KLASIFIKASI UJARAN KEBENCIAN *MULTI LABEL* MENGGUNAKAN ARSITEKTUR *LONG SHORT TERM MEMORY* DAN *TRANSFORMER* DENGAN *BACK TRANSLATION* DAN BERT

SKRIPSI

Sebagai Salah Satu Syarat untuk Memperoleh Gelar

Sarjana Matematika

Oleh:

PUTRI PRATIWI

NIM. 08011282126028

JURUSAN MATEMATIKA

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS SRIWIJAYA

2025

LEMBAR PENGESAHAN

KLASIFIKASI UJARAN KEBENCIAN *MULTI LABEL* MENGGUNAKAN ARSITEKTUR *LONG SHORT TERM MEMORY* DAN *TRANSFORMER* DENGAN AUGMENTASI *BACK TRANSLATION* DAN BERT

SKRIPSI

Sebagai Salah Satu Syarat untuk Memperoleh Gelar

Sarjana Matematika

Oleh

PUTRI PRATIWI

NIM. 08011282126028

Indralaya, 20 Maret 2025

Pembimbing Kedua	**Pembimbing Utama**
Dr. Bambang Suprihatin, S.Si., M.Si	**Dr. Anita Desiani, S.Si., M.Kom.**
NIP. 197101261994121001	**NIP.197712112003122002**

Mengetahui,
Ketua Jurusan Matematika

Dr. Dian Cahyawati Sukanda, S.Si., M.Si.
NIP. 197303212000122001

PERNYATAAN KEASLIAN KARYA ILMIAH

Yang bertanda tangan di bawah ini:

Nama Mahasiswa	:	Putri Pratiwi
NIM	:	08011282126028
Jurusan	:	Matematika
Fakultas	:	Matematika dan Ilmu Pengetahuan Alam

Menyatakan bahwa skripsi ini adalah hasil karya ilmiah saya sendiri dan karya ilmiah ini belum pernah diajukan sebagai pemenuhan persaratan untuk memperoleh gelar kesarjanaan strata satu (S1) dari Universitas Sriwijaya maupun perguruan tinggi lain. Semua informasi yang dimuat didalam skripsi ini yang berasal dari penulis lain baik yang dipublikasi atau tidak telah diberikan penghargaan dengan mengutip nama sumber penulis baik yang secara benar. Semua isi dari skripsi ini sepenuhnya menjadi tanggung jawab saya sebagai penulis.

Demikianlah surat pernyataan ini saya buat dengan sebenarnya.

Indralaya, 20 Maret 2025

Penulis

Putri Pratiwi

NIM. 08011282126028

HALAMAN PERSEMBAHAN

Kupersembahkan skripsi ini untuk:

Yang Maha Kuasa Allah Subhanahu Wa Ta'ala

Kedua orang tuaku tercinta,

Adik-adikku tersayang,

Keluarga besarlu,

Semua guru dan dosenku,

Sahabat-sahabatku,

Almamaterku.

Diriku sendiri

Motto

"Tidak mudah bukan berarti tidak mungkin, kamu bisa lebih dari apa yang kamu kira"

-Putri Pratiwi

KATA PENGANTAR

Puji syukur atas kehadirat Allah Subhanahu Wa Ta'ala yang telah memberikan rahmat dan hidayah-Nya, sehingga penulis dapat menyelesaikan skripsi berjudul "Klasifikasi Ujaran Kebencian *Multi Label* Menggunakan Arsitektur *Long Short Term Memory* dan *Transformer* dengan Augmentasi *Back Translation* dan BERT". Skripsi ini ditulis sebagai salah satu syarat memperoleh gelar sarjana sains studi Matematika di Fakultas MIPA Universitas Sriwijaya.

Penulis menyadari bahwa skripsi ini tidak mungkin terselesaikan tanpa adanya semangat, dukungan, bantuan, bimbingan dan nasihat yang diberikan berbagai pihak selama proses penyusunan ini berlangsung. Penulis mengucapkan terima kasih yang sebesar-besarnya kepada orang tua tercinta, Ayahku **Supriadi dan** Ibuku **Sri Wahyuningsih** yang tidak pernah berhenti berjuang dan memberikan yang terbaik untukku sebagai putrinya. Terima kasih karena tak pernah lelah mendidik, menasehati, membimbing, mendukung dan terus mendo'akan. Penulis juga ingin menyampaikan ucapan terima kasih kepada sebesar-besarnya dan penghargaan setinggi-tingginya kepada:

1. Bapak **Prof. Hermansyah, S.Si., M.Si., Ph.D** selaku Dekan Fakultas MIPA Universitas Sriwijaya.
2. Ibu **Dr. Dian Cahyawati Sukanda, S.Si., M.Si**. selaku Ketua Jurusan Matematika dan Ibu **Des Alwine Zayanti, S.Si., M.Si.** selaku Sekretaris Jurusan Matematika yang telah membimbing dan mengarahkan dalam urusan akademik selama menempuh perkuliahan di Jurusan Matematika FMIPA Universitas Sriwijaya.

3. Ibu **Dr. Anita Desiani, S.Si., M.Kom.** selaku Dosen Pembimbing Pertama dan Bapak **Dr. Bambang Suprihatin, S.Si., M.Si.** selaku Dosen Pembimbing Kedua yang telah bersedia meluangkan waktu, tenaga, dan pikiran untuk memberikan bimbingan, arahan dan didikan yang berharga selama pembuatan skripsi, perlombaan dan proses perkuliahan.
4. Bapak **Drs. Endro Setyo Cahyono, M.Si** dan Ibu **Irmeilyana, S.Si., M.Si** selaku dosen pembahas, telah memberikan respons, kritik, dan saran yang sangat berguna untuk perbaikan dan penyelesaian skripsi ini.
5. **Seluruh Dosen di Jurusan Matematika FMIPA** yang telah memberikan ilmu, nasihat, motivasi, serta bimbingan selama proses perkuliahan. Bapak **Irwansyah** dan Ibu **Hamidah** selaku staf administrasi di Jurusan Matematika FMIPA Universitas Sriwijaya yang telah membantu penulis selama perkuliahan.
6. **Himpunan Mahasiswa Matematika (Himastik)** yang telah menjadi wadah untuk mengembangkan minat, bakat, dan kreativitas di bidang matematika.
7. Semua sahabat seperjuangan **Komputasi 2021** selama masa perkuliahan dan proses skripsi. Kakak-kakak tingkat bidang komputasi yang telah membantu dan membagikan ilmunya kepada penulis, serta adik-adik tingkat yang telah membantu proses skripsi.
8. Saudaraku, keluarga besarku, serta sahabat-sahabatku yang senantiasa memberikan semangat dan doa terbaik untuk penulis.
9. Semua pihak yang tidak dapat penulis sebutkan satu persatu yang telah memberikan bantuan dalam menyelesaikan skripsi ini hanya ucapan terima

kasih yang dapat penulis berikan.

Semoga skripsi ini dapat menambah pengetahuan dan bermanfaat bagi mahasiswa/i Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Sriwijaya dan semua pihak yang membutuhkan.

Indralaya, Maret 2025

Penulis

MULTI-LABEL HATE SPEECH CLASSIFICATION USING LONG SHORT TERM MEMORY AND TRANSFORMER ARCHITECTURE WITH BACK TRANSLATION AUGMENTATION AND BERT

***By*:**

PUTRI PRATIWI
NIM.08011282126028

ABSTRACT

Multi-label hate speech can be categorized by more than one label at once, such as individual, religious, racial hate speech, with different levels of severity. In the multi-label context, class and label are used to determine the categories in the data. Labels refer to the types of hate speech found in the data, while class refers to the combination of the various labels. One of the labeling methods that can be used to determine the labels in the data is label powerset. Multi-label hate speech spreads quickly and widely, so automatic early detection is needed. This research uses a combination of classification architecture and augmentation techniques. The classification architecture used is Long Short Term Memory (LSTM) to process the sequence of important information and Transformer to understand the context globally. The combination of architectures requires a large amount of data. The method used to multiply the data is to perform augmentation using the back translation method and Bidirectional Encoder Representation from Transformer (BERT). The results show an increase in the amount of data up to three times. The combination of architectures has good model performance. The accuracy result of 86% shows that the model is able to predict the class correctly as a whole. The precision result of 86% shows that the model can identify positive classes with a low error rate. The recall result of 85% shows that the model can detect most of the positive classes from the available data. The f1-score result of 85% shows that the model is consistent in classifying positive classes. In each class, the model performed very well in detecting the H0 and H26 classes with accuracy, precision, recall, and f1-score results of more than 90% each. In the H17, H23, and H31 classes, the model still struggled with classification. This research shows that the combination of LSTM and Transformer architecture as well as the combination of back translation augmentation and BERT can be used for multi-label hate speech classification. For further research, the balance of data between classes needs to be considered so that the model's performance is more optimal.

Keywords: hate speech, back translation, Bidirectional Encoder Representation from Transformer (BERT), Long Short Term Memory (LSTM), Transformer

KLASIFIKASI UJARAN KEBENCIAN *MULTI LABEL* MENGGUNAKAN ARSITEKTUR *LONG SHORT TERM MEMORY* DAN *TRANSFORMER* DENGAN AUGMENTASI *BACK TRANSLATION* DAN BERT

Oleh:

PUTRI PRATIWI
NIM.08011282126028

ABSTRAK

Ujaran kebencian *multi label* dapat dikategorikan lebih dari satu *label* sekaligus, seperti ujaran kebencian individu, agama, ras, dengan tingkat keparahan yang berbeda. Pada konteks *multi label,* kelas dan *label* digunakan untuk menentukan kategori pada data. *Label* merujuk pada jenis ujaran kebencian yang ditemukan dalam data, sementara kelas merujuk pada hasil kombinasi dari berbagai label tersebut. Salah satu metode pelabelan yang dapat digunakan untuk menentukan *label* pada data adalah *label powerset*. Ujaran kebencian *multi label* menyebar dengan cepat dan luas, sehingga diperlukan deteksi dini secara otomatis. Penelitian ini menggunakan kombinasi arsitektur klasifikasi dan teknik augmentasi. Arsitektur klasifikasi yang digunakan adalah *Long Short Term Memory* (LSTM) untuk memproses urutan informasi penting dan *Transformer* untuk memahami konteks secara global. Kombinasi arsitektur memerlukan jumlah data yang banyak. Cara yang digunakan untuk memperbanyak data adalah melakukan augmentasi menggunakan metode *back translation* dan *Bidirectional Encoder Representation from Transformer* (BERT). Hasil penelitian menunjukkan peningkatan jumlah data hingga tiga kali lipat. Kombinasi arsitektur memiliki kinerja model yang baik. Hasil akurasi sebesar 86% menunjukkan bahwa model mampu memprediksi kelas dengan benar secara keseluruhan. Hasil presisi sebesar 86% menunjukkan bahwa model dapat mengidentifikasi kelas positif dengan tingkat kesalahan yang rendah. Hasil *recall* sebesar 85% menunjukkan bahwa model dapat mendeteksi sebagian besar kelas positif dari data yang tersedia. Hasil *f1-score* sebesar 85% menunjukkan bahwa model konsisten dalam mengklasifikasikan kelas positif. Pada masing-masing kelas, model menunjukkan performa sangat baik dalam mendeteksi kelas H0 dan H26 dengan hasil akurasi, presisi, *recall,* dan *f1-score* masing lebih dari 90%. Pada kelas H17, H23, dan H31 model masih kesulitan dalam melakukan klasifikasi. Penelitian ini menunjukkan bahwa kombinasi arsitektur LSTM dan *Transformer* serta kombinasi augmentasi *back translation* dan BERT dapat digunakan untuk klasifikasi ujaran kebencian *multi label*. Untuk penelitian selanjutnya, keseimbangan data antar kelas perlu diperhatikan agar konerja model lebih optimal.

Kata kunci: ujaran kebencian, *back translation, Bidirectional Encoder Representation from Transformer* (BERT), *Long Short Term Memory* (LSTM), *Transformer*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR LAMPIRAN

BAB I

PENDAHULUAN

1.1 Latar Belakang

Cuitan di media sosial seperti Twitter dapat berdampak negatif, salah satunya dalam bentuk ujaran kebencian. Ujaran kebencian memiliki berbagai tingkat keparahan, mulai dari rendah, sedang, hingga tinggi (Hana *et al*., 2020). Pada satu ujaran kebencian dapat masuk ke dalam lebih dari satu kategori, seperti ujaran kebencian individu, agama, atau ras, dengan tingkat keparahan yang berbeda sehingga perlu dilakukan pelabelan agar dapat memberi penangan sesuai kategorinya. Ujaran kebencian memiliki dampak terhadap negara berupa ketegangan sosial antar kelompok masyarakat (Elsafoury, 2023). Untuk mengatasi masalah tersebut, dilakukan klasifikasi otomatis menggunakan algoritma yang ada pada *deep learning*.

Transformer merupakan algoritma *deep learning* yang banyak digunakan untuk melakukan klasifikasi (Mozafari *et al.*, 2020). Salah satu keunggulan *Transformer* adalah mampu menangkap informasi global melalui mekanisme *self attention*, karena memproses *input* secara pararael (Kovaleva *et al*., 2019). Penerapan *Transformer* pada klasifikasi telah banyak dilakukan. Bilal *et al*. (2023) menggunakan arsitektur *Transformer* untuk klasifikasi ujaran kebencian bahasa Roman Urdu dengan akurasi, *f-measure*, presisi, dan *recall* sebesar 83%. Sarkar *et al*. (2021) menggunakan arsitektur *Transformer* untuk klasifikasi ujaran kebencian bahasa Inggris dengan *f1-score* sebesar 87%. *Transformer* adalah arsitektur yang

memiliki parameter besar sehingga membutuhkan jumlah data yang banyak. Jumlah data cuitan *tweet* masih terbatas terutama pada bahasa selain bahasa Inggris (Ibrohim *and* Budi., 2023). Jumlah data yang terbatas dengan parameter yang besar dapat menyebabkan pembelajaran *Transformer* menjadi *overfitting* (Tra *et al*., 2019).

Untuk mengatasi keterbatasan data terutama pada cuitan *tweet* adalah dengan melakukan teknik augmentasi (Shorten *et al*., 2021). Augmentasi merupakan teknik untuk memperbanyak data. Pada suatu teks, augmentasi dilakukan dengan cara menghapus, menyisipkan atau mengganti kata sehingga diperoleh kalimat baru yang lebih bervariasi (Sabty *et al*., 2021). Salah satu metode augmentasi pada teks adalah *Bidirectional Encoder Representations from Transformers* (BERT).

Augmentasi BERT adalah teknik untuk menambah jumlah data teks dengan menggunakan *self atttention* (Wu *et al*., 2019). *Self attention* pada BERT digunakan untuk menangkap hubungan antar kata dalam satu kalimat, sehingga dapat memperhatikan hubungan dua arah yaitu arah sebelum dan arah setelah kata yang di-*mask*. Augmentasi BERT tidak langsung menyisipkan kata baru, tetapi melakukan pembobotan menggunakan fungsi *softmax*. Kata dengan bobot tertinggi digunakan untuk mengganti kata yang di-*mask*. (Kolesnichenko *et al*., 2023). Kelebihan dari augmentasi BERT yaitu menghasilkan teks bervariasi dengan makna sesuai kalimat aslinya, karena penggantian kata dihasilkan dengan melakukan pembobotan nilai.

Beberapa penelitian telah melakukan augmentasi data menggunakan BERT. Kapil *and* Ekbal (2022) menerapkan augmentasi BERT pada *dataset* bahasa India menggunakan arsitektur *Transformer* dengan nilai *f1-score* sebesar 74%. Takawane *et al.* (2023) menerapkan augmentasi BERT pada *dataset* bahasa Inggris menggunakan arsitektur *Transformer* dengan nilai *f1-score* sebesar 72%. BERT banyak diterapkan dalam bahasa Inggris karena korpus pelatihannya berasal dari teks berbahasa Inggris sehingga, penerapan pada bahasa lain menjadi lebih rumit (Devlin *et al.*, 2019).

Penerapan augmentasi BERT pada bahasa selain bahasa Inggris, diperlukan proses penerjemahan menggunakan teknik *back translation*. *Back translation* merupakan teknik sederhana untuk menerjemahkan teks dari bahasa asal ke bahasa Inggris, kemudian menerjemahkannya kembali ke bahasa asal (Beddiar *et al.*, 2021). Desiani *et al.* (2023) melakukan augmentasi *back translation* dan EDA untuk klasifikasi ujaran kebencian pada dataset bahasa Indonesia menggunakan arsitektur klasifikasi *Transformer*. Hasil penelitian menunjukkan bahwa nilai akurasi, presisi, *recall*, dan *f1-score* sebesar 85% dan hanya menggunakan dua *label* yaitu *hate speech* dan *non hate speech*. Beddiar *et al.* (2021) melakukan augmentasi *back translation* dan *paraphrasing* untuk klasifikasi ujaran kebencian pada dataset bahasa Jerman dengan menggunakan arsitektur klasifikasi LSTM. Hasil penelitian menunjukkan bahwa nilai akurasi dan *f1-score* sebesar 99%, namun hanya melakukan klasifikasi pada dua kelas yaitu *hate* dan *non hate*. Penerapan metode augmentasi dapat memenuhi kebutuhan data *training* yang besar oleh arsitektur *Transformer* (Ansari *et al.*, 2021). Selain membutuhkan data yang besar,

Transformer tidak dapat menyeleksi informasi penting dan beresiko kehilangan penekanan informasi yang benar-benar penting (Vaswani *et al*., 2017).

