusak.	Netral
... Saya menguji lagi pantai ini saat libur liburan dan menikmati kondisi yang sangat ramai, ramai, di luar hari libur besar, sebenarnya tidak ada masalah. Selain itu, untuk	Netral
4483 kegiatan hiburan, bisa menyebarkan dari pantai ini ke beberapa pantai terdekat lainnya.	Netral
4484 Tempat ini memiliki pantai yang sangat indah dan nyaman untuk bermain.	Netral
4485 Pantai ini memiliki keindahan yang bagus. Pengunjung juga cukup memadai dan bersih. Sangat direkomendasikan.	Netral
4486 Selain itu, pantai ini memiliki fasilitas yang sangat penting untuk menjaga keindahannya agar pengunjung juga bisa menikmati.	Netral
4487 keindahan yang luar biasa bisa dimanfaatkan di sini.	Netral
4488 Pantai ini terletak di ujung pulau yang cantik, namun masalah sampah masih menjadi perhatian utama.	Netral
... Diperlukan pembangunan fasilitas seperti tempat sampah di sekitar pantai untuk kenyamanan pengunjung, serta pemeliharaan yang lebih baik bagi para pedagang agar	Netral
4489 pantainya tetap bersih dan indah.	Netral
... Pantai ini memiliki pasir putih yang bagus dan air laut yang bersih, meskipun masih terdapat beberapa sampah yang terdapat di sekitar pantai ini, dan bisa juga disediakan yang bisa mengantar pengunjung ke pulau, tapi harga parkir yang lebih baik dari pemerintah untuk menarik lebih banyak	Netral
4490 pengunjung.	Netral
4491 Untuk mengunjungi tempat ini sebaiknya datang dengan kendaraan pribadi karena lokasi sangat jauh untuk bus. Selain itu, lokasi ini juga.	Netral
4492 Pantai ini memiliki pasir putih yang halus dan indah bersih, serta banyak wahana permainan yang seru.	Netral
4493 Meskipun bagus, namun parkir, banyak pengunjung yang parkir di pantai ini.	Netral
4494 Dari segi keindahan, pantai ini menawarkan nilai 5 dari 10, namun untuk menikmati fasilitas di sini, pengunjung harus membawa diri sendiri.	Netral
4495 Pantai ini memiliki pasir putih yang indah. Banyak pedagang dan tempat parkir yang memadai, serta tersedia pengalangan dan keamanan di sekitarnya.	Netral
4496 Pantai ini memiliki pasir putih yang sangat bagus.	Netral
4497 Tempat ini sangat ramai setiap liburan. Mengingat lebih baik jika mengunjungi saat libur besar.	Netral
4498 Harga tiket masuk yang dibebankan oleh pengelola tidak sesuai dengan yang tertera di papan informasi.	Netral
4499 Meskipun bersih dan bagus, akan saja merasa pantai ini cukup membosankan.	Netral
4500 Pantai ini sangat bersih dan pasarnya banyak.	Netral