Arsitektur yang dapat menyeleksi informasi penting adalah arsitektur *Long Short-Term Memory* (LSTM). Arsitektur LSTM mampu memilih informasi penting melalui mekanisme *gate* (Fazil *et al*., 2023). LSTM memiliki tiga *gate* utama yaitu *input gate, output gate*, dan *forgot gate*. *Input gate* berfungsi untuk menambahkan informasi baru yang akan disimpan pada tempat penyimpanan informasi. *Output gate* berfungsi untuk menghasilkan informasi dari pembelajaran model yang digunakan sebagai *output*. *Forgot gate* berfungsi untuk menyeleksi informasi penting (Ayo *et al*., 2020). Mekanisme *gate* memungkinkan LSTM untuk memahami konteks kalimat dalam menyeleksi informasi penting, sehingga menghasilkan pemahaman yang lebih baik dan pemrosesan data (Marpaung *et al*., 2021).

Das *et al*. (2021) menggunakan LSTM untuk klasifikasi ujaran kebenncian bahasa Bangla. Hasil penelitian menunjukkan nilai akurasi, presisi, *recall*, dan *f1-score* sebesar 75% menggunakan tujuh kelas yaitu *aggressive comment, poilitical comment, hate speech, ethnical attack, religious hatred, religious comment,* dan *suicidal comment*. Verma *et al*. (2023) menggunakan LSTM untuk klasifikasi ujaran kebencian bahasa Inggris. Hasil penelitian menunjukkan nilai akurasi, presisi, *recall*, dan *f1-score* sebesar 95% namun, hanya menggunakan 2 kelas yaitu *hate* dan *non-hate*.

Penelitian ini menggabungkan teknik augmentasi dan teknik klasifikasi ujaran kebencian pada cuitan *tweet* bahasa Indonesia. Teknik augmentasi yang diajukan

pada penelitian ini adalah *back translation* dan augmentasi BERT untuk menambah jumlah data. Proses augmentasi dilakukan dengan menerjemahkan teks awal ke bahasa Inggris, lalu diproses dengan BERT, dan diterjemahkan kembali ke bahasa Indonesia. Augmentasi BERT digunakan untuk menambah keragaman data tanpa mengubah makna, dan *back translation* digunakan untuk mengatasi korpus yang terbatas.

Klasifikasi teks ujaran kebencian *multi label* dalam bahasa Indonesia menggunakan kombinasi LSTM dan *Transformer*. LSTM digunakan pada bagian awal model untuk menyaring informasi yang kurang penting, sedangkan *Transformer* digunakan untuk memahami hubungan antar kata dalam kalimat secara global. Pendekatan ini bertujuan untuk meningkatkan efisiensi model dalam mengenali berbagai jenis dan tingkatan ujaran kebencian. Hasil pengujian dari metode yang diusulkan akan diukur menggunakan akurasi, presisi, *recall*, dan *f1-score*. Pengukuran keberhasilan model dilakukan untuk melihat performa model dalam melakukan klasifikasi teks ujaran kebencian.

1.2 Perumusan Masalah

Rumusan masalah pada penelitian ini, sebagai berikut:

1. Bagaimana penerapan teknik *back translation* dan augmentasi BERT dalam menambah jumlah data ujaran kebencian pada *tweet* bahasa Indonesia?
2. Bagaimana hasil evaluasi kinerja arsitektur LSTM dan *Transformer* pada klasifikasi teks *dataset* ujaran kebencian *tweet* bahasa Indonesia yang telah

di augmentasi menggunakan ukuran kinerja akurasi, presisi, *recall*, dan *f1-score*?

1.3 Pembatasan Masalah

Pembatasan masalah pada penelitian ini, sebagai berikut:

1. Data yang digunakan pada penelitian ini terdiri dari 44 kelas, yang merupakan kombinasi dari 12 *label* yaitu ujaran kebencian terhadap individu, kelompok, agama, ras, fisik, gender, serta tingkatan keparahannya.
2. Ukuran evaluasi kinerja pada augmentasi dan klasifikasi teks dalam mendeteksi ujaran kebencian pada *tweet* menggunakan akurasi, presisi, *recall*, dan *f1- score.*

1.4 Tujuan

Tujuan penelitian ini, sebagai berikut:

1. Menerapkan teknik *back translation* dan augmentasi BERT dalam menambah jumlah data ujaran kebencian pada *tweet* bahasa Indonesia .
2. Mengetahui hasil evaluasi kinerja arsitektur LSTM dan *Transformer* pada klasifikasi teks *dataset* ujaran kebencian *tweet* bahasa Indonesia yang telah di augmentasi menggunakan ukuran kinerja akurasi, presisi, *recall*, dan *f1-score*.

1.5 Manfaat

Manfaat penelitian ini yaitu dapat diterapkan dalam sistem otomatis untuk mendeteksi ujaran kebencian sesuai dengan jenis dan tingkatannya di media sosial, sehingga dapat mengenali dan mengatasi konten ujaran kebencian dengan efektif.

BAB II

TINJAUAN PUSTAKA

2.1 Ujaran Kebencian

Ujaran kebencian merupakan perilaku dikrimintaif yang dilakukan kepada individu maupun kelompok (Beyhan *et al*., 2022). Bentuk dari ujaran kebencian dapat berupa tulisan yang menyatakan kencian terhadap atribut tertentu seperti jenis kelamin, etnis, ras, agama, dan karakteristik lainnya (Alkomah *and* Ma, 2022).

Penelitian mengenai klasifikasi teks ujaran kebencian telah dilakukan oleh beberapa penelitian sebelumnya. Ibrohim *and* Budi (2019) melakukan kalsifikasi teks ujaran kebencian menggunakan arsitektur *Random Forest Decission Tree* (RFDT) dengan hasil akurasi sebesar 77.36%. Kurniawan *and* Budi (2020) melakukan kalsifikasi teks ujaran kebencian menggunakan arsitektur SVM dengan hasil akurasi sebesar 84.77%. Aurora *et al.* (2023) melakukan kalsifikasi teks ujaran kebencian menggunakan arsitektur BiLSTM dengan hasil presisi, *recall*, dan *f1-score*nya sebesar 80.25%. Penelitian tersebut menggunakan *dataset* ujaran kebencian *multi label* bahasa Indonesia

2.2. *Back Translation*

Back translation merupakan metode augmentasi sederhana untuk menerjemahkan kalimat dari satu bahasa ke bahasa lain. Kalimat yang telah diterjemahkan kemudian diterjemahkan kembali ke bahasa asal (Or *et al*., 2018). Contoh penerapan *back translation* dapat dilihat pada Gambar 2.1.

Gambar 2.1. Contoh penerapan *back translation*

Pada Gambar 2.1, dapat dilihat bahwa proses *back translation* dimulai dengan *input* kalimat yang akan diterjemahkan, misalkan *input* berupa kalimat dengan bahasa Indonesia. Kalimat tersebut kemudian diterjemahkan ke bahasa Inggris. Setelah diterjemahkan ke bahasa Inggris, kalimat diterjemahkan kembali ke bahasa asal yaitu bahasa Indonesia. Kalimat bahasa Indonesia yang diterjemahkan digunakan sebagai data baru.

2.3 *Text Preprocessing*

Text preprocessing merupakan proses untuk mempersiapkan data agar menjadi lebih mudah dipahami ketika digunakan dalam pemrosesan model (Chai, 2023). *Text preprocessing* terdiri dari *case folding*, *slang word, stopword removal*, dan *stemming*.

1. *Case Folding*

   Case folding merupakan tahapan mengubah huruf kapital yang ada pada data menjadi huruf kecil (Arbaatun *et al.*, 2022). Tujuannya agar kata-kata yang memiliki kesamaan makna namun memiliki perbedaan dalam penulisan kapitalisasi dapat dimaknai dan diperlakukan sama saat melakukan klasifikasi teks. Namun, pada beberapa kasus penggunaan kapitalisasi memberikan makna berbeda, sehingga tidak semua *text preprocessing* melakukan *case folding*.

2. *Remove Punctuation*

Remove punctuation merupakan proses menghapus tanda baca yang ada pada data seperti titik, tanda tanya, koma, tanda seru, dan lain sebagainya. *Remove punctuation* bertujuan agar model fokus membaca kata-kata yang memiliki makna bukan pada simbol tanda baca, dan menghindari pemecahan kata menjadi *token* yang tidak berguna pada tahap tokenisasi agar klasifikasi teks dapat memberikan hasil yang akurat (Arbaatun *et al.*, 2022).

3. *Slang Word*

Slang word merupakan proses mengubah kata yang tidak baku menjadi kata baku (Dogra *et al.*, 2022). Kamus bahasa yang digunakan pada tahap *slang word* adalah kamus khusus yang dibuat menyesuaikan percakapan sehari-hari dengan kata atau frasa formal yang memiliki makna sama. Proses pada tahap ini dilakukan dengan cara memeriksa apakah suatu kata terdapat pada kamus bahasa khusus *slang word* atau tidak. Tujuan dari tahap *slang word* adalah untuk meningkatkan akurasi dan efektivitas model dalam melakukan klasifikasi teks.

4. *Stopword Removal*

Stopword removal merupakan tahap untuk mengidentifikasi dan menghapus kata yang sering muncul tetapi tidak memiliki nilai informasi penting dalam proses klasifikasi teks (Hana *et al.*, 2020). Dengan menghapus *stop word*, pemrosesan dapat fokus pada kata-kata penting sehingga kinerja model menjadi lebih efektif dan efisien.

5. *Stemming*

Stemming pada tahap *text preprocessing* merupakan proses menghapus kata yang memiliki imbuhan menjadi kata dasar menggunakan PySastrawi untuk kata dengan bahasa Indonesia (Hana *et al*., 2020). Tujuan dari *stemming* adalah untuk menyederhanakan variasi kata, mengurangi kompleksitas data, meningkatkan konsistensi dalam klasifikasi teks, dan mengenali suatu kata sebagai nilai yang sama (Dogra *et al*., 2022).

2.4 Tokenisasi

Tokenisasi adalah proses memisahkan data menjadi kata dalam bentuk angka (Vasiu *and* Potolea, 2020). Proses tokenisasi dilakukan dengan menyusun daftar kata unik dan menentukan urutan kemunculannya (Mostafa *et al*., 2021). Kata yang memiliki jumlah kemunculan tertinggi diberi indeks terkecil. Manfaat tokenisasi adalah mengubah teks menjadi angka agar dapat dibaca oleh mesin untuk mencapai nilai akurasi optimal dalam model klasifikasi teks (Gasparetto *et al*., 2022).

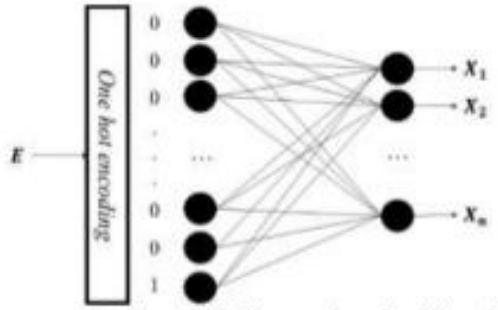
2.5 *Pad Sequence*

Pad sequence merupakan tahap menyamakan panjang *token* dalam sebuah data (Ezhilarasi *and* Maheswari, 2021). Proses *pad sequence* dilakukan dengan menambahkan nilai tertentu diawal atau akhir kalimat (Chotirat *and* Meesad, 2021). Tujuan *pad sequence* adalah menghasilkan data dengan panjang yang seragam agar model *deep learning* dapat memproses data secara efisien.

2.6. *Embedding Layer*

Embedding layer merupakan tahapan dalam model *neural network* yang digunakan untuk mengubah data dalam bentuk teks menjadi representasi vektor

(Egger, 2022). Tujuan dilakukan *embedding layer* adalah agar model dapat memahami hubungan antar kata (Wang *et al.*, 2020). Cara kerja *embedding layer* adalah dengan melakukan *input* data dalam bentuk teks, baik kata maupun kalimat. Data dalam bentuk kalimat dipisahkan menjadi kata-kata individual. Masing-masing kata diberikan bobot nilai dalam bentuk matriks. Setiap baris dalam matriks mewakili bobot nilai dari satu kata. *Entry* matriks yang dihasilkan pada tahap *embedding layer* dijadikan *output* untuk diproses pada tahap berikutnya. Ilustrasi *embedding layer* dapat dilihat pada Gambar 2.2.



Gambar 2.2. Ilustrasi *embedding layer*

Pada Gambar 2.2. dapat dilihat bahwa proses *embedding layer* dimulai dengan *input* E. Nilai ini berupa bilangan bulat hasil konversi dari teks. Nilai *input* ini kemudian diubah menjadi *one hot encoding* dengan nilai 0 dan 1 dalam bentuk kolom. Hasil *one hot encoding* dari *input* E akan digunakan untuk mendapatkan nilai *output* dari proses *embedding layer.* Persamaan *embedding layer* dapat dilihat pada Persamaan (2.1).

$$X_i = E_i W \tag{2.1}$$

E_i merupakan representasi dari *token input,* $i = 1,2, \dots, n$ dengan n merupakan banyaknya *input,* W merupakan matriks bobot, X_i merupakan vektor *embedding*. Hasil *embedding layer* X_i akan digunakan untuk proses selanjutnya.

2.7 *Bidirectional Encoder Representation from Transformer* (BERT)

Augmentasi BERT adalah teknik augmentasi berbasis *Transformer* yang memahami hubungan antar kata menggunakan mekanisme *self-attention* (Devlin *et al.*, 2019). Dalam prosesnya, BERT menentukan kata pengganti yang sesuai dengan konteks kalimat melalui pembobotan nilai menggunakan fungsi *softmax*, dimana calon kata pengganti dengan nilai tertinggi akan dipilih sebagai pengganti kata yang di*mask*. Kata-kata yang dicalonkan sebagai pengganti berasal dari korpus teks yang digunakan dalam pelatihan model. BERT juga menghitung *positional encoding* untuk mempertahankan informasi posisi kata dalam kalimat sebelum melakukan *masking*, sehingga model tetap dapat memahami urutan dan struktur kalimat meskipun proses masking dilakukan secara acak.

Tahapan dalam penerapan BERT untuk augmentasi adalah sebagai berikut.

1. Melakukan tokenisasi dan embedding dengan tahapan dalam melakukan tokenisasi, embedding, *positional encoding, segment embedding* dan dilakukan *masking* dari kalimat *input.*
2. Melakukan *self attention* untuk menangkap hubungan antar kata pada kalimat *input.*
3. Melakukan *feed forward layer*
4. Melakukan *softmax* untuk menentukan probabilitas dari setiap token yang dicalonkan sebagai pengganti kata yang di*mask.*
5. Melakukan *Argmax* untuk menentukan token pengganti dengan probabilitas tertingi.

Persamaan yang digunakan dapat dilihat pada Persamaan (2.2), Persamaan (2.3), Persamaan (2.4), Persamaan (2.5), Persamaan (2.6), Persamaan (2.7), Persamaan (2.8), dan Persamaan (2.9).

$$\hat{P}_{(pos,2i)} = sin\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.2}$$

$$\hat{P}_{(pos,2i+1)} = cos\left(\frac{pos}{10000^{\frac{2i}{d_{model}}}}\right) \tag{2.3}$$

$$Z_{(pos_i)} = \hat{P}_{(pos_i)} + \hat{S}_{(pos)} \tag{2.4}$$

$$K = Z_{(pos_i)} W_K \tag{2.5}$$

$$Q = Z_{(pos_i)} W_Q \tag{2.6}$$

$$V = Z_{(pos_i)} W_V \tag{2.7}$$

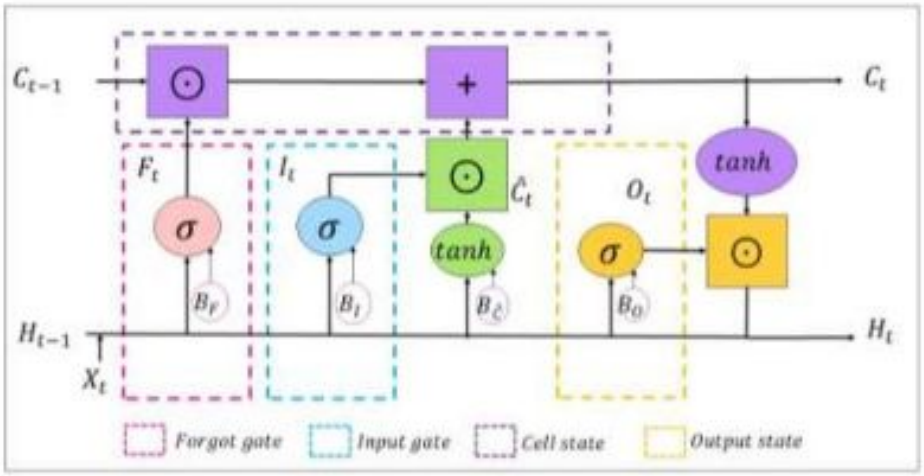
$$A = Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{2.8}$$

$$X_k = ReLU(AW_k + b_k) \tag{2.9}$$

Dimana, pos merupakan posisi kata dalam urutan *input*, i merupakan indeks dimensi, d_{model} merupakan panjang vektor representasi kata, $Z_{(pos_i)}$ merupakan hasil penjumlah *positionalencoding*, $\hat{S}_{(pos)}$ merupakan vektor dari posisi kalimat dalam suatu data, K merupkan *output* dari proses perhitungan *key*, Q merupakan *output* dari proses perhitungan *query*, V merupakan *output* dari proses perhitungan *value*, A merupakan *output* dari *self attention*, b_k merupakan bias, k merupakan dimensi.

2.8 Arsitektur *Long Short Term Memory* (LSTM)

LSTM merupakan jenis jaringan saraf tiruan yang mampu menyimpan informasi jangka panjang (Salau *and* Yesufu, 2020). Komponen utama pada LSTM yaitu *cell state, forgot gate, input gate,* dan *output gate* (Verma *et al.*, 2023). *Gate* pada LSTM digunakan untuk menyimpan informasi penting dan menghapus informasi yang tidak diperlukan. Ilustrasi LSTM *layer* dapat dilihat pada Gambar 2.3.



Gambar 2.3. Ilustrasi LSTM *Layer*

Pada Gambar 2.3 dapat dilihat bahwa LSTM dimulai dengan langkah pertama yaitu memutuskan informasi apa yang akan dihapus menggunakan *input* kata yang telah diubah dalam bentuk angka melalui proses *embedding* X_t . X_t merupakan *input* LSTM *layer.* Selain menggunakan X_t sebagai *input*, digunakan H_{t-1} sebagai *hidden layer* awal. Informasi dihapus pada *forgot gate* dengan menerapkan fungsi aktivasi *sigmoid.* Langkah kedua, yaitu menentukan informasi baru pada *input gate* dengan menambahkan fungsi aktivasi *sigmoid.* Informasi dari *forgot gate* dan *input gate* digunakan untuk menentukan informasi baru dari C_{t-1} menjadi C_t. Untuk

menentukan *output* informasi, LSTM memproses X_t dan H_{t-1} menggunakan fungsi aktivasi *sigmoid* sehingga menghasilkan nilai O_t. Selanjutnya, nilai O_t dikalikan dengan C_t yang telah diproses melalui fungsi tanh untuk menghasilkan *output* H_t.

Arsitektur LSTM bergerak dari $t = 0$ hingga $t = T$, dimana T merupakan panjang urutan suatu informasi. Persamaan LSTM *layer* dapat dilihat pada Persamaan (2.10), Persamaan (2.11), Persamaan (2.12), Persamaan (2.13), Persamaan (2.14), dan Persamaan (2.15).

$$F_t = sigmoid(W_{XF} X_t + W_{HF}\, H_{t-1} + B_F) \tag{2.10}$$

$$I_t = sigmoid(W_{XI} X_t + W_{HI}\, H_{t-1} + B_I) \tag{2.11}$$

$$\hat{C}_t = tanh(W_{X\hat{C}}\, X_t + W_{H\hat{C}}\, H_{t-1} + B_{\hat{C}}) \tag{2.12}$$

$$C_t = F_t \odot C_{t-1} + I_t \odot \hat{C}_t \tag{2.13}$$

$$O_t = sigmoid(W_{XO}\, X_t + W_{HO}\, H_{t-1} + B_O) \tag{2.14}$$

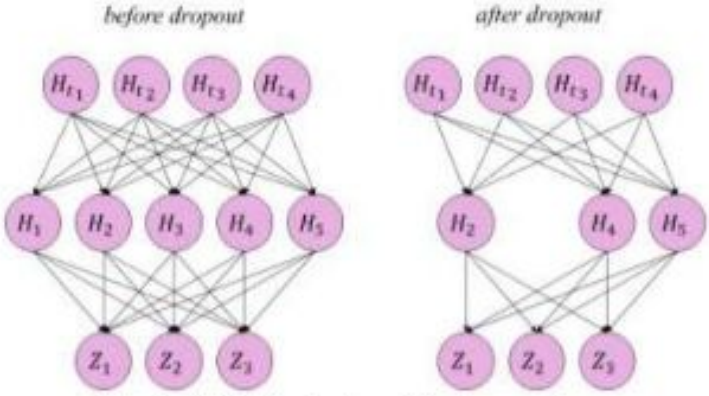
$$H_t = O_t \odot tanh(C_t) \tag{2.15}$$

Dimana, *sigmoid* merupakan fungsi aktivasi untuk mengubah nilai yang berasal dari hasil perhitungan F_t, I_t, dan O_t menjadi rentang 0 hingga 1, $tanh$ merupakan fungsi aktivasi untuk mengubah nilai dari hasil perhitungan $\hat{C}_t$ dan H_t menjadi rentang -1 hingga 1, $\odot$ merupakan operator perkalian elemen demi elemen, X_t merupakan *input* pada waktu ke- t. W_{XF}, W_{XI}, $W_{X\hat{C}}$, W_{XO} berturut-turut merupakan matriks bobot yang menghubungkan *input* X_t ke *forgot gate, input gate, candidate cell state,* dan *output gate*. W_{HF}, W_{HI}, $W_{H\hat{C}}$, W_{HO} berturut-turut merupakan matriks bobot yang menghubungkan H_{t-1} ke *forgot gate, input gate, candidate cell state,* dan *output gate*. Variabel B_F, B_I, $B_{\hat{C}}$, B_O berturut-turut merupakan bias untuk *forgot gate, input gate, candidate cell state,* dan *output gate*. F_t, I_t, $\hat{C}_t$, C_t, O_t, H_t, secara

berurutan merupakan interpretasi nilai dari *forgot gate, input gate, candidate memory cell, cell state, output gate,* dan *hidden state* pada waktu ke-t. H_t adalah *output* dari LSTM yang akan digunakan ke langkah berikutnya dalam model.

2.9 *Dropout Layer*

Dropout layer adalah teknik yang digunakan untuk mencegah *overfitting* (Mehta *and* Passi, 2022). Teknik ini bekerja dengan cara menonaktifkan beberapa layer secara acak pada setiap iterasi selama proses pelatihan berlangsung, sehingga model tidak bergantung pada neuron tertentu. Ilustrasi *dropout layer* dapat dilihat pada Gambar 2.4.

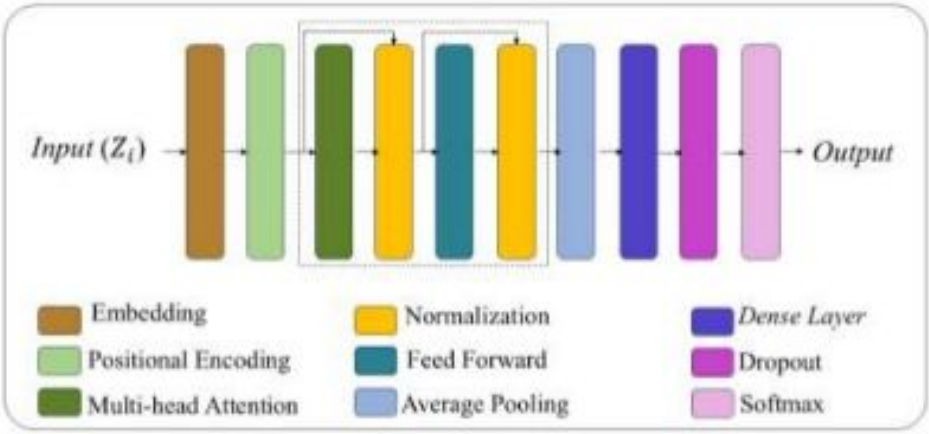


Gambar 2.4. Ilustrasi model *dropout layer*

Pada Gambar 2.4, dapat diketahui bahwa sebelum *dropout* semua neuron pada *layer* tersembunyi yaitu H_1 hingga H_5 menerima sinyal dari neuron *input* H_{t_1} hingga H_{t_4} dan mengirimkan sinyal ke neuron *output* Z_1 hingga Z_3. Setelah proses *dropout,* hanya neuron $H_2, H_4,$ dan H_5 yang menerima dan mengirimkan sinyal, karena beberapa neuron pada *layer* tersembunyi dinonaktifkan.

2.10Arsitektur *Transformer*

Transformer merupakan arsitektur yang dapat digunakan dalam pemrosesan bahasa dengan komponen utama yaitu encoder (Malik *et al*., 2022). Encoder digunakan untuk memproses *input* teks melalui tahapan yaitu *embedding*, *self attention, multi-head attention, normalization layer,* dan *feed forward layer*. *Transformer* mampu menjaga keterkaitan antar kata yang memiliki jarak jauh secara pararel (Liu *et al.*, 2020). Ilustrasi arsitektur *Transformer* dapat dilihat pada Gambar 2.5.



Gambar 2.5. Ilustrasi arsitektur *Transformer*

Berdasarkan Gambar 2.5, proses *Transformer* dimulai dari *embedding*, *positional encoding, multi-head attention, normalitation layer,* dan *feed forward layer* dalam menangkap hubungan antar kata pada data serta digunakan *average pooling* dan *dense layer* untuk mengoptimalkan data.

2.10.1 *Self Attention*

Self attention merupakan teknik untuk menangkap hubungan antar kata dalam satu urutan data (Saeidi *et al*., 2020). *Self attention* dilakukan dengan tiga tahap, yaitu membaca informasi *input* dalam urutan yang sama, menentukan kata-kata

yang relevan, dan memberikan nilai aktual dari kata yang relevan. Pada proses ini, model membuat tiga matriks yaitu *key, value,* dan *query* dari setiap *input layer* (Hernández *and* Amigó, 2021). Matriks *key* merupakan kata yang sedang dianalisis, merupakan kata lainnya dalam urutan *input*, dan merupakan bobot nilai. Pada tahap ini, digunakan matriks Z_i sebagai *input*.

2.10.2 *Multi-head Attention*

Multi-head attention adalah bagian dari *Transformer* yang bekerja dengan membagi hasil *self-attention* menjadi beberapa bagian yang lebih kecil, yang disebut dengan *head* (Qiu *and* Yang, 2022). Setiap *head* menangkap hubungan yang berbeda antar kata dalam urutan teks dan melakukannya secara paralel. Hasil perhitungan dari masing-masing *head* kemudian digabungkan untuk menghasilkan *output multi-head attention* (Li *et al*., 2021).

2.10.3 *Normalization Layer*

Normalization layer merupakan lapisan yang membantu meningkatkan kinerja model dengan cara menormalkan mean dan varians dari *input* di setiap *layer* (Ba *et al.,* 2016) . Persamaan *normalization layer* dapat dilihat pada Persamaan (2.16), Persamaan (2.17), dan Persamaan (2.18).

$$\bar{x} = \frac{1}{d}\sum_{i=1}^{d} Z_i \tag{2.16}$$

$$s_i^2 = \frac{1}{d-1}\sum_{i=1}^{d}(Z_i - \bar{x}_i)^2 \tag{2.17}$$

$$n_i = \frac{Z_i - \bar{x}_i}{\sqrt{s_i^2}} \tag{2.18}$$

2.10.4 *Feed Forward Layer*

Feed forward layer merupakan bagian arsitektur *Transformer* yang berfungsi untuk menyaring fitur dari data pada setiap entri matriks. *Feed forward layer* memproses setiap *token* secara independen tanpa mempedulikan posisi tetapi tetap mempertahankan informasi kontekstual berkat mekanisme attention sebelumnya (Ohiri, 2024).

2.10.5 *Average Pooling*

Average pooling merupakan teknik yang digunakan untuk mengurangi jumlah parameter agar model menjadi lebih sederhana dan menyaring informasi penting dari seluruh peta fitur (Lin *et al.*, 2014). Tujuannya adalah untuk mengurangi *overfitting*. *Average pooling* mengurangi jumlah parameter dengan cara menggantikan *fully connected layers* dengan nilai dalam setiap kanal.

2.10.6 *Dense Layer*

Dense layer berfungsi untuk menggabungkan informasi dalam matriks *input* menjadi representasi yang lebih relevan dengan cara menghubungkan setiap neuron ke neuron sebelumnya dan dilakukan perkalian dengan bobot tertentu (Domor *et al.*, 2024). Adapun rumus perhitungan fungsi *dense layer* dituliskan pada Persamaan (2.19).

$$\hat{d} = softmax(W_f\, n_i + b_f) \tag{2.19}$$

W_f merupakan nilai bobot untuk *dense layer*, R_k merupakan *input* dari lapisan *feed forward layer*, b_f merupakan bias, S merupakan nilai *output*.

2.10.7 Fungsi Aktivasi

Fungsi aktivasi merupakan komponen pada jaringan syaraf tiruan yang digunakan untuk mengubah *output* yang akan di teruskan ke neuron selanjutnya (Apicella et al., 2021). Pada penelitian ini, menggunakan fungsi aktivasi *sigmoid, tanh, softmax* dan ReLu. Fungsi aktivasi *sigmoid* adalah aturan yang menemankan setiap elemen dalam himpunan domain dengan tepat satu elemen dalam himpunan kodomain yang memiliki rentang nilai dari 0 sampai 1. Adapun rumus perhitungan fungsi aktivasi *sigmoid* dituliskan pada Persamaan (2.20).

$$\hat{b}_i = sigmoid(\hat{d}) = \frac{1}{1 + e^{-\hat{d}}} \tag{2.20}$$

$\hat{b}_i$ merupakan hasil dari fungsi aktivasi *sigmoid*, $\hat{d}$ merupakan *input* dari fungsi aktivasi *sigmoid*, e merupakan bilangan *Euler*. Fungsi aktivasi lainnya yang digunakan pada penelitian ini adalah *tanh* yang dapat dilihat pada Persamaan (2.21).

$$\hat{h}_i = tanh(\hat{b}_i) = \frac{e^{\hat{b}_i} - e^{-\hat{b}_i}}{e^{\hat{b}_i} + e^{-\hat{b}_i}} \tag{2.21}$$

Nilai *output* dari fungsi aktivasi *tanh* antara -1 sampai dengan 1. Selanjutnya, fungsi aktivasi ReLU digunakan untuk memetakakn nilai *input* ke nilai *output* dengan menggunakan Persamaan (2.22).

$$\hat{f}_i = ReLU(\hat{h}_i) = max\ (0, \hat{h}_i) \tag{2.22}$$

$\hat{h}_i$ merupakan nilai *input*. Jika nilai $\hat{h}_i$ positif maka nilai *output* adalah nilai $\hat{h}_i$ itu sendiri, sedangkan jika nilai $\hat{h}_i$ negatif maka nilai *output* adalah 0. Selanjutnya, fungsi aktivasi lain yang digunakan adalah *softmax*. Fungsi aktivasi *softmax* merupakan fungsi yang digunakan pada *layer output* dari model klasifikasi *multi label* atau *multi class*. Fungsi aktivasi *softmax* dapat dilihat pada Persamaan (2.23).

$$\hat{s}_i = softmax(\hat{f}_i) = \frac{e^{\hat{f}_i}}{\sum_{i=1}^{n} e^{\hat{f}_i}} \quad (2.23)$$

Softmax memiliki rentang nilai dari 0 sampai 1 yang diterapkan pada nilai *output*.

2.11 *Loss Function*

Loss function merupakan fungsi yang digunakan untuk mengukur kinerja model dalam memprediksi data (Zhang *et al.*, 2019). *Loss function* memiliki beberapa tipe, salah satunya adalah *categorical cross-entropy loss*. Fungsi *categorical cross-entropy loss* digunakan pada klasifikasi teks *multi-label* dan *multi-class* (Rongbo *et al*., 2020). Rumus untuk fungsi *categorical cross-entropy loss* dapat dilihat pada Persamaan (2.24).

$$l = -\sum_{i=1}^{m} \hat{y}_i \ln \hat{s}_i \quad (2.24)$$

$\hat{y}_i$ merupakan entri matrix dari nilai *label* sebenarnya, $\hat{s}_i$ merupakan entri matriks dari nilai prediksi *label* pada baris ke-i, m merupakan jumlah baris, n merupakan jumlah kolom, dan l merupakan nilai dari *loss function* menggunakan tipe *categorical cross-entropy loss*.