Lampiran 6. Proses epoch

```
Epoch 1/10
254/254 - 3s - loss: 1.0727 - acc: 0.4321 - val_loss: 1.0183 - val_acc: 0.5511 - 3s/epoch - 13ms/step
Epoch 2/10
254/254 - 1s - loss: 0.9074 - acc: 0.5822 - val_loss: 0.9810 - val_acc: 0.6289 - 1s/epoch - 5ms/step
Epoch 3/10
254/254 - 1s - loss: 0.7580 - acc: 0.6647 - val_loss: 0.7949 - val_acc: 0.6778 - 1s/epoch - 4ms/step
Epoch 4/10
254/254 - 1s - loss: 0.6337 - acc: 0.7402 - val_loss: 0.7167 - val_acc: 0.7133 - 1s/epoch - 4ms/step
Epoch 5/10
254/254 - 1s - loss: 0.5460 - acc: 0.7847 - val_loss: 0.6506 - val_acc: 0.7467 - 1s/epoch - 5ms/step
Epoch 6/10
254/254 - 1s - loss: 0.4704 - acc: 0.8217 - val_loss: 0.6040 - val_acc: 0.7489 - 1s/epoch - 4ms/step
Epoch 7/10
254/254 - 1s - loss: 0.4163 - acc: 0.8373 - val_loss: 0.5806 - val_acc: 0.7667 - 1s/epoch - 5ms/step
Epoch 8/10
254/254 - 1s - loss: 0.3639 - acc: 0.8662 - val_loss: 0.5643 - val_acc: 0.7689 - 1s/epoch - 5ms/step
Epoch 9/10
254/254 - 1s - loss: 0.3268 - acc: 0.8805 - val_loss: 0.5449 - val_acc: 0.7756 - 1s/epoch - 4ms/step
Epoch 10/10
254/254 - 1s - loss: 0.2947 - acc: 0.8931 - val_loss: 0.5414 - val_acc: 0.7733 - 1s/epoch - 6ms/step

Epoch 1/10
225/225 - 2s - loss: 1.1045 - acc: 0.3819 - val_loss: 1.0467 - val_acc: 0.4911 - 2s/epoch - 9ms/step
Epoch 2/10
225/225 - 1s - loss: 1.0048 - acc: 0.4961 - val_loss: 0.9634 - val_acc: 0.5856 - 1s/epoch - 5ms/step
Epoch 3/10
225/225 - 1s - loss: 0.8547 - acc: 0.6147 - val_loss: 0.8553 - val_acc: 0.6656 - 1s/epoch - 5ms/step
Epoch 4/10
225/225 - 1s - loss: 0.7132 - acc: 0.6953 - val_loss: 0.7547 - val_acc: 0.7122 - 1s/epoch - 5ms/step
Epoch 5/10
225/225 - 1s - loss: 0.6015 - acc: 0.7551 - val_loss: 0.6871 - val_acc: 0.7411 - 1s/epoch - 5ms/step
Epoch 6/10
225/225 - 1s - loss: 0.5143 - acc: 0.7986 - val_loss: 0.6323 - val_acc: 0.7633 - 2s/epoch - 6ms/step
Epoch 7/10
225/225 - 2s - loss: 0.4506 - acc: 0.8258 - val_loss: 0.6024 - val_acc: 0.7611 - 2s/epoch - 8ms/step
Epoch 8/10
225/225 - 1s - loss: 0.3858 - acc: 0.8558 - val_loss: 0.5936 - val_acc: 0.7689 - 1s/epoch - 5ms/step
Epoch 9/10
225/225 - 1s - loss: 0.3501 - acc: 0.8731 - val_loss: 0.5780 - val_acc: 0.7600 - 1s/epoch - 5ms/step
Epoch 10/10
225/225 - 1s - loss: 0.3084 - acc: 0.8847 - val_loss: 0.5507 - val_acc: 0.7711 - 1s/epoch - 5ms/step

Epoch 1/10
197/197 - 2s - loss: 1.1254 - acc: 0.3784 - val_loss: 1.0412 - val_acc: 0.5104 - 2s/epoch - 10ms/step
Epoch 2/10
197/197 - 1s - loss: 0.9911 - acc: 0.5111 - val_loss: 0.9490 - val_acc: 0.6170 - 1s/epoch - 6ms/step
Epoch 3/10
197/197 - 2s - loss: 0.8617 - acc: 0.6210 - val_loss: 0.8563 - val_acc: 0.6756 - 2s/epoch - 8ms/step
Epoch 4/10
197/197 - 1s - loss: 0.7430 - acc: 0.6886 - val_loss: 0.7531 - val_acc: 0.7356 - 1s/epoch - 7ms/step
Epoch 5/10
197/197 - 1s - loss: 0.6438 - acc: 0.7273 - val_loss: 0.6920 - val_acc: 0.7467 - 972ms/epoch - 5ms/step
Epoch 6/10
197/197 - 1s - loss: 0.5603 - acc: 0.7803 - val_loss: 0.6417 - val_acc: 0.7585 - 1s/epoch - 5ms/step
Epoch 7/10
197/197 - 1s - loss: 0.4846 - acc: 0.8095 - val_loss: 0.6162 - val_acc: 0.7644 - 1s/epoch - 5ms/step
Epoch 8/10
197/197 - 1s - loss: 0.4424 - acc: 0.8111 - val_loss: 0.6207 - val_acc: 0.7652 - 990ms/epoch - 5ms/step
Epoch 9/10
197/197 - 1s - loss: 0.3778 - acc: 0.8527 - val_loss: 0.5573 - val_acc: 0.7815 - 978ms/epoch - 5ms/step
Epoch 10/10
197/197 - 1s - loss: 0.3329 - acc: 0.8733 - val_loss: 0.5464 - val_acc: 0.7881 - 1s/epoch - 5ms/step
```



```

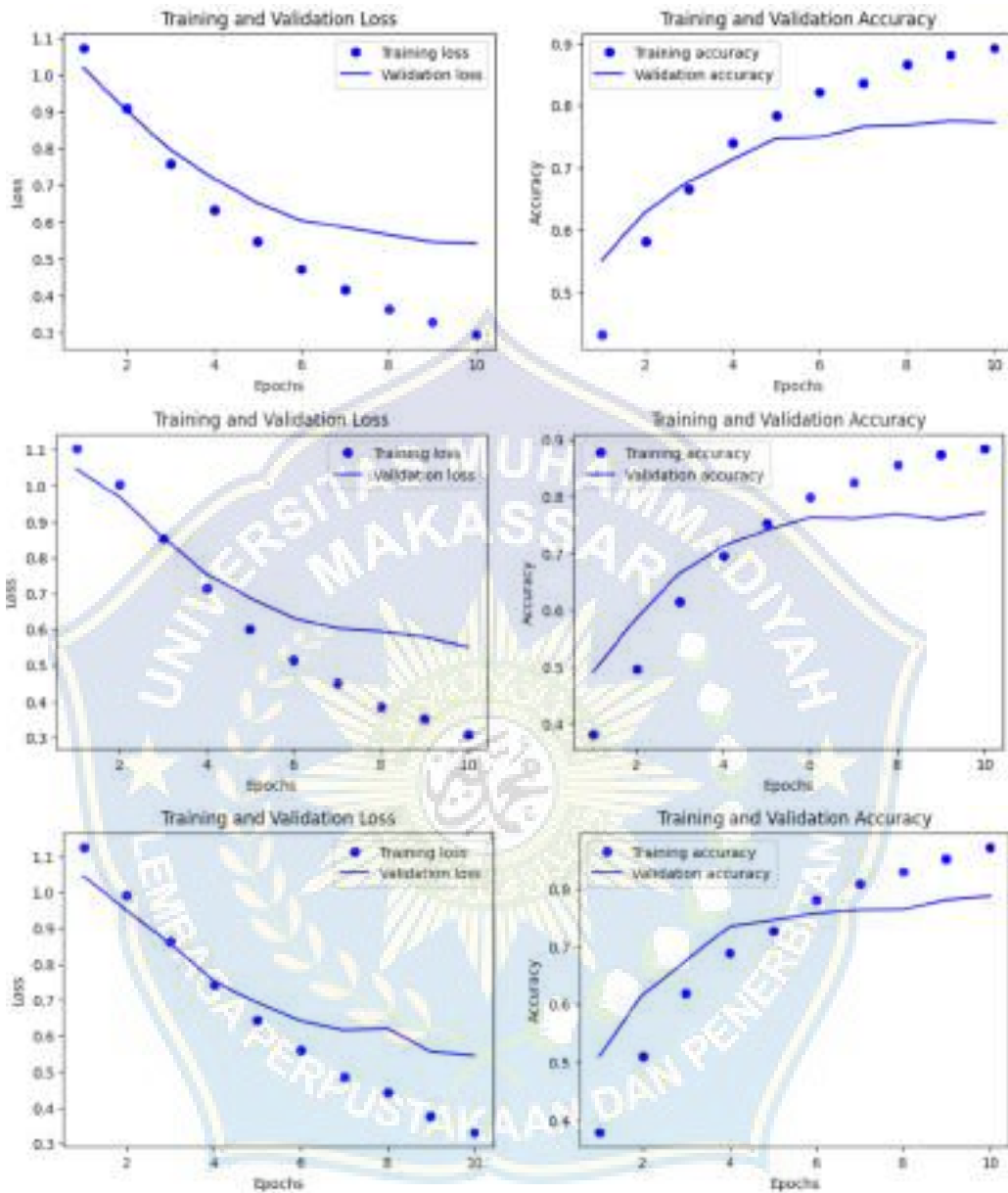
Epoch 1/10
254/254 [#####] - 7s 23ms/step - loss: 0.8177 - accuracy: 0.6894 - val_loss: 0.8368 - val_accuracy: 0.7209
Epoch 2/10
254/254 [#####] - 5s 20ms/step - loss: 0.4622 - accuracy: 0.8165 - val_loss: 0.6279 - val_accuracy: 0.7956
Epoch 3/10
254/254 [#####] - 5s 18ms/step - loss: 0.3165 - accuracy: 0.8805 - val_loss: 0.6814 - val_accuracy: 0.7209
Epoch 4/10
254/254 [#####] - 4s 23ms/step - loss: 0.2138 - accuracy: 0.9208 - val_loss: 0.7882 - val_accuracy: 0.7889
Epoch 5/10
254/254 [#####] - 5s 18ms/step - loss: 0.1601 - accuracy: 0.9472 - val_loss: 0.8829 - val_accuracy: 0.7111
Epoch 6/10
254/254 [#####] - 4s 28ms/step - loss: 0.1180 - accuracy: 0.9687 - val_loss: 1.0978 - val_accuracy: 0.6998
Epoch 7/10
254/254 [#####] - 7s 27ms/step - loss: 0.1074 - accuracy: 0.9657 - val_loss: 1.1275 - val_accuracy: 0.6944
Epoch 8/10
254/254 [#####] - 4s 28ms/step - loss: 0.0993 - accuracy: 0.9672 - val_loss: 1.1168 - val_accuracy: 0.7011
Epoch 9/10
254/254 [#####] - 6s 22ms/step - loss: 0.0868 - accuracy: 0.9725 - val_loss: 1.1492 - val_accuracy: 0.7244
Epoch 10/10
254/254 [#####] - 4s 21ms/step - loss: 0.0768 - accuracy: 0.9796 - val_loss: 1.2887 - val_accuracy: 0.7323

Epoch 1/10