2.12 Adam *Optimizer*

Adam merupakan metode optimasi stokastik yang efisien karena hanya membutuhkan gradien orde pertama dengan kebutuhan memori yang sangat rendah (Yi *et al*., 2020). Metode ini bekerja dengan menghitung tingkat pembelajaran adaptif untuk setiap parameter secara individu, berdasarkan estimasi momen pertama dan momen kedua dari gradien (Chen *et al.*, 2020). Proses menghitung

momen pertama dan kedua nilai momen pertama dan momen kedua, dapat dilihat pada Persamaan (2.25) dan Persamaan (2.26) berikut.

$$r_t = \beta_t r_{t-1} + (1 - \beta_t)\hat{g}_t \tag{2.25}$$

$$a_t = \beta_t a_{t-1} + (1 - \beta_t)\hat{g}_t^{\,2} \tag{2.26}$$

r_t dan a_t adalah perkiraan momen pertama (rata-rata) dan momen kedua dari gradien masing-masing. Parameter β_t adalah *decay rate* atau koefisien estimasi untuk momen pertama dan $\hat{g}_t$ adalah gradien. Metode momen digunakan untuk menghitung nilai momen pertama r_t dan momen kedua a_t dimana dapat dilihat pada Persamaan (2.27) dan Persamaan (2.28).

$$\hat{r_t} = \frac{r_t}{(1 - \beta_1^t)} \tag{2.27}$$

$$\hat{a_t} = \frac{a_t}{(1 - \beta_2^t)} \tag{2.28}$$

Kemudian, bobot (W_t) diperbarui tiap *epoch* menggunakan *learning rate* sebesar c menggunakan Persamaan (2.29).

$$w_t = w_{t-1} - \frac{c.\hat{r_t}}{\sqrt{\hat{a_t}} + \varepsilon} \tag{2.29}$$

dimana w_t merupakan parameter model pada iterasi ke-t, dan c merupakan laju pembelajaran yang ditentukan.

2.13 *Confussion Matrix*

Confusion matrix merupakan gambaran mengenai keakuratan suatu model dalam melakukan klasifikasi. Berdasarkan Pereira-Kohatsu *et al*., (2019), *confusion matrix* memiliki ukuran 2 × 2 dengan elemen-elemen seperti pada Tabel 2.1.

Tabel 2.1. *Confussion matrix* ukuran 2 × 2

Data Asli	Hasil Prediksi	
	Ujaran Kebencian	*Abusive*
Ujaran Kebencian	*True Positive* (TP)	*False Negatif* (FN)
Abusive	*False Positive* (FP)	*True Negative* (TN)

Keterangan,

TP : Hasil prediksi Ujaran Kebencian dan label sebenarnya Ujaran Kebencian

FP : Hasil prediksi Ujaran Kebencian dan label sebenarnya *Abusive*

FN : Hasil prediksi *Abusive* dan label sebenarnya Ujaran Kebencian

TN : Hasil prediksi *Abusive* dan label sebenarnya *Abusive*

Ukuran evaluasi terhadap kinerja model dalam melakukan klasifikasi berdasarkan *confussion matrix* yaitu akurasi, presisi, *recall,* dan *f1-score* (Pereira-Kohatsu *et al*., 2019). Akurasi adalah prediksi benar baik dari data aktual *positif* maupun *negatif* dibagi dengan total jumlah prediksi Pereira-Kohatsu *et al*., (2019). Presisi adalah proporsi dari semua prediksi positif yang benar-benar positif, yang mengukur seberapa akurat model dalam mengidentifikasi kasus positif (Powers, 2020). *Recall* adalah jumlah kasus positif yang berhasil diprediksi dengan benar oleh model, dibandingkan dengan total kasus positif yang ada (Powers, 2020). *F1-score* adalah gambaran keseimbangan antara presisi dan *recall* (Pereira-Kohatsu *et al*., 2019). Evaluasi kinerja model dapat dihitung menggunakan Persamaan (2.30), (2.31), (2.32), dan (2.33).

$$Akurasi = \frac{TP + TN}{TP + FN + TN + FP} \times 100\% \tag{2.30}$$

$$presisi = \frac{TP}{TP + FP} \times 100\% \tag{2.31}$$

$$recall = \frac{TP}{TP + FN} \times 100\% \tag{2.32}$$

$$f1 - score = 2 \times \frac{recall \times preisi}{recall + preisi} \times 100\% \quad (2.33)$$

Nilai kinerja arsitektur memiliki beberapa kategori pada Tabel 2.2.

Tabel 2.2. Kategori nilai kinerja arsitektur

Nilai Kinerja (%)	Kategori
>90	Sangat Baik
81-90	Baik
71-80	Cukup Baik
61-70	Kurang Baik
<60	Gagal

Berdasarkan Tabel 2.2, dapat dilihat bahwa nilai kinerja arsitektur dibawah 60% dikategorikan gagal, sedangkan nilai antara 61% sampai 70% dikategorikan kurang baik. Nilai kinerja dikategorikan cukup baik ketika nilai berada diantara 71% sampai 80%, baik ketika nilai berada diantara 81% sampai 90%, dan sangat baik ketika nilai kinerja lebih dari 90%.

BAB III

METODOLOGI PENELITIAN

3.1 Tempat

Penelitian ini dilakukan di Jurusan Matematika Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Sriwijaya.

3.2 Waktu

Waktu yang dibutuhkan untuk melakukan penelitian ini adalah 6 bulan, yakni dari bulan Oktober 2024 hingga Maret 2025.

3.3 Alat

Alat yang digunakan dalam penelitian ini, sebagai berikut:

1. Komputer dengan sistem operasi Windows 10 64 bit, processor Intel ® Core (TM) i5-2450M CPU @ 2.50GHz (4 CPUs), 2.5GHz, RAM 10240 MB, serta grafika NVIDIA GeForce GT 520M.
2. Aplikasi *Google Colaboratory.*
3. Bahasa pemerogaman *phyton.*

3.4 Metode Penelitian

Adapun langkah-langkah dalam penelitian ini, sebagai berikut:

1. Melakukan deskripsi data menggunakan *dataset multi label hate speech tweet* dalam bahasa Indonesia dari GitHub. *Dataset* dapat diakses melalui link https://github.com/okkyibrohim/id-multi-label-hate-speech-and-abusive-language-detection.

2. Melakukan augmentasi data menggunakan kombinasi metode *back translation* dan augmenentasi BERT. Hasil augmentasi menggunakan metode *back translation* digunakan untuk augmentasi menggunakan metode augmentasi BERT. Perhitungan untuk augmentasi BERT menggunakan Persamaan (2.2), Persamaan (2.3), Persamaan (2.4), Persamaan (2.5), Persamaan (2.6), Persamaan (2.7), Persamaan (2.8), dan Persamaan (2.9).
3. Melakukan *text preprocessing* dengan tahapan sebagai berikut.
   a. Mengubah huruf kapital yang ada pada data menjadi huruf kecil.
   b. Menghapus tanda baca yang ada pada data.
   c. Mengubah kata yang tidak baku menjadi kata baku
   d. Menghapus kata yang sering muncul tetapi tidak memiliki nilai informasi penting
   e. Menghapus kata yang memiliki imbuhan menjadi kata dasar
4. Melakukan pelabelan data menggunakan teknik *label powerset*, yaitu metode pengelompokkan semua *label* menjadi satu *label* tunggal atau kelas. Dalam penelitian ini, terdapat 12 *label* pada *dataset* yang dapat dilihat pada Tabel 3.1.

Tabel 3.1. *Label* ujaran kebencian bahasa Indonesia pada *tweet*

Posisi Bit	***Label***	**Keterangan**
1	HS_1	Kategori umum untuk ujaran kebencian.
2	$Abusive_2$	Ucapan yang bersifat kasar atau menghina tetapi tidak selalu termasuk ujaran kebencian.
3	$HS\_Individual_3$	Ujaran kebencian yang ditujukan kepada individu tertentu.
4	HS_Group_4	Ujaran kebencian yang menyerang sekelompok orang.

Tabel 3.1. *Label* ujaran kebencian bahasa Indonesia pada *tweet*

Posisi bit	*Label*	Keterangan
5	$HS\_Religion_5$	Ujaran kebencian yang menargetkan suatu agama atau kepercayaan tertentu.
6	HS_Race_6	Ujaran kebencian yang menyerang ras atau etnis tertentu.
7	$HS\_Phydical_7$	Ujaran kebencian yang berkaitan dengan kondisi fisik seseorang, seperti disabilitas atau penampilan fisik.
8	HS_Gender_8	Ujaran kebencian yang berkaitan dengan gender atau orientasi seksual.
9	HS_Other_9	Ujaran kebencian yang tidak termasuk dalam kategori spesifik di atas.
10	HS_Weak_{10}	Ujaran kebencian dengan tingkat keparahan rendah, mungkin berupa sindiran atau ejekan ringan.
11	$HS\_Moderate_{11}$	Ujaran kebencian dengan tingkat sedang, yang lebih eksplisit tetapi tidak bersifat ekstrem.
12	HS_Strong_{12}	Ujaran kebencian dengan tingkat keparahan tinggi, mengandung ancaman atau seruan untuk melakukan kekerasan.

Tabel 3.1 menunjukkan 12 *label* pada *dataset* ujaran kebencian bahasa Indonesia. Dengan menerapkan teknik *label powerset,* diperoleh 49 kelas unik yang merupakan kombinasi 12 *label* ujaran kebencian yang dikodekan dalam bentuk vektor *one hot encoding*. Makna untuk masing-maisng bit pada vektor *one hot encoding* dapat dilihat pada Tabel 3.2.

Tabel 3.2 Makna bit dalam vektor *one hot encoding*

bit	Makna
0	Sampel tidak termasuk dalam kategori *label* sesuai posisi
1	Sampel termasuk dalam kategori *label* sesuai posisi

Pada Tabel 3.2 dapat dilihat bahwa jika pada vektor *one hot encoding* memiliki bit 0 diposisi tertentu, artinya sampel tidak masuk kedalam kategori. Sebaliknya, jika memiliki bit 1, maka sampel termasuk kedalam kategori. Kelas unik hasil teknik *label powerset* digunakan sebagai target dalam klasifikasi *multi label* menggunakan kombinasi arsitektur

LSTM dan *Transformer*. Hasil pelabelan menggunakan teknik *label powerset* dapat dilihat pada Tabel 3.3.

Tabel 3.3. Kelas data ujaran kebencian menggunakan pendekatan *label powerset*

Kelas	Pelabelan	Keterangan	Jumlah
H0	0 0 0 0 0 0 0 0 0 0 0 0	Netral	23440
H1	0 1 0 0 0 0 0 0 0 0 0 0	Abusive	6992
H2	1 0 0 1 0 0 0 0 1 0 0 1	Hate speech, hate speech kelompok, lainnya, kuat	400
H3	1 0 0 1 0 0 0 0 1 0 1 0	Hate speech, hate speech kelompok, lainnya, sedang	100
H4	1 0 0 1 0 1 0 0 0 0 0 1	Hate speech, hate speech kelompok, ras, kuat	376
H5	1 0 0 1 0 1 0 0 0 0 1 0	Hate speech, hate speech kelompok, ras, sedang	668
H6	1 0 0 1 1 0 0 0 0 0 0 1	Hate speech, hate speech kelompok, agama, kuat	128
H7	1 0 0 1 1 0 0 0 0 0 1 0	Hate speech, hate speech kelompok, agama, sedang	776
H8	1 0 0 1 1 0 0 1 0 0 1 0	Hate speech, hate speech kelompok, agama, jenis kelamin, sedang	44
H9	1 0 0 1 1 1 0 0 0 0 0 1	Hate speech, hate speech kelompok, agama, ras, kuat	140
H10	1 0 0 1 1 1 0 0 0 0 1 0	Hate speech, hate speech kelompok, agama, ras, sedang	588
H11	1 0 1 0 0 0 0 0 1 0 0 1	Hate speech, hate speech individu, lainnya, kuat	4184
H12	1 0 1 0 0 0 0 1 0 1 0 0	Hate speech, hate speech individu, jenis kelamin, lemah	24
H13	1 0 1 0 0 0 1 0 0 1 0 0	Hate speech, hate speech individu, keadaan fisik, lemah	20
H14	1 0 1 0 0 1 0 0 0 0 0 1	Hate speech, hate speech individu, ras, kuat	48
H15	1 0 1 0 0 1 0 0 0 1 0 0	Hate speech, hate speech individu, ras, lemah	228
H16	1 0 1 0 1 0 0 0 0 0 0 1	Hate speech, hate speech individu, agama, kuat	32
H17	1 0 1 0 1 0 0 0 0 1 0 0	Hate speech, hate speech individu, agama,lemah	360
H18	1 0 1 0 1 1 0 0 0 0 0 1	Hate speech, hate speech individu, agama, ras, kuat	12
H19	1 0 1 0 1 1 0 0 0 1 0 0	Hate speech, hate speech individu, agama, ras, lemah	28
H20	1 1 0 1 0 0 0 0 1 0 0 1	Hate speech, abusive, hate speech kelompok, lainnya, kuat	32
H21	1 1 0 1 0 0 0 0 1 0 1 0	Hate speech, abusive, hate speech kelompok, lainnya, sedang	2732
H22	1 1 0 1 0 0 0 1 0 0 1 0	Hate speech, abusive, hate speech kelompok, jenis kelamin, sedang	156
H23	1 1 0 1 0 0 1 0 0 0 1 0	Hate speech, abusive, hate speech kelompok, keadaan fisik, sedang	136

Tabel 3.3. Kelas data ujaran kebencian menggunakan pendekatan *label powerset*

Kelas	Pelabelan	Keterangan	Jumlah
H24	1 1 0 1 0 1 0 0 0 0 0 1	Hate speech, abusive, hate speech kelompok, ras, kuat	60
H25	1 1 0 1 0 1 0 0 0 0 1 0	Hate speech, abusive, hate speech kelompok, ras, sedang	252
H26	1 1 0 1 0 1 0 1 0 0 1 0	Hate speech, abusive, hate speech kelompok, ras, jenis kelamin, sedang	12
H27	1 1 0 1 1 0 0 0 0 0 0 1	Hate speech, abusive, hate speech kelompok, agama, kuat	56
H28	1 1 0 1 1 0 0 0 0 0 1 0	Hate speech, abusive, hate speech kelompok, agama, sedang	780
H29	1 1 0 1 1 0 0 1 0 0 1 0	Hate speech, abusive, hate speech kelompok, agama, jenis kelamin, sedang	56
H30	1 1 0 1 1 0 1 0 0 0 1 0	Hate speech, abusive, hate speech kelompok, agama, keadaan fisik, sedang	20
H31	1 1 0 1 1 1 0 0 0 0 0 1	Hate speech, abusive, hate speech kelompok, agama, ras, kuat	16
H32	1 1 0 1 1 1 0 0 0 0 1 0	Hate speech, abusive, hate speech kelompok, agama, ras, sedang	76
H33	1 1 1 0 0 0 0 0 1 0 0 1	Hate speech, abusive, hate speech individu, lainnya, kuat	68
H34	1 1 1 0 0 0 0 0 1 1 0 0	Hate speech, abusive, hate speech individu, lainnya, lemah	5956
H35	1 1 1 0 0 0 0 1 0 1 0 0	Hate speech, abusive, hate speech individu, jenis kelamin, lemah	832
H36	1 1 1 0 0 0 1 0 0 1 0 0	Hate speech, abusive, hate speech individu, keadaan fisik, lemah	980
H37	1 1 1 0 0 0 1 1 0 1 0 0	Hate speech, abusive, hate speech individu, keadaan fisik, jenis kelamin, lemah	68
H38	1 1 1 0 0 1 0 0 0 1 0 0	Hate speech, abusive, hate speech individu, ras, lemah	228
H39	1 1 1 0 1 0 0 0 0 1 0 0	Hate speech, abusive, hate speech individu, agama, lemah	488
H40	1 1 1 0 1 0 0 1 0 1 0 0	Hate speech, abusive, hate speech individu, agama, jenis kelamin, lemah	28
H41	1 1 1 0 1 0 1 0 0 1 0 0	Hate speech, abusive, hate speech individu, agama, keadaan fisik, lemah	28
H42	1 1 1 0 1 0 1 1 0 1 0 0	Hate speech, abusive, hate speech individu, agama, keadaan fisik, jenis kelamin, lemah	16
H43	1 1 1 0 1 1 0 0 0 1 0 0	Hate speech, abusive, hate speech individu, agama,, ras, lemah	60

5. Melakukan tokenisasi dengan cara memisahkan kalimat pada data menjadi satuan kata yang memiliki angka atau disebut *token*. Setiap *token* dipetakan mengguanakan representasi numerik untuk diproses pada model yang digunakan pada saat klasifikasi teks.

6. Melakukan *Pad Sequences* yang digunakan untuk menyamakan panjang dari masing-masing *token* pada data setelah dilakukan tokenisasi.
7. Mengimplementasi arsitektur LSTM dan *Transformer.* Pada tahap ini, proses dibagi menjadi dua bagian yaitu data *training* dan data *testing*. Proses data *training* menggunakan 80% data dan data *testing* menggunakan 20% data. Implementasi arsitektur LSTM dan *Transformer* dapat diuraikan sebagai berikut:
   a. Melakukan Data *Training*
      1) *Menginput dataset* ujaran kebencian twitter yang akan dihitung menjadi vektor dalam *embedding layer* menggunakan Persamaan (2.1).
      2) Melakukan perhitungan pada lapisan LSTM yaitu *forgot gate, input gate, candidate memory cell, cell state, output gate,* dan *hidden gate*. Perhitungan untuk *forgot gate* menggunakan Persamaan(2.10). Perhitungan untuk *input gate* menggunakan Persamaan (2.11), Perhitungan *candidate memory cell* menggunakan Persamaan (2.12). Perhitungan *cell state* menggunakan Persamaan (2.13). Perhitungan untuk *output gate* menggunakan Persamaan (2.14). Perhitungan untuk *hidden gate* menggunakan Persamaan (2.15).
      3) Menghitung matriks *Query, Key, Value,* dan matriks *output* pada tahapan *self attention. Query* dihitung menggunakan Persamaan (2.5). *Key* dihitung menggunakan Persamaan (2.6). *Value*

dihitung menggunakan Persamaam (2.7). Matriks *output* dari tahapan *self attention* dihitung menggunakan Persamaan (2.8).

4) Menghitung *Multi-head Attention* dengan menggabungkan setiap *head* hasil dari tahapan *self attention.*
5) Melakukan *normalization layer* dengan menggunakan Persamaan (2.16) untuk menghitng nilai rata-rata, Persamaan (2.17) untuk menghitung nilai varians, Persamaan (2.18) untuk menghitung nilai normalisasi.
6) Menghitung nilai untuk *feed forward layer* menggunakan Persamaan (2.9).
7) Menghitung nilai untuk *dense layer* menggunakan Persamaan (2.19).
8) Menghitung nilai fungsi aktivasi menggunakan Persamaan (2.20) untuk fungsi aktivasi *sigmoid,* Persamaan (2.21) untuk fungsi aktivasi *tanh,* Persamaan(2.22) untuk fungsi aktivasi ReLu, dan Persamaan (2.23) untuk fungsi aktivasi *softmax*.
9) Menghitung nilai *loss function* menggunakan Persamaan (2.24).
10) Menghitung nilai *adam layer* dengan menggunakan Persamaan (2.25) dan Persamaan (2.26) untuk momen pertama, Persamaan (2.27) dan Persamaan (2.28) untuk momen kedua, serta Persamaan (2.29) untuk menghitung bobot *adam layer.*
11) Mengulangi proses (1) sampai (10) hingga *epoch* terakhir.

b. Melakukan Data *Testing*

Data *testing* dilakukan untuk mengetahui kemampuan model yang telah di *training*. Tujuan data *testing* untuk mengetahui keakuratan model yang didesain dalam melakukan klasifikasi teks ujaran kebencian *multi label* pada data twitter.

8. Melakukan evaluasi kinerja model. Pada tahap ini akan dilakukan perhitungan evaluasi kinerja model berupa nilai akurasi menggunakan Persamaan (2.30), presisi menggunakan Persamaan (2.31), *recall* menggunakan Persamaan (2.32), dan *f1-score* menggunakan Persamaan (2.33). Hasil evaluasi dianalisis dan diinterpretasikan hasilnya.

BAB IV

HASIL DAN PEMBAHASAN

4.1 Deskripsi Data

Pada penelitian ini, data yang digunakan diperoleh dari GitHub melalui laman https://github.com/okkyibrohim/id-multi-label-hate-speech-and-abusive-language-detection. Jumlah data yang terdapat pada *dataset* kumpulan *tweet* bahasa Indonesia, yakni sebanyak 13.169 data dengan 12 *label* yaitu ujaran kebencian (HS_1), kata kasar ($Abusive_2$), ujaran kebencian perorangan ($HS\_Individual_3$), ujaran kebencian kelompok (HS_Group_4), ujaran kebencian terhadap agama ($HS\_Religion_5$), ujaran kebencian terhadap suku atau ras (HS_Race_6), ujaran kebencian terhadap keadaan fisik ($HS\_Physical_7$), ujaran kebencian terhadap jenias kelamin (HS_Gender_8), ujaran kebencian lainnya (HS_Other_9), ujaran kebencian yang bersifat lemah (HS_Weak_{10}), ujaran kebencian yang bersifat menengah ($HS\_Moderate_{11}$), dan ujaran kebencian yang bersifat kuat (HS_Strong_{12}). Beberapa data sampel teks beserta label disajikan pada Tabel 4.1.

Tabel 4.1. Beberapa contoh pada *dataset tweet* ujaran kebencian

No.	*Tweet*	HS_1	⋯	HS_Strong_{12}
1	- disaat semua cowok berusaha melacak perhatian gue. loe lantas remehkan perhatian yg gue kasih khusus ke elo. basic elo cowok bego ! ! !'	1	⋯	0
2	RT USER: USER siapa yang telat ngasih tau elu?edan sarap gue bergaul dengan cigax jifla calis sama siapa noh licew juga'	0	⋯	0
3	41. Kadang aku berfikir, kenapa aku tetap percaya pada Tuhan padahal aku selalu jatuh berkali-kali. Kadang aku merasa Tuhan itu ninggalkan aku sendirian. Ketika orangtuaku berencana berpisah, ketika kakakku lebih memilih jadi Kristen. Ketika aku anak ter	0	⋯	0

Tabel 4.1. Beberapa contoh pada *dataset tweet* ujaran kebencian

No.	*Tweet*	HS_1	⋯	HS_Strong_{12}
4	USER USER AKU ITU AKU\n\nKU TAU MATAMU SIPIT TAPI DILIAT DARI MANA ITU AKU'	0	⋯	0
5	USER USER Kaum cebong kapir udah keliatan dongoknya dari awal tambah dongok lagi hahahah'	1	⋯	0
6	USER Ya bani taplak dkk \xf0\x9f\x98\x84\xf0\x9f\x98\x84\xf0\x9f\x98\x84'	1	⋯	0
7	deklarasi pilkada 2018 aman dan anti hoax warga dukuh sari jabon	0	⋯	0
⋮	⋮	⋮	⋯	⋮
13169	USER Mana situ ngasih(": itu cuma foto ya kutil onta'	1	⋯	0

4.2 Augmentasi Data

Augmentasi dilakukan untuk menambah jumlah data dari *dataset* pada Tabel 4.1 tanpa perlu menambahkan data baru. Pada penelitian ini, teknik augmentasi yang digunakan adalah *back translation* dan augmentasi BERT. Adapun tahapan dalam melakukan proses augmentasi sebagai berikut:

1. Menerjemahkan data ke bahasa Inggris

   Menerapkan *back translation* untuk menerjemahkan data dari bahasa Indonesia ke bahasa Inggris. Hasil menerjemahkan data dari bahasa Indonesia ke bahasa Inggris dapat dilihat pada Gambar 4.1.

Gambar 4.1 Hasil penerjemahan bahasa Indonesia ke bahasa Inggris

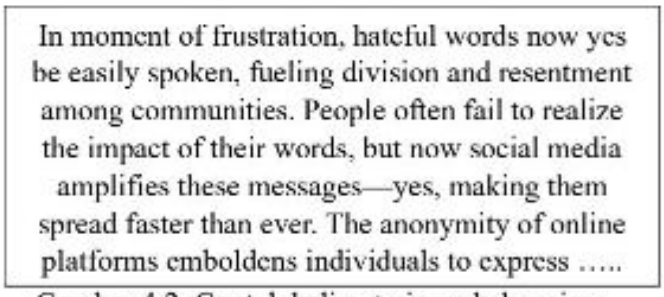
2. Menerapkan teknik BERT

   Proses augmentasi memiliki beberapa tahapan, yaitu tokenisasi dan *embedding, selft attention, feed forward network,* fungsi aktivasi *softmax,* dan

argmax. Untuk proses perhitungan manual augmentasi BERT, digunakan kalimat *input "only dumb-brained hummans are now shouting"*. Proses augmentasi menggunakan teknik BERT adalah sebagai berikut:

Tahap 1. Tokenisasi

Tokenisasi adalah proses mengubah setiap kata dalam kalimat menjadi angka untuk membantu komputer memahami informasi. Contoh informasi ujaran kebencian dapat dilihat pada Gambar 4.2.



Gambar 4.2. Contoh kalimat ujaran kebencian

Tahap pertama dalam tokenisasi yaitu memberikan nomor unik pada masing-masing kata berdasarkan urutan kemunculan yang disebut *vocabulary building*. Hasil *vocabulary building* dapat dilihat pada Tabel 4.2.

Tabel 4.2. *Vocabulary building* data ujaran kebencian bahasa Inggris

Word	*Token*
in	1
moment	2
⋮	⋮
now	7
yes	8
⋮	⋮
only	52
when	53
⋮	⋮
shouting	66
stupid	67
dumb-brained	117
delete	118
⋮	⋮
hummans	337
no	338

Tabel 4.2. *Vocabulary building* data ujaran kebencian bahasa Inggris

Word	*Token*
⋮	⋮
are	821

Berdasarkan Tabel 4.2, *vocabulary building* menampilkan kata dengan diberikan *token* unik untuk merepresentasikan urutan kemunculan kata dalam bentuk numerik. Tokenisasi mengubah setiap kata menjadi *token* dan setiap *token* diberikan indeks. Hasil tokenisasi dapat dilihat pada Tabel 4.3.

Tabel 4.3. Tokenisasi kata bahasa Inggris

Word	*Token*	*Indeks*
in	1	6
moment	2	179
⋮	⋮	⋮
dumb-brained	7	117
yes	8	158
⋮	⋮	⋮
only	52	52
when	53	32
⋮	⋮	⋮
shouting	66	66
stupid	67	120
now	117	7
delete	118	89
⋮	⋮	⋮
hummans	337	337
no	338	90
⋮	⋮	⋮
are	5850	821

Berdasarkan Tabel 4.3, dapat dilihat nilai tokenisasi dari masing-masing kata. Untuk kalimat *input* "*only dumb-brained hummans are now shouting*", diperoleh hasil tokenisasi sebagai berikut.

"*only*":52, "*dumb-brained*":117, "*hummans*":337, "*are*":821, "*now*":7, "*shouting*":66

Tahap 2. *Embedding*

Pada tahap embedding, terdiri dari beberapa tahapan yaitu menghitung embedding, *positional encoding*, dan *segment embedding* sebagai berikut.

1. Menghitung *embedding* dari masing-maisng kata menggunakan indeks hasil tokenisasi. Setiap *token* direpresentasikan sebagai *one hot encodding,* yaitu vektor dengan nilai 1 hanya di indeks *token* tersebut dalam kosakata atau dapat dengan mudah didcteksi berdasarkan urutannya dalam satu kalimat. Pada *input* hasil tokenisasi, diperoleh urutan indcks yaitu kata *now* urutan pertama, kata *only* urutan kedua, kata *shouting* urutan ketiga, kata *dumb-brained* urutan keempat, kata *hummans* urutan kelima, kata *are* urutan keenam. Schingga, diperolch *one hot encodding* sebagai berikut.

$$\hat{E}_1 = k(\text{"}only\text{"})=[0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$\hat{E}_2 = k(\text{"}dumb\text{-}brained\text{"})=[0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0]$$

$$\hat{E}_3 = k(\text{"}hummans\text{"})=[0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0]$$

$$\hat{E}_4 = k(\text{"}are\text{"})=[0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1]$$

$$\hat{E}_5 = k(\text{"}now\text{"})=[1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$\hat{E}_6 = k(\text{"}shouting\text{"})=[0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0]$$

Untuk menghitung nilai *embedding*, diperlukan matriks bobot W_{token} dengan entri 6×5 dimana jumlah baris menyesuaikan jumlah *input* kata. Berikut adalah matriks bobot W_{token} untuk proses *embedding* augmentasi BERT.

$$W_{token} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Menghitung nilai *embedding* menggunakan Persamaan (2.1) berikut.

$$E = \hat{E}_i W_{token}$$

$$E_1 = \hat{E}_1 W_{token}$$

$$= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 2{,}1 & 0 \\ 0 & 0 & 0 & 1{,}4 & 1 \\ 1 & 0 & 0 & 1{,}8 & 0 \\ 0 & 0 & 1 & 2{,}3 & 0 \\ 0 & 1 & 1 & 2{,}6 & 1 \\ 1 & 0 & 0 & 1{,}3 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 1{,}4 & 1 \end{bmatrix}$$

Dilakukan perhitungan yang sama untuk *token embedding* ke 2,3, 4, 5, dan 6, sehingga diperoleh hasil sebagai berikut.

$$E_2 = \begin{bmatrix} 0 & 0 & 1 & 2{,}3 & 0 \end{bmatrix}$$

$$E_3 = \begin{bmatrix} 0 & 1 & 1 & 2{,}6 & 1 \end{bmatrix}$$

$$E_4 = \begin{bmatrix} 1 & 0 & 0 & 1{,}3 & 0 \end{bmatrix}$$

$$E_5 = \begin{bmatrix} 1 & 0 & 0 & 2{,}1 & 0 \end{bmatrix}$$

$$E_6 = \begin{bmatrix} 1 & 0 & 0 & 1{,}8 & 0 \end{bmatrix}$$

Dilakukan penggabungan matriks E_i sehingga diperoleh matriks sebagai berikut.

$$E = \begin{bmatrix} 0 & 0 & 0 & 1{,}4 & 1 \\ 0 & 0 & 1 & 2{,}3 & 0 \\ 0 & 1 & 1 & 2{,}6 & 1 \\ 1 & 0 & 0 & 1{,}3 & 0 \\ 1 & 0 & 0 & 2{,}1 & 0 \\ 1 & 0 & 0 & 1{,}8 & 0 \end{bmatrix}$$

2. Menghitung *positional encoding* menggunakan Persamaan (2.23) dan Persamaan (2.24). Diketahui bahwa pos bergerak dari 0 sampai 5, i bergerak dari 0 sampai 4, dan d=5.

   $pos = 0$, maka

$$PE_{(0,0)} = sin\left(\frac{0}{10000\frac{0}{5}}\right) = \sin(0) = 0$$

$$PE_{(0,1)} = cos\left(\frac{0}{10000\frac{0}{5}}\right) = \cos(0) = 1$$

$$PE_{(0,2)} = sin\left(\frac{0}{10000\frac{0}{5}}\right) = \sin(0) = 0$$

$$PE_{(0,3)} = cos\left(\frac{0}{10000\frac{0}{5}}\right) = \cos(0) = 1$$

$$PE_{(0,4)} = sin\left(\frac{4}{10000\frac{0}{5}}\right) = \sin(0) = 0$$

Jadi, untuk pos 0, diperoleh:

$$PE(0) = [0 \quad 1 \quad 0 \quad 1 \quad 0]$$

Dengan menggunakan cara yang sama diperoleh matriks PE_{pos} untuk $pos = 1, 2, 3, 4,$ dan 5 sebagai berikut.

$$PE(1) = [0{,}8415 \quad 0{,}5403 \quad 0{,}0251 \quad 0{,}9997 \quad 0{,}0006]$$

$$PE(2) = [0{,}9093 \quad -0{,}4161 \quad 0{,}0502 \quad 0{,}9987 \quad 0{,}0013]$$

$$PE(3) = [0{,}1411 \quad -0{,}9900 \quad 0{,}0753 \quad 0{,}9972 \quad 0{,}0019]$$

$$PE(4) = [-0{,}7568 \quad -0{,}6536 \quad 0{,}1003 \quad 0{,}9950 \quad 0{,}0025]$$

$$PE(5) = [-0{,}9589 \quad 0{,}2837 \quad 0{,}1253 \quad 0{,}9921 \quad 0{,}0032]$$

Dilakukan penggabungan matriks PE_{pos} sehingga diperoleh matriks sebagai berikut.

$$PE_{pos} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 0{,}8415 & 0{,}5403 & 0{,}0251 & 0{,}9997 & 0{,}0006 \\ 0{,}9093 & -0{,}4161 & 0{,}0502 & 0{,}9987 & 0{,}0013 \\ 0{,}1411 & -0{,}9900 & 0{,}0753 & 0{,}9972 & 0{,}0019 \\ -0{,}7568 & -0{,}6536 & 0{,}1003 & 0{,}9950 & 0{,}0025 \\ -0{,}9589 & 0{,}2837 & 0{,}1253 & 0{,}9921 & 0{,}0032 \end{bmatrix}$$

3. Melakukan *segment embedding*, karena proses pada augmentasi akan melibatkan banyak kalimat. Untuk melakukan *segment embedding*, diperlukan bobot nilai dari beberapa contoh kalimat. Dimisalkan ada 6 kalimat dengan nilai sebagai berikut.