115/115 [#####] - 1s 11ms/step - loss: 0.8578 - accuracy: 0.6917 - val_loss: 0.8488 - val_accuracy: 0.7200
Epoch 2/10
115/115 [#####] - 1s 12ms/step - loss: 0.4848 - accuracy: 0.8176 - val_loss: 0.6485 - val_accuracy: 0.7289
Epoch 3/10
115/115 [#####] - 1s 11ms/step - loss: 0.3828 - accuracy: 0.8925 - val_loss: 0.6992 - val_accuracy: 0.7576
Epoch 4/10
115/115 [#####] - 1s 20ms/step - loss: 0.2133 - accuracy: 0.9298 - val_loss: 0.7888 - val_accuracy: 0.7179
Epoch 5/10
115/115 [#####] - 1s 16ms/step - loss: 0.1554 - accuracy: 0.9439 - val_loss: 0.9092 - val_accuracy: 0.7044
Epoch 6/10
115/115 [#####] - 1s 22ms/step - loss: 0.1242 - accuracy: 0.9183 - val_loss: 1.0583 - val_accuracy: 0.6998
Epoch 7/10
115/115 [#####] - 1s 18ms/step - loss: 0.1018 - accuracy: 0.9658 - val_loss: 1.1119 - val_accuracy: 0.7089
Epoch 8/10
115/115 [#####] - 1s 23ms/step - loss: 0.0852 - accuracy: 0.9736 - val_loss: 1.1659 - val_accuracy: 0.7167
Epoch 9/10
115/115 [#####] - 1s 19ms/step - loss: 0.0735 - accuracy: 0.9764 - val_loss: 1.2005 - val_accuracy: 0.7222
Epoch 10/10
115/115 [#####] - 1s 16ms/step - loss: 0.0639 - accuracy: 0.9776 - val_loss: 1.2869 - val_accuracy: 0.7111

Epoch 1/24
197/197 [#####] - 5s 22ms/step - loss: 0.8861 - accuracy: 0.5917 - val_loss: 0.8614 - val_accuracy: 0.7252
Epoch 2/24
197/197 [#####] - 4s 39ms/step - loss: 0.5770 - accuracy: 0.8148 - val_loss: 0.6512 - val_accuracy: 0.7181
Epoch 3/24
197/197 [#####] - 4s 31ms/step - loss: 0.3847 - accuracy: 0.8978 - val_loss: 0.7887 - val_accuracy: 0.7231
Epoch 4/24
197/197 [#####] - 4s 32ms/step - loss: 0.2998 - accuracy: 0.9349 - val_loss: 0.7945 - val_accuracy: 0.7101
Epoch 5/24
197/197 [#####] - 4s 37ms/step - loss: 0.2628 - accuracy: 0.9087 - val_loss: 0.9882 - val_accuracy: 0.7074
Epoch 6/24
197/197 [#####] - 4s 22ms/step - loss: 0.2076 - accuracy: 0.9665 - val_loss: 1.0580 - val_accuracy: 0.6954
Epoch 7/24
197/197 [#####] - 4s 27ms/step - loss: 0.1887 - accuracy: 0.9717 - val_loss: 1.1382 - val_accuracy: 0.6929
Epoch 8/24
197/197 [#####] - 4s 22ms/step - loss: 0.1755 - accuracy: 0.9746 - val_loss: 1.1608 - val_accuracy: 0.7148
Epoch 9/24
197/197 [#####] - 4s 28ms/step - loss: 0.1685 - accuracy: 0.9768 - val_loss: 1.2029 - val_accuracy: 0.7222
Epoch 10/24
197/197 [#####] - 5s 23ms/step - loss: 0.1613 - accuracy: 0.9787 - val_loss: 1.3278 - val_accuracy: 0.7181

```

Lampiran 7. Grafik accuracy dan loss



Lampiran 8. Hasil prediksi

F1 Score: 0.7604043715952232
 Precision: 0.821946619840725
 Recall: 0.7177777777777777

	precision	recall	f1-score	support
0	0.88	0.75	0.81	138
1	0.83	0.59	0.69	160
2	0.76	0.82	0.79	152
micro avg	0.82	0.72	0.76	450
macro avg	0.82	0.72	0.76	450
weighted avg	0.82	0.72	0.76	450
samples avg	0.72	0.72	0.72	450

F1 Score: 0.7603766942271739
 Precision: 0.800833901673023
 Recall: 0.7255555555555555

	precision	recall	f1-score	support
0	0.83	0.78	0.80	295
1	0.75	0.62	0.68	294
2	0.82	0.78	0.80	311
micro avg	0.80	0.73	0.76	900
macro avg	0.80	0.72	0.76	900
weighted avg	0.80	0.73	0.76	900
samples avg	0.73	0.73	0.73	900

F1 Score: 0.7639640933244367
 Precision: 0.8351513971266455
 Recall: 0.7222222222222222

	precision	recall	f1-score	support
0	0.82	0.86	0.84	450
1	0.87	0.51	0.64	428
2	0.82	0.79	0.80	472
micro avg	0.83	0.72	0.77	1350
macro avg	0.84	0.72	0.76	1350
weighted avg	0.84	0.72	0.76	1350
samples avg	0.72	0.72	0.72	1350

Lampiran 9. Hasil klasifikasi

F1 Score: 0.7712182955373931
 Precision: 0.7787719286745893
 Recall: 0.7733333333333333

	precision	recall	f1-score	support
0	0.84	0.81	0.82	138
1	0.79	0.65	0.71	160
2	0.72	0.87	0.79	152
accuracy			0.77	450
macro avg	0.78	0.78	0.77	450
weighted avg	0.78	0.77	0.77	450

F1 Score: 0.7705654957440907
 Precision: 0.7702958526513878
 Recall: 0.7711111111111111

	precision	recall	f1-score	support
0	0.81	0.81	0.81	295
1	0.71	0.69	0.70	294
2	0.79	0.81	0.80	311
accuracy			0.77	900
macro avg	0.77	0.77	0.77	900
weighted avg	0.77	0.77	0.77	900

F1 Score: 0.7835929377898251
 Precision: 0.7905746005758145
 Recall: 0.7881481481481482

	precision	recall	f1-score	support
0	0.78	0.89	0.83	450
1	0.81	0.62	0.70	428
2	0.78	0.85	0.81	472
accuracy			0.79	1350
macro avg	0.79	0.78	0.78	1350
weighted avg	0.79	0.79	0.78	1350