$$W_{pos} = \begin{bmatrix} 0{,}12 & -0{,}45 & 0{,}88 & -0{,}23 & 0{,}55 \\ -0{,}33 & 0{,}67 & -0{,}12 & 0{,}90 & -0{,}76 \\ 0{,}78 & -0{,}11 & 0{,}45 & -0{,}34 & 0{,}22 \\ -0{,}56 & 0{,}81 & -0{,}67 & 0{,}13 & 0{,}44 \\ 0{,}23 & -0{,}39 & 0{,}79 & -0{,}92 & 0{,}31 \\ -0{,}14 & 0{,}58 & -34 & 0{,}67 & -0{,}25 \end{bmatrix}$$

Kalimat pada perhitungan manual BERT dianggap sebagai kalimat pada baris pertama, sehingga diperoleh segment ID = 0, maka vektor untuk *segment embedding* adalah sebagai berikut.

$$\hat{S}_{pos} = [0{,}12 \quad -0{,}45 \quad 0{,}88 \quad -0{,}23 \quad 0{,}55]$$

Pada kalimat *input*, kata yang akan diprediksi adalah *"dumb-brained"* ,*"hummans"*dan *"shouting"*, maka dilakukan *masking* pada *token* ke 2, 3, dan 6. Sehingga, diperoleh *embedding* akhir untuk kata yang di *masking* yaitu sebagai berikut.

$$Z_2 = PE(1) + \hat{S}_{pos}$$

$$Z_2 = [0{,}8415 + 0{,}12 \quad 0{,}5403 - 0{,}45 \quad 0{,}0251 + 0{,}88 \quad 0{,}9997 - 0{,}23 \quad 0{,}0006 - 0{,}55]$$

$$Z_2 = [0{,}9615 \quad 0{,}0903 \quad 0{,}9051 \quad 0{,}7697 \quad 0{,}5506]$$

$$Z_3 = PE(2) + S_{pos}$$

$$Z_3 = [0{,}9093 + 0{,}12 \quad -0{,}4161 - 0{,}45 \quad 0{,}0502 + 0{,}88 \quad 0{,}9987 - 0{,}23 \quad 0{,}0013 - 0{,}55]$$

$$Z_3 = [1{,}0293 \quad -0{,}8661 \quad 0{,}9302 \quad 0{,}7687 \quad -0{,}5487]$$

$$Z_6 = PE(5) + S_{pos}$$

$$Z_6 = [-0{,}9589 + 0{,}12 \quad 0{,}2837 - 0{,}45 \quad 0{,}1253 + 0{,}88 \quad 0{,}9921 - 0{,}23 \quad 0{,}0032 - 0{,}55]$$

$$Z_6 = [-0{,}8389 \quad -0{,}1663 \quad 1{,}0053 \quad 0{,}7621 \quad -0{,}5468]$$

Tahap 3. ***Self attention***

$$W_Q = \begin{bmatrix} 0{,}3 & -0{,}1 \\ -0{,}2 & 0{,}4 \\ 0{,}5 & -0{,}3 \\ 0{,}1 & 0{,}2 \\ -0{,}4 & 0{,}6 \end{bmatrix}, W_K = \begin{bmatrix} 0{,}2 & 0{,}4 \\ 0{,}5 & -0{,}3 \\ -0{,}6 & 0{,}2 \\ 0{,}3 & 0{,}7 \\ 0{,}1 & -0{,}5 \end{bmatrix}, W_V = \begin{bmatrix} -0{,}5 & 0{,}6 \\ 0{,}3 & -0{,}1 \\ 0{,}4 & 0{,}8 \\ -0{,}2 & 0{,}5 \\ 0{,}7 & -0{,}3 \end{bmatrix}$$

Melakukan perhitungan nilai untuk *query*, *key*, dan *value* dengan Persamaan (2.5), Persamaan (2.6), Persamaan (2.7), dan Persamaan (2.8).

$$Q_2 = Z_2 W_Q$$

$$Q_2 = [0{,}9615 \quad 0{,}0903 \quad 0{,}9051 \quad 0{,}7697 \quad 0{,}5506] \begin{bmatrix} 0{,}3 & -0{,}1 \\ -0{,}2 & 0{,}4 \\ 0{,}5 & -0{,}3 \\ 0{,}1 & 0{,}2 \\ -0{,}4 & 0{,}6 \end{bmatrix}$$

$$Q_2 = [0{,}57967 \quad 0{,}15274]$$

$$K_2 = Z_2 W_K$$

$$K_2 = [0{,}9615 \quad 0{,}0903 \quad 0{,}9051 \quad 0{,}7697 \quad 0{,}5506] \begin{bmatrix} 0{,}2 & 0{,}4 \\ 0{,}5 & -0{,}3 \\ -0{,}6 & 0{,}2 \\ 0{,}3 & 0{,}7 \\ 0{,}1 & -0{,}5 \end{bmatrix}$$

$$K_2 = [-0{,}01964 \quad 0{,}80202]$$

$$V_2 = Z_2 W_V$$

$$V_2 = [0{,}9615 \quad 0{,}0903 \quad 0{,}9051 \quad 0{,}7697 \quad 0{,}5506] \begin{bmatrix} -0{,}5 & 0{,}6 \\ 0{,}3 & -0{,}1 \\ 0{,}4 & 0{,}8 \\ -0{,}2 & 0{,}5 \\ 0{,}7 & -0{,}3 \end{bmatrix}$$

$$V_2 = [0{,}13986 \quad 1{,}51162]$$

Dengan perhitungan yang sama untuk *token* "*hummans*"dan "*shouting*", diperoleh diperoleh nilai $Q, K, dan\ V$ sebagai berikut.

Selanjutnya dihitung nilai *self-attention score* sebagai berikut.

$$Attention_2 = \frac{QK^T}{\sqrt{d}}$$

$$Attention_2 = \frac{[0{,}57967 \quad 0{,}15274][-0{,}01964 \quad 0{,}80202]^T}{\sqrt{2}}$$

$$Attention_2 = \frac{[0{,}57967 \quad 0{,}15274] \begin{bmatrix} -0{,}01964 \\ 0{,}80202 \end{bmatrix}}{\sqrt{2}}$$

$$Attention_2 = 0{,}0785$$

Dengan perhitungan yang sama untuk *token* "*hummans*"dan "*shouting*", pada indeks ketiga dan keenam sehingga diperoleh diperoleh nilai sebagai berikut.

$$Attention_3 = -1{,}6034$$

$$Attention_6 = -0{,}0635$$

Selanjutnya, dihitung menggunakan fungsi *softmax*.

$$softmax(S_i) = \frac{e^{S_i}}{\sum_{j=1}^{n} e^{S_j}}$$

$$softmax(0{,}0785) = \frac{e^{0{,}0785}}{e^{0{,}0785} + e^{-1{,}6034} + e^{-0{,}0635}} = \frac{1{,}0817}{2{,}2213} = 0{,}4869$$

$$softmax(S_3) = \frac{e^{-1,6034}}{e^{0,0785} + e^{-1,6034} + e^{-0,0635}} = \frac{0,2012}{2,2213} = 0,0906$$

$$softmax(S_6) = \frac{e^{-0,0635}}{e^{0,0785} + e^{-1,6034} + e^{-0,0635}} = \frac{0,9384}{2,2213} = 0,4225$$

Melakukan perkalian dengan matriks *value*

$$S_2 = 0,4869[0,13986 \quad 1,51162] = [0,0681 \quad 0,73601]$$

$$S_3 = 0,0906[0,13986 \quad 1,51162] = [0,0127 \quad 0,13695]$$

$$S_6 = 0,4225[0,13986 \quad 1,51162] = [0,0591 \quad 0,63866]$$

Tahap 3. ***Feed Forward Network***

Digunakan bobot dan bias untuk kata kedua dan keenam sebagai berikut.

$$W_1 = \begin{bmatrix} 0,6 & -0,4 \\ 0,3 & 0,8 \end{bmatrix}, B_1 = [0,1]$$

$$W_2 = \begin{bmatrix} 0,5 & 0,7 \\ -0,6 & 0,4 \end{bmatrix}, B_2 = \begin{bmatrix} 0,2 \\ -0,1 \end{bmatrix}$$

Dihitung *feet forward network* untuk kata "*dumb-brained*" sebagai berikut.

$$H = ReLU(W_1 S_2 + B_1)$$

$$H = ReLU\left(\begin{bmatrix} 0,6 & -0,4 \\ 0,3 & 0,8 \end{bmatrix}\begin{bmatrix} 0,0681 \\ 0,7360 \end{bmatrix} + [0,1]\right)$$

$$H = ReLU\left(\begin{bmatrix} -0,15354 \\ 0,70923 \end{bmatrix}\right)$$

$$H = \begin{bmatrix} 0 \\ 0,70923 \end{bmatrix}$$

Output akhir:

$$O_2 = \begin{bmatrix} 0,5 & 0,7 \\ -0,6 & 0,4 \end{bmatrix}\begin{bmatrix} 0 \\ 0,70923 \end{bmatrix} + [0,2]$$

$$O_2 = \begin{bmatrix} 0,69646 \\ 0,48369 \end{bmatrix}$$

Dilakukan langkah yang sama, sehingga diperoleh nilai O_6 sebagai berikut.

$$O_3 = \begin{bmatrix} 0{,}37577 \\ 0{,}25364 \end{bmatrix}$$

$$O_6 = \begin{bmatrix} 0{,}24034 \\ 0{,}40305 \end{bmatrix}$$

Tahap 4. *Softmax*

$$softmax(O_i) = \frac{e^{O_i}}{\sum_{j=1}^{n} e^{O_j}}$$

Memprediksi kata yang cocok untuk menggantikan kata yang di *mask* yaitu *dumb-brained.* Digunakan 2 kata yang akan menggantikan kata *dumb-brained* yaitu *ignore* dan *unwise* berdasarkan korpus BERT.

$$softmax(0{,}69646) = \frac{e^{0{,}69646}}{e^{0{,}69646} + e^{0{,}48369}} = \frac{2{,}0066}{3{,}6286} = 0{,}6078$$

$$softmax(0{,}48369) = \frac{e^{0{,}48369}}{e^{0{,}69646} + e^{0{,}48369}} = \frac{1{,}6220}{3{,}6286} = 0{,}3922$$

Memprediksi kata yang cocok untuk menggantikan kata yang di *mask* yaitu *hummans.* Digunakan 2 kata yang akan menggantikan kata *hummans* yaitu *people* dan *individuals* berdasarkan korpus BERT.

$$softmax(0{,}37577) = \frac{e^{0{,}37577}}{e^{0{,}37577} + e^{0{,}25364}} = \frac{1{,}45611}{2{,}57740} = 0{,}56$$

$$softmax(0{,}25364) = \frac{e^{0{,}25364}}{e^{0{,}37577} + e^{0{,}25364}} = \frac{1{,}28870}{2{,}57740} = 0{,}44$$

Memprediksi kata yang cocok untuk menggantikan kata yang di *mask* yaitu *shouting.* Digunakan 2 kata yang akan menggantikan kata *shouting* yaitu *yelling* dan *screaming* berdasarkan korpus BERT.

$$softmax(0{,}24034) = \frac{e^{0{,}24034}}{e^{0{,}24034} + e^{0{,}40305}} = \frac{1{,}2716}{2{,}7679} = 0{,}45941$$

$$softmax(0{,}40305) = \frac{e^{0{,}40305}}{e^{0{,}24034} + e^{0{,}40305}} = \frac{1{,}4963}{2{,}76798} = 0{,}54057$$

Tahap 5. *Argmax*

Hasil dari fungsi aktivasi memiliki vektor probabilitas sebagai berikut:

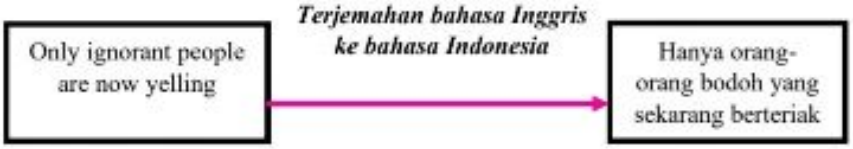
$$S_2 = \begin{bmatrix} 0{,}6078 \\ 0{,}3922 \end{bmatrix}, S_3 = \begin{bmatrix} 0{,}56 \\ 0{,}44 \end{bmatrix} \; dan \; S_6 = \begin{bmatrix} 0{,}6030 \\ 0{,}3977 \end{bmatrix}$$

Diperoleh nilai terbesar untuk S_2 yaitu 0,6078 pada indeks pertama, S_3 yaitu 0,56 pada indeks pertama dan S_6 yaitu 0,6030 pada indeks pertama sehingga, diperoleh kata baru untuk pengganti kata yang di *mask*, yaitu *ignorant, people* dan *yelling*. Kalimat baru hasil augmentasi BERT adalah *only ignorant people are now yelling*. Hasil penerapan *back translation* ke teknik BERT dapat dilihat pada Gambar 4.2.

Gambar 4.2 Hasil penerapan augmentasi BERT

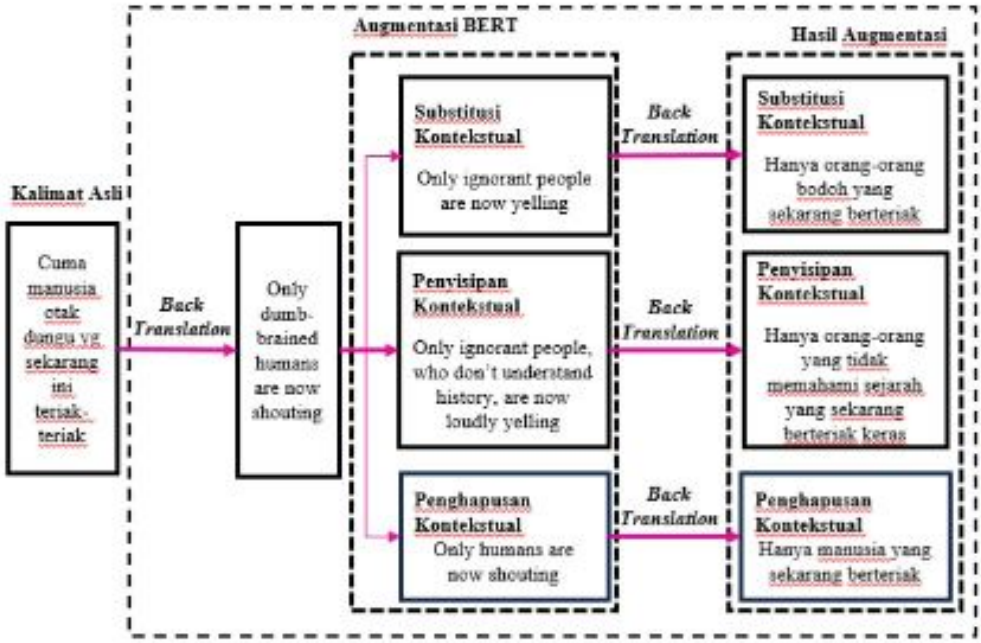
3. Menerapkan *back translation* untuk terjemah ke bahasa Indonesia

   Menerjemahkan kembali data *tweet* ujaran kebencian hasil augmentasi BERT ke bahasa Indonesia. Hasil penerapan *back translation* untuk menerjemahkan bahasa Inggris dari hasil teknik BERT ke bahasa Indoensia dapat dilihat pada Gambar 4.3.



Gambar 4.3 Hasil penerjemahan bahasa Inggris ke bahasa Indonesia

Proses augmentasi BERT melibatkan 3 teknik yaitu substitusi kontekstual, penyisipan kontekstual, dan penghapusan kontekstual. Ilustrasi augmentasi data pada data ujaran kebencian *tweet* bahasa Indonesia dapat dilihat pada Gambar 4.4.



Gambar 4.4. Hasil augmentasi data menggunakan *back translation* dan augmentasi BERT

Data baru hasil augmentasi berjumlah 39.507 data menggunakan augmentasi *back translation* dan augmentasi BERT yang terdiri dari substitusi kontekstual, penyisipan kontekstual, serta penghapusan kontekstual. Pada penelitian ini digunakan 52.676 data yang terdiri dari data asli dan data hasil augmentasi.

4.3 *Text Preprocessing*

Penerapan *text preprocessing* bertujuan untuk memperbaiki kualitas teks dengan menghilangkan *noise* dan memilih kata-kata yang diprioritaskan untuk diolah agar model bekerja dengan optimal dengan tahapan sebagai berikut:

1. Melakukan *input* data mengguanakan *dataset* ujaran kebencian *tweet*.

2. Menerapkan *case folding* pada *dataset* ujaran kebencian *tweet* sehingga huruf pada data menjadi seragam. Dalam tahap ini, *case folding* yang digunakan adalah *lowercase* atau menyamakan huruf menjadi huruf kecil.
3. Menerapkan *remove punctuation* untuk menghapus tanda baca pada data.
4. Menerapkan *slang word* untuk mengubah kata yang disingkat menjadi kata utuh dengan bantuan kamus alay.
5. Menerapkan *stopword removal* untuk menghapus kata yang sering muncul tetapi tidak memiliki informasi penting.
6. Menerapkan *stemming* untuk menghapus kata yang memiliki imbuhan menjadi kata dasar.

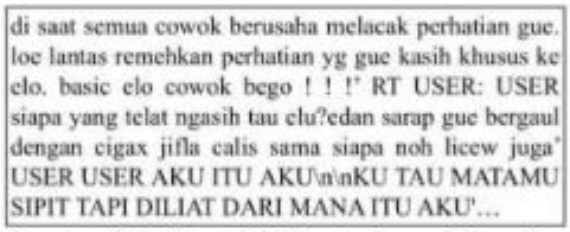
Adapun hasil dari *text preprocessing* menggunakan seluruh langkah-langkah dapat dilihat pada Gambar 4.5.

Gambar 4.5. Penerapan *text preprocessing* pada *dataset* ujaran kebencian *tweet*

Berdasarkan Gambar 4.5, dilakukan *text preprocessing* pada kalimat asli menggunakan *case folding, remove punctuation, slang word, stopeord removal,* dan *stemming* dengan tujuan agar teks lebih mudah dipahami oleh komputer dalam melakukan klasifikasi teks.

4.4 Tokenisasi

Tokenisasi adalah proses mengubah setiap kata dalam kalimat menjadi angka untuk membantu komputer memahami dan memproses informasi. Contoh informasi ujaran kebencian dapat dilihat pada Gambar 4.6.


di saat semua cowok berusaha melacak perhatian gue. loe lantas remehkan perhatian yg gue kasih khusus ke elo. basic elo cowok bego ! ! !' RT USER: USER siapa yang telat ngasih tau elu?edan sarap gue bergaul dengan cigax jifla calis sama siapa noh licew juga' USER USER AKU ITU AKU\n\nKU TAU MATAMU SIPIT TAPI DILIAT DARI MANA ITU AKU'...

Gambar 4.6. Contoh kalimat ujaran kebencian

Tahap pertama dalam tokenisasi yaitu memberikan nomor unik pada masing-masing kata berdasarkan urutan kemunculan yang disebut dengan *vocabulary building*. Hasil *vocabulary building* dapat dilihat pada Tabel 4.4.

Tabel 4.4. *Vocabulary building*

Kata	*Token*
di	1
saat	2
⋮	⋮
ketika	70
orang	71
⋮	⋮
hanya	494
⋮	⋮
4x	566
sekarang	567
tapi	904
bodoh	905
⋮	⋮
hapus	5811
⋮	⋮
teriak	5850

Berdasarkan Tabel 4.2, *vocabulary building* menampilkan kata dengan diberikan *token* unik untuk merepresentasikan urutan kemunculan kata dalam bentuk numerik. Tokenisasi mengubah setiap kata menjadi *token* dan setiap *token* diberikan indeks. Hasil tokenisasi dapat dilihat pada Tabel 4.5.

Tabel 4.5. Tokenisasi kata

Kata	*Token*	Indeks
di	1	6
saat	2	179
⋮	⋮	⋮
ketika	70	153
orang	71	117
⋮	⋮	⋮
hanya	494	52
⋮	⋮	⋮
4x	566	117
sekarang	567	821
⋮	⋮	⋮
tapi	904	78
bodoh	905	337
⋮	⋮	⋮
hapus	5811	66
⋮	⋮	⋮
teriak	5850	7

Berdasarkan Tabel 4.3, dapat dilihat nilai tokenisasi dari masing-masing kata. Tokenisasi dilakukan dengan mengubah kata menjadi *token* dengan indeks yang ditentukan berdasarkan frekuensi kemunculan dalam data. Kata yang lebih sering muncul mendapat indeks lebih kecil. Proses ini membantu model memahami dan memproses teks secara lebih efisien.

Contoh 1. Tokenisasi

Diberikan suatu *input* teks kalimat yang terdiri dari 6 kata, yaitu.

"hanya orang bodoh sekarang teriak hapus"

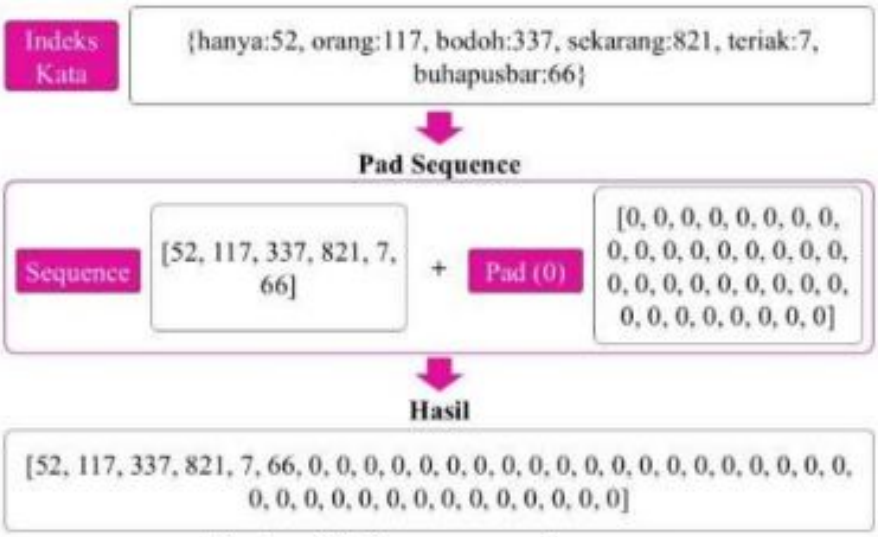
Kalimat diatas dibagi antar kata dan setiap kata diubah menjadi angka. Berdasarkan Tabel 4.2, hasil tokenisasi yaitu.

"hanya":52 "orang":117 "bodoh":337 "sekarang":821 "teriak":7 "hapus":66

Hasil proses tokenisasi dilanjutkan dengan proses *padding sequences*.

4.5 *Pad Sequence*

Pad sequence adalah proses yang bertujuan untuk menyamakan panjang setiap kalimat dalam suatu *dataset* karena setiap kata yang telah diubah dalam bentuk angka memiliki panjang yang bervariasi. *Padding* dilakukan dengan menambahkan nilai 0 pada setiap kalimat dengan panjang kata maksimum yang telah ditentukan. Pada penelitian ini, digunakan panjang kata maksimum yaitu 40. Proses *padding sequence* pada penelitian ini dapat dilihat pada Gambar 4.7.

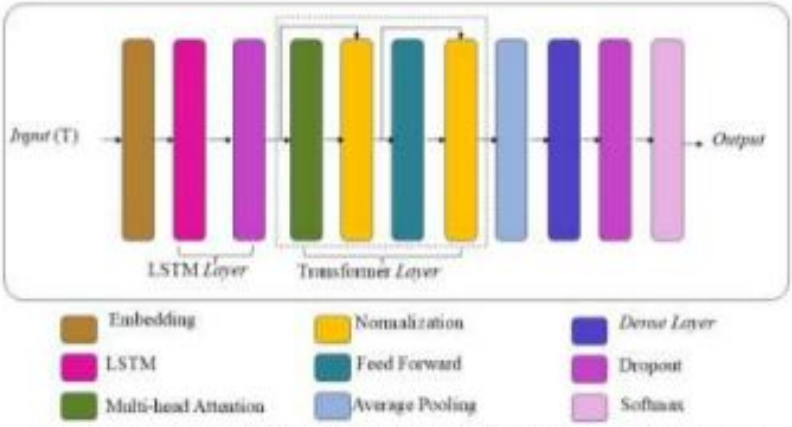


Gambar 4.7. Penerapan *pad sequence*

Berdasarkan Gambar 4.7, proses *pad sequence* dimulai dengan menggunakan nilai indeks dari setiap *token* hasil dari proses tokenisasi. Tahap selanjutnya yaitu menambahkan indeks nol setelah indeks dari *token-token* yang ada. Dalam hal ini, panjang *padding* yang digunakan adalah 40, sehingga ditambahkan indeks nol sebanyak 34.

4.6 Kombinasi Arsitektur LSTM dan *Transformer*

Setelah melewati tahap *preprocessing* dan augmentasi data, tahap selanjutnya adalah penerapan kombinasi arsitektur LSTM dan *Transformer* menggunakan bahasa pemrogaman *Phyton.* Kombinasi arsitektur LSTM dan *Transformer* dapat dilihat pada Gambar 4.8.



Gambar 4.8. Kombinasi arsitektur LSTM dan *Transformer*

Arsitektur LSTM terdiri dari tiga *layer* yaitu *embedding layer,* LSTM *layer,* dan *dropout layer.* Setelah dilakukan perhitungan menggunakan arsitektur LSTM, selanjutnya dilakukan perhitungan menggunakan arsitektur *Transformer*. Arsitektur *Transformer* dimulai dari *self attention, multi-head attention, normalitation layer,* dan *feed forward layer* dalam menangkap hubungan antar kata pada data. Data yang telah diproses menggunakan arsitektur *Transformer* kemudian dioptimalkan menggunakan *average pooling,* fungsi aktivasi, dan *loss function.* Penjelasan langkah-langkah pada kombinarsitektur LSTM dan *Transformer* yakni sebagai berikut:

1. Mengubah kata-kata yang diperoleh dari proses *pad sequence* menjadi bentuk *entry* matriks yang dilakukan pada tahap *embedding layer.*

2. Menerapkan LSTM *layer* dengan *input* nilai yang diperoleh dari proses *embedding layer* menggunakan tiga *gate, gate,* yaitu *forgot gate, input gate,* dan *output gate.*
3. Matriks *input* arsitektur *Transformer* adalah matriks *hidden layer* H_t dari pemrosesan arsitektur LSTM yang akan diproses pada lapisan *self attention.*
4. Hasil *self attention* akan diproses pada *multi-head attention.*
5. Hasil *multi head attenion* dinormalkan menggunakan *normalization layer.*
6. Selanjutnya hasil normalisasi diproses pada tahap *feed forward layer.*
7. Setelah tahap *feed forward layer,* selanjutnya dilakukan *Average Pooling.*
8. Menentukan nilai *loss function.*
9. Melakukan *adam optimizer.*

4.7 Operasi Manual Kombinasi Arsitektur LSTM dan Transformer

Perhitungan manual pada kombinasi arsitektur LSTM dan *Trasnsformer* serta parameter-parameter yang digunakan pada penelitian ini adalah sebagai berikut:

1. *Embedding Layer*

Berdasarkan Gambar 2.3, *embedding layer* merupakan *layer* untuk mengubah data dalam bentuk teks menjadi *entry* matriks. Proses perhitungan *embedding layer* menggunakan Persamaan (2.10).

Contoh 2. *Embedding Layer*

Misalkan diberi suatu *input* kata $\hat{k}$ yang telah diberikan indeks dan terdapat satu matriks W dengan tiap *entry* acak berukuran 6×6 dimana jumlah baris pada matriks W menyesuaikan dengan jumlah *input* kata.

$\hat{k}$

= [hanya":52, "orang ":117, "bodoh":337, "sekarang":821, "teriak":7, "hapus":66]

$$W = \begin{bmatrix} 1{,}3 & -2{,}3 & 0{,}9 & 1{,}5 & 3 & 2{,}1 \\ 0{,}6 & 1{,}7 & -2{,}4 & 2 & 0{,}8 & 0{,}8 \\ 2 & 1{,}1 & 0{,}3 & 2{,}6 & -0{,}7 & 1{,}9 \\ 1{,}9 & 2{,}1 & -1{,}6 & 0{,}9 & 2{,}9 & 2{,}4 \\ 0{,}8 & -2{,}1 & 1{,}5 & 2{,}7 & 1{,}9 & 0{,}7 \\ 2{,}3 & 1{,}8 & 2 & 0{,}6 & -1{,}4 & 0{,}2 \end{bmatrix}$$

Kemudian dilakukan proses *one hot encoding*, yakni mengubah kata yang telah diberikan indeks menjadi sekumpulan angka 0 dan 1 sesuai dengan besaran indeks dari setiap kata. Indeks dari setiap kata diurutkn dari yang terkecil ke yang terbesar sehingga berdasarkan $\hat{k}$, diketahui bahwa "teriak" urutan pertama, "cuma" urutan kedua, "bubar" urutan ketiga, "manusia" urutan keempat, "otak" urutan kelima, dan"dungu" urutan keenam. Pada tahap *one hot encoding*, panjang kata E yang digunakan sebesar 6 karena dalam kalimat terdapat 6 *token*.

$$E_1 = \hat{k}(\text{"hanya"}) = [0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$E_2 = \hat{k}(\text{"orang"}) = [0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0]$$

$$E_3 = \hat{k}(\text{"bodoh"}) = [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0]$$

$$E_4 = \hat{k}(\text{"sekarang"}) = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1]$$

$$E_5 = \hat{k}(\text{"teriak"}) = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]$$

$$E_6 = \hat{k}(\text{"hapus"}) = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0]$$

Setelah masing-masing *token* diubah kedalam bentuk *one hot encoding*, dilakukan perkalian terhadap bobot matriks W. Proses perkalian terhadap matriks bobot menggunakan Persamaan (2.10).

$$X_i = E_i W \qquad ; i = 1,2,3,4,5,6$$

Untuk kata "hanya"

$$X_1 = E_1 W$$

$$X_1 = [0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0] \begin{bmatrix} 1{,}3 & -2{,}3 & 0{,}9 & 1{,}5 & 3 & 2{,}1 \\ 0{,}6 & 1{,}7 & -2{,}4 & 2 & 0{,}8 & 0{,}8 \\ 2 & 1{,}1 & 0{,}3 & 2{,}6 & -0{,}7 & 1{,}9 \\ 1{,}9 & 2{,}1 & -1{,}6 & 0{,}9 & 2{,}9 & 2{,}4 \\ 0{,}8 & -2{,}1 & 1{,}5 & 2{,}7 & 1{,}9 & 0{,}7 \\ 2{,}3 & 1{,}8 & 2 & 0{,}6 & -1{,}4 & 0{,}2 \end{bmatrix}$$

$$= [0{,}6 \quad 1{,}7 \quad -2{,}4 \quad 2 \quad 0{,}8 \quad 0{,}8]$$

Untuk kata "orang"

$$X_2 = E_2 W$$

$$X_2 = [0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0] \begin{bmatrix} 1{,}3 & -2{,}3 & 0{,}9 & 1{,}5 & 3 & 2{,}1 \\ 0{,}6 & 1{,}7 & -2{,}4 & 2 & 0{,}8 & 0{,}8 \\ 2 & 1{,}1 & 0{,}3 & 2{,}6 & -0{,}7 & 1{,}9 \\ 1{,}9 & 2{,}1 & -1{,}6 & 0{,}9 & 2{,}9 & 2{,}4 \\ 0{,}8 & -2{,}1 & 1{,}5 & 2{,}7 & 1{,}9 & 0{,}7 \\ 2{,}3 & 1{,}8 & 2 & 0{,}6 & -1{,}4 & 0{,}2 \end{bmatrix}$$

$$= [1{,}9 \quad 2{,}1 \quad -1{,}6 \quad 0{,}9 \quad 2{,}9 \quad 2{,}4]$$

Untuk kata "bodoh"

$$X_3 = E_3 W$$

$$X_3 = [0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0] \begin{bmatrix} 1{,}3 & -2{,}3 & 0{,}9 & 1{,}5 & 3 & 2{,}1 \\ 0{,}6 & 1{,}7 & -2{,}4 & 2 & 0{,}8 & 0{,}8 \\ 2 & 1{,}1 & 0{,}3 & 2{,}6 & -0{,}7 & 1{,}9 \\ 1{,}9 & 2{,}1 & -1{,}6 & 0{,}9 & 2{,}9 & 2{,}4 \\ 0{,}8 & -2{,}1 & 1{,}5 & 2{,}7 & 1{,}9 & 0{,}7 \\ 2{,}3 & 1{,}8 & 2 & 0{,}6 & -1{,}4 & 0{,}2 \end{bmatrix}$$

$$= [0{,}8 \quad -2{,}1 \quad 1{,}5 \quad 2{,}7 \quad 1{,}9 \quad 0{,}7]$$

Untuk kata "sekarang"

$$X_4 = E_4 W$$

$$X_4 = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1] \begin{bmatrix} 1,3 & -2,3 & 0,9 & 1,5 & 3 & 2,1 \\ 0,6 & 1,7 & -2,4 & 2 & 0,8 & 0,8 \\ 2 & 1,1 & 0,3 & 2,6 & -0,7 & 1,9 \\ 1,9 & 2,1 & -1,6 & 0,9 & 2,9 & 2,4 \\ 0,8 & -2,1 & 1,5 & 2,7 & 1,9 & 0,7 \\ 2,3 & 1,8 & 2 & 0,6 & -1,4 & 0,2 \end{bmatrix}$$