Lampiran 10. Dataset uji hasil prediksi

id	Uraian	Label sebenarnya	Prediksi
1	Kurang memuaskan nilai ujian bahasa Indonesia yang tidak terlalu banyak belajar	Ya (Y)	Ya (Y)
2	Tidak yakin apakah GDP sudah bisa naik lebih	Ya (Y)	Ya (Y)
3	Tidak yakin dan tidak ngerti	Ya (Y)	Ya (Y)
4	Sayang sekali sudah bisa diurusah pengatahian tidak begitu memuaskan	Ya (Y)	Ya (Y)
5	Cocok untuk analisis yaitu ya karena di dan pih an membuat tempat ini supaya ada area main anak anak yg murah dan ramah yg murah	Ya (Y)	Ya (Y)
6	Tempat parkirnya luas tetapi suasana malamnya kurang bagus	Ya (Y)	Ya (Y)
7	Balkon nyaman an yang luas dan ada perabotnya dan tempatnya itu pengumpul an dan orang tua yang mendampingi anak-anaknya di tempat itu kurang	Ya (Y)	Ya (Y)
8	Tempat luas ada air wajan tidak under ya di dalam kuali dan sayur atasannya cukup enak	Ya (Y)	Ya (Y)
9	Tapi suasana malamnya tidak yang berhibur untuk diputar an	Ya (Y)	Ya (Y)
10	di sini suasana dan area luar yang sangat tempatnya enak kurang bersih	Ya (Y)	Ya (Y)
11	Suamiku sangat nyaman dan menyenangkan cocok untuk liburan	Ya (Y)	Ya (Y)
12	Tempat lagi bagus untuk orang	Ya (Y)	Ya (Y)
13	Tidak mahal tapi suasana tidak	Ya (Y)	Ya (Y)
14	Tempat untuk ngopi dan santai	Ya (Y)	Ya (Y)
15	Lumayan bagus selain lama tidak masalah ini tidak ada yg istimewa yang membuat saya dikembalikan pilihan saya per pertimbangan... murah dan	Ya (Y)	Ya (Y)
16	Vila ini lagi dikembang untuk orang	Ya (Y)	Ya (Y)
17	Bukan tempat yang bagus sih... tapi ini bersama keluarga yang melakukan wisata ke sini... suasana dan fasilitas dengan fasilitas yang dibudaya	Ya (Y)	Ya (Y)
18	Cocok untuk pengantar keluarga di	Ya (Y)	Ya (Y)
19	tidak terlalu mahal	Ya (Y)	Ya (Y)
20	Tempat baru untuk keluarga di kota ini	Ya (Y)	Ya (Y)
21	Fasilitas yang lengkap tidak mahal dengan baik banget area yang tidak kecil	Ya (Y)	Ya (Y)
22	Tempat yang perlu revisi	Ya (Y)	Ya (Y)



MAJELIS PENDIDIKAN TINGGI PIMPINAN PUSAT MUHAMMADIYAH
UNIVERSITAS MUHAMMADIYAH MAKASSAR
UPT PERPUSTAKAAN DAN PENERBITAN

Alamat Kantor: Jl. Sultan Azzuddin No.259 Makassar 90221 Telp:(0411) 816072,881593, Fax:(0411) 800588

بِسْمِ اللَّهِ الرَّحْمَنِ الرَّحِيمِ

SURAT KETERANGAN BEBAS PLAGIAT

UPT Perpustakaan dan Penerbitan Universitas Muhammadiyah Makassar,
Menerangkan bahwa mahasiswa yang tersebut namanya di bawah ini:

Nama : Arvianda

Nim : 105841102520

Program Studi : Teknik Informatika

Dengan nilai:

No	Bab	Nilai	Ambang Batas
1	Bab 1	6 %	10 %
2	Bab 2	22 %	25 %
3	Bab 3	8 %	10 %
4	Bab 4	3 %	10 %
5	Bab 5	4 %	5 %

Dinyatakan telah lulus cek plagiat yang diadakan oleh UPT- Perpustakaan dan Penerbitan
Universitas Muhammadiyah Makassar Menggunakan Aplikasi Turnitin.

Demikian surat keterangan ini diberikan kepada yang bersangkutan untuk dipergunakan
seperhnya.

Makassar, 14 Agustus 2024

Mengetahui,

Kepala UPT- Perpustakaan dan Penerbitan,



Jl. Sultan Azzuddin no 259 makassar 90222
Telepon (0411)888872,881 593, fax (0411)865 588
Website: www.library.umh.ac.id
E-mail: ucp@pustakam2010@umh.ac.id

ARVIANDA 105841102520 Bab I

by Tahap Tutup



Submission date: 14-Aug-2024 10:46AM (UTC+0700)

Submission ID: 2431809750

File name: BAB_I_10.docx (26K)

Word count: 1008

Character count: 6541

ARVIANDA 105841102520 Bab I

ORIGINALITY REPORT

6%	6%	5%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	core.ac.uk Internet Source	4%
2	repositori.usu.ac.id Internet Source	2%



ARVIANDA 105841102520 Bab

II

by Tahap Tutup

Submission date: 12-Aug-2024 02:16PM (UTC+0700)

Submission ID: 2430904859

File name: BAB_II_1.docx (219.44K)

Word count: 2580

Character count: 17461

ARVIANDA 105841102520 Bab II

ORIGINALITY REPORT

22% SIMILARITY INDEX	21% INTERNET SOURCES	7% PUBLICATIONS	2% STUDENT PAPERS
--------------------------------	--------------------------------	---------------------------	-----------------------------

PRIMARY SOURCES

1	ejournal.poltektegal.ac.id Internet Source	5%
2	eprints.unisla.ac.id Internet Source	3%
3	www.researchgate.net Internet Source	3%
4	ejournal-poltekiparmks.ac.id Internet Source	2%
5	media.neliti.com Internet Source	2%
6	jurnal.ugm.ac.id Internet Source	2%
7	core.ac.uk Internet Source	2%
8	e-journal.stmiklombok.ac.id Internet Source	2%
9	Nurhaliza Khesya. "MENGENAL FLOWCHART DAN PSEUDOCODE DALAM ALGORITMA DAN	2%

PEMROGRAMAN", Open Science Framework,
2021

Publication

Exclude quotes

Exclude matches < 2%

Exclude bibliography



ARVIANDA 105841102520 Bab

III

by Tahap Tutup



Submission date: 14-Aug-2024 10:46AM (UTC+0700)

Submission ID: 2431809974

File name: BAB_III_10.docx (155.12K)

Word count: 1110

Character count: 7348

ARVIANDA 105841102520 Bab III

ORIGINALITY REPORT

8%

SIMILARITY INDEX

4%

INTERNET SOURCES

5%

PUBLICATIONS

3%

STUDENT PAPERS

PRIMARY SOURCES



1	repository.unimal.ac.id Internet Source	2%
2	Raden Dewi Setiani. "Implementasi Kebijakan Pembentukan Kabupaten/Kota Layak Anak Pada Bidang Pendidikan dan Kesehatan di Kabupaten Pandeglang", Open Science Framework, 2018 Publication	2%
3	Submitted to Universitas Negeri Makassar Student Paper	2%
4	Submitted to University of Wollangong Student Paper	2%
5	digilib.unila.ac.id Internet Source	2%

Exclude quotes

On

Exclude matches

< 2%

Exclude bibliography

On

ARVIANDA 105841102520 Bab
IV
by Tahap Tutup

Submission date: 12-Aug-2024 02:18PM (UTC+0700)
Submission ID: 2430905504
File name: BAB_IV_1.docx (1.73M)
Word count: 8289
Character count: 62841

ARVIANDA 105841102520 Bab IV

ORIGINALITY REPORT

3% SIMILARITY INDEX	4% INTERNET SOURCES	2% PUBLICATIONS	2% STUDENT PAPERS
-------------------------------	-------------------------------	---------------------------	-----------------------------

PRIMARY SOURCES

1	digilibadmin.unismuh.ac.id Internet Source		3%
----------	---	--	-----------

Exclude quotes Exclude matches < 2%
Exclude bibliography



ARVIANDA 105841102520 Bab

V

by Tahap Tutup



Submission date: 14-Aug-2024 10:47AM (UTC+0700)

Submission ID: 2431810188

File name: BAB_V_7.docx (21.46K)

Word count: 286

Character count: 1931

ARVIANDA 105841102520 Bab V

ORIGINALITY REPORT

4%

SIMILARITY INDEX

4%

INTERNET SOURCES

3%

PUBLICATIONS

0%

STUDENT PAPERS

PRIMARY SOURCES

1

repository.umy.ac.id
Internet Source



4%

Exclude quotes On
Exclude bibliography On

Exclude matches < 2%