$$= [2,3 \quad 1,8 \quad 2 \quad 0,6 \quad -1,4 \quad 0,2]$$

Untuk kata "teriak"

$$X_5 = E_5 W$$

$$X_5 = [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \begin{bmatrix} 1,3 & -2,3 & 0,9 & 1,5 & 3 & 2,1 \\ 0,6 & 1,7 & -2,4 & 2 & 0,8 & 0,8 \\ 2 & 1,1 & 0,3 & 2,6 & -0,7 & 1,9 \\ 1,9 & 2,1 & -1,6 & 0,9 & 2,9 & 2,4 \\ 0,8 & -2,1 & 1,5 & 2,7 & 1,9 & 0,7 \\ 2,3 & 1,8 & 2 & 0,6 & -1,4 & 0,2 \end{bmatrix}$$

$$= [1,3 \quad -2,3 \quad 0,9 \quad 1,5 \quad 3 \quad 2,1]$$

Untuk kata "hapus"

$$X_6 = E_6 W$$

$$X_6 = [0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0] \begin{bmatrix} 1,3 & -2,3 & 0,9 & 1,5 & 3 & 2,1 \\ 0,6 & 1,7 & -2,4 & 2 & 0,8 & 0,8 \\ 2 & 1,1 & 0,3 & 2,6 & -0,7 & 1,9 \\ 1,9 & 2,1 & -1,6 & 0,9 & 2,9 & 2,4 \\ 0,8 & -2,1 & 1,5 & 2,7 & 1,9 & 0,7 \\ 2,3 & 1,8 & 2 & 0,6 & -1,4 & 0,2 \end{bmatrix}$$

$$= [2 \quad 1,1 \quad 0,3 \quad 2,6 \quad -0,7 \quad 1,9]$$

Selanjutnya, dilakukan penggabungan entri-entri matriks $X_1, X_2, X_3, X_4, X_5, dan\ X_6$ sehingga diperoleh matriksukuran 6×6. Proses penggabungan entri-entri matriks $X_1, X_2, X_3, X_4, X_5, dan\ X_6$ adalah sebagai berikut:

$$X = \begin{bmatrix} 0,6 & 1,7 & -2,4 & 2,0 & 0,8 & 0,8 \\ 1,9 & 2,1 & -1,6 & 0,9 & 2,9 & 2,4 \\ 0,8 & -2,1 & 1,5 & 2,7 & 1,9 & 0,7 \\ 2,3 & 1,8 & 2,0 & 0,6 & -1,4 & 0,2 \\ 1,3 & -2,3 & 0,9 & 1,5 & 3,0 & 2,1 \\ 2,0 & 1,1 & 0,3 & 2,6 & -0,7 & 1,9 \end{bmatrix}$$

2. LSTM *Layer*

Proses pada LSTM *layer* dilakukan menggunakan Persamaan (2.10), Persamaan (2.11), Persamaan (2.12), Persamaan (2.13), Persamaan (2.14), Persamaan (2.15).

Contoh 3. LSTM *layer*

Misalkan diambil sebuah matriks X enam entri di setiap masing-masing kolom sebagai *input* waktu ke-1, ke-2, ke-3, ke-4, ke-5, dan ke-6 yang diinisialisasi sebagai $\hat{X}_1, \hat{X}_2, \hat{X}_3, \hat{X}_4, \hat{X}_5$, dan $\hat{X}_6$ sebagai berikut:

$$\hat{X}_1 = \begin{bmatrix} 0,6 \\ 1,9 \\ 0,8 \\ 2,3 \\ 1,3 \\ 2 \end{bmatrix}, \hat{X}_2 = \begin{bmatrix} 1,7 \\ 2,1 \\ -2,1 \\ 1,8 \\ -2,3 \\ 1,1 \end{bmatrix}, \hat{X}_3 = \begin{bmatrix} -2,4 \\ -1,6 \\ 1,5 \\ 2 \\ 0,9 \\ 0,3 \end{bmatrix}, \hat{X}_4 = \begin{bmatrix} 2 \\ 0,9 \\ 2,7 \\ 0,6 \\ 1,5 \\ 2,6 \end{bmatrix}, \hat{X}_5 = \begin{bmatrix} 0,8 \\ 2,9 \\ 1,9 \\ -1,4 \\ 3 \\ -0,7 \end{bmatrix}, \hat{X}_6 = \begin{bmatrix} 0,8 \\ 2,4 \\ 0,7 \\ 0,2 \\ 2,1 \\ 1,9 \end{bmatrix}$$

Setelah mendapatkan nilai *input* $\hat{X}_1, \hat{X}_2, \hat{X}_3, \hat{X}_4, \hat{X}_5$, dan $\hat{X}_6$, melakukan perhitungan dengan langkah-langkah sebagai berikut.

a. Melakukan perhitungan *output forget gate* F_1 menggunakan Persamaan (2.10). Untuk menghitung hasil F_1, ditentukan matriks bobot untuk masing-masing *state*, yakni W_{XF} sebagai matriks bobot *input state* dan W_{HF} sebagai matriks bobot *hidden state*. Nilai untuk setiap entri matriks W_{XF} dan W_{HF} dibangkitkan secara acak oleh komputer dimana jumlah kolom pada matriks bobot disesuaikan dengan jumlah baris dari matriks *input*, seperti dibawah ini:

$$W_{XF} = \begin{bmatrix} 1 & 0 & 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & -1 & 0 & 0 \\ 1 & 0 & 1 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix}$$

$$W_{HF} = \begin{bmatrix} 0 & 1 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Kemudian, ditentukan bias B_F dengan nilai seluruh entrinya diinisialisasi 0,1 sebagai berikut:

$$B_F = \begin{bmatrix} 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \end{bmatrix}$$

Jika $t = 1$, maka $H_{t-1} = H_0$ dan $C_{t-1} = C_0$ sehingga H_0 dan C_0 diinisialisasikan sebagai matriks nol karena tidak terdapat langkah yang terjadi pada waktu ke-0, yakni:

$$H_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{dan } C_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Setelah itu hitung *output* pada *forgot gate* menggunakan Persamaan (2.10).

$$F_1 = sigmoid\left(W_{XF}\, \tilde{X}_1 + W_{HF}\, H_{1-1} + B_F\right)$$

$$= sigmoid\left(\begin{bmatrix} 1 & 0 & 0 & 1 & -1 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & 1 & -1 & -1 & 0 & 0 \\ 1 & 0 & 1 & 0 & -1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \end{bmatrix} \begin{bmatrix} 0{,}6 \\ 1{,}9 \\ 0{,}8 \\ 2{,}3 \\ 1{,}3 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & -1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \end{bmatrix} \right)$$

$$= sigmoid\left(\begin{bmatrix}1,6\\0,4\\-1,2\\2,1\\1,9\\0,3\end{bmatrix}+\begin{bmatrix}0\\0\\0\\0\\0\\0\end{bmatrix}+\begin{bmatrix}0,1\\0,1\\0,1\\0,1\\0,1\\0,1\end{bmatrix}\right)$$

$$= sigmoid\left(\begin{bmatrix}1,7\\0,5\\-1,1\\2,2\\2\\0,4\end{bmatrix}\right)$$

Setiap *entry* matriks disubstitusikan ke dalam fungsi aktivasi *sigmoid.* Diketahui $f_{11} = 1,7$, $f_{21} = 0,5$, $f_{31} = -1,1$, $f_{41} = 2,2$, $f_{51} = 2$, dan $f_{61} = 0,4$.

$$\hat{b}_{11} = sigmoid(f_{11}) = \frac{1}{1+e^{-f_{11}}} = \frac{1}{1+e^{-(1,7)}} = \frac{1}{1+e^{-1,7}} = \frac{1}{1,183}$$

$$= 0,846$$

$$\hat{b}_{21} = sigmoid(f_{21}) = \frac{1}{1+e^{-f_{21}}} = \frac{1}{1+e^{-(0,5)}} = \frac{1}{1+e^{-0,5}} = \frac{1}{1,607}$$

$$= 0,622$$

$$\hat{b}_{31} = sigmoid(f_{31}) = \frac{1}{1+e^{-f_{31}}} = \frac{1}{1+e^{-(-1,1)}} = \frac{1}{1+e^{1,1}} = \frac{1}{4,004}$$

$$= 0,250$$

$$\hat{b}_{41} = sigmoid(f_{41}) = \frac{1}{1+e^{-f_{41}}} = \frac{1}{1+e^{-(2,2)}} = \frac{1}{1+e^{-2,2}} = \frac{1}{1,111}$$

$$= 0,9$$

$$\hat{b}_{51} = sigmoid(f_{51}) = \frac{1}{1+e^{-f_{51}}} = \frac{1}{1+e^{-(2)}} = \frac{1}{1+e^{-2}} = \frac{1}{1,135}$$

$$= 0,881$$

$$\hat{b}_{61} = sigmoid(f_{61}) = \frac{1}{1+e^{-f_{61}}} = \frac{1}{1+e^{-(0,4)}} = \frac{1}{1+e^{-0,4}} = \frac{1}{1,67}$$

$$= 0,599$$

Setelah setiap *entry* diterapkan kedalam fungsi aktivasi *sigmoid,* diperoleh hasil sebagai berikut:

$$F_1 = \begin{bmatrix} 0{,}846 \\ 0{,}622 \\ 0{,}250 \\ 0{,}9 \\ 0{,}881 \\ 0{,}599 \end{bmatrix}$$

I.akukan perhitungan yang sama untuk menghitung *forgot gate* (F_t) untuk $t = 1,2,3,4,5,6$ sehingga diperoleh hasil sebagai berikut.

$$F_2 = \begin{bmatrix} 0{,}997 \\ 0{,}450 \\ 0{,}924 \\ 0{,}957 \\ 0{,}378 \\ 0{,}690 \end{bmatrix}, F_3 = \begin{bmatrix} 0{,}231 \\ 0{,}976 \\ 0{,}007 \\ 0{,}182 \\ 0{,}198 \\ 0{,}870 \end{bmatrix}, F_4 = \begin{bmatrix} 0{,}769 \\ 0{,}450 \\ 0{,}091 \\ 0{,}997 \\ 0{,}973 \\ 0{,}130 \end{bmatrix}, F_5 = \begin{bmatrix} 0{,}029 \\ 0{,}015 \\ 0{,}924 \\ 0{,}289 \\ 0{,}980 \\ 0{,}354 \end{bmatrix}, F_6 = \begin{bmatrix} 0{,}269 \\ 0{,}109 \\ 0{,}832 \\ 0{,}802 \\ 0{,}953 \\ 0{,}168 \end{bmatrix}$$

b. Menghitung *output* untuk *input gate* I_t menggunakan Persamaan (2.11). Untuk menghitung hasil I_t, ditentukan matriks bobot untuk masing-masing *state,* yakni W_{XI} sebagai matriks bobot *input state* dan W_{HI} sebagai matriks bobot *hidden state.*Nilai untuk setiap entri matriks W_{XI} dan W_{HI} dibangkitkan secara acak oleh komputer dengan jumlah kolom yang sama dengan jumlah baris pada matriks *input,* seperti dibawah ini:

$$W_{XI} = \begin{bmatrix} 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix}$$

$$W_{HI} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 \end{bmatrix}$$

Kemudian, ditentukan bias B_I, H_0, dan C_0 yang telah diinisialisasi sebelumnya. Setelah itu, hitung *output* pada *input gate* I_t menggunakan Persamaan (2.11).

$$I_1 = sigmoid\left(W_{XI}\tilde{X}_1 + W_{HI}H_{1-1} + B_I\right)$$

$$= sigmoid\left(\begin{bmatrix} 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & -1 & 0 & 0 & -1 \end{bmatrix}\begin{bmatrix} 0{,}6 \\ 1{,}9 \\ 0{,}8 \\ 2{,}3 \\ 1{,}3 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 0 & -1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \end{bmatrix}\right)$$

$$= sigmoid\left(\begin{bmatrix} -0{,}6 \\ -1{,}5 \\ 2{,}9 \\ 0{,}6 \\ 1{,}4 \\ -2{,}8 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \end{bmatrix}\right)$$

$$= sigmoid\begin{bmatrix} -0{,}5 \\ -1{,}4 \\ 3 \\ 0{,}7 \\ 1{,}5 \\ -2{,}7 \end{bmatrix}$$

Setiap *entry* matriks tersebut didistribusikan ke dalam fungsi aktivasi *sigmoid* 1dengan proses yang sama seperti langkah (a). Setelah setiap entri diterapkan kedalam fungsi aktivasi *sigmoid*, diperoleh hasil sebagai berikut:

$$I_1 = \begin{bmatrix} 0{,}378 \\ 0{,}198 \\ 0{,}953 \\ 0{,}668 \\ 0{,}818 \\ 0{,}063 \end{bmatrix}$$

Lakukan perhitungan yang sama untuk menghitung *input gate* (I_t) untuk $t = 1,2,3,4,5,6$ sehingga diperoleh hasil sebagai berikut.

$$I_2 = \begin{bmatrix} 0{,}013 \\ 0{,}022 \\ 0{,}973 \\ 0{,}989 \\ 0{,}378 \\ 0{,}750 \end{bmatrix}, I_3 = \begin{bmatrix} 0{,}931 \\ 0{,}401 \\ 0{,}426 \\ 0{,}083 \\ 0{,}937 \\ 0{,}168 \end{bmatrix}, I_4 = \begin{bmatrix} 0{,}668 \\ 0{,}9 \\ 0{,}937 \\ 0{,}378 \\ 0{,}668 \\ 0{,}005 \end{bmatrix}, I_5 = \begin{bmatrix} 0{,}55 \\ 0{,}968 \\ 0{,}378 \\ 0{,}5 \\ 0{,}198 \\ 0{,}25 \end{bmatrix}, I_6 = \begin{bmatrix} 0{,}45 \\ 0{,}646 \\ 0{,}750 \\ 0{,}599 \\ 0{,}769 \\ 0{,}076 \end{bmatrix}$$

c. Menghitung *output* $\hat{C}_t$ untuk membuat satu kandidat dengan nilai baru yang akan ditambahkan ke dalam *cell state* C_t menggunakan Persamaan (2.12). Untuk menghitung hasil $\hat{C}_t$, ditentukan terlebih dahulu matriks bobot untuk masing-masing *state,* yakni $W_{X\hat{C}}$ sebagai matriks bobot *input state* dan $W_{H\hat{C}}$ sebagai matriks bobot *hidden state.* Nilai untuk setiap entri matriks $W_{X\hat{C}}$ dan $W_{H\hat{C}}$ dibangkitkan secara acak oleh komputer dimana jumlah kolomnya sama dengan jumlah baris pada matriks *input.*

$$W_{X\hat{C}} = \begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

$$W_{H\hat{C}} = \begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix}$$

Kemudian, ditentukan bias $B_{\hat{C}}$, H_0, dan C_0 yang telah diinisialisasi sebelumnya. Setelah itu, hitung *output* $\hat{C}_t$ menggunakan Persamaan (2.12).

$$\hat{C}_1 = tanh(W_{X\hat{C}}\, \hat{X}_1 + W_{H\hat{C}}\, H_{1-1} + B_{\hat{C}})$$

$$= tanh\left(\begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0{,}6 \\ 1{,}9 \\ 0{,}8 \\ 2{,}3 \\ 1{,}3 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \end{bmatrix}\right)$$

$$= tanh\left(\begin{bmatrix} 0{,}5 \\ 0{,}6 \\ -1{,}2 \\ -1{,}4 \\ 0{,}4 \\ -1{,}7 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \end{bmatrix}\right)$$

$$= \tanh\begin{bmatrix} 0,6 \\ 0,7 \\ -1,1 \\ -1,3 \\ 0,5 \\ -1,6 \end{bmatrix}$$

Setiap *entry* matriks tersebut disubstitusikan ke dalam fungsi aktivasi *tanh*. Diketahui $\hat{C}_{11} = 0,6$, $\hat{C}_{21} = 0,7$, $\hat{C}_{31} = -1,1$, $\hat{C}_{41} = -1,3$, $\hat{C}_{51} = 0,5$, dan $\hat{C}_{61} = -1,6$.

$$\hat{h}_{11} = \tanh(\hat{C}_{11}) = \frac{e^{\hat{C}_{11}} - e^{\hat{C}_{11}}}{e^{\hat{C}_{11}} + e^{\hat{C}_{11}}} = \frac{e^{0,6} - e^{0,6}}{e^{0,6} + e^{0,6}} = \frac{1,273}{2.371} = 0,537$$

$$\hat{h}_{21} = \tanh(\hat{C}_{21}) = \frac{e^{\hat{C}_{21}} - e^{\hat{C}_{21}}}{e^{\hat{C}_{21}} + e^{\hat{C}_{21}}} = \frac{e^{0,7} - e^{0,7}}{e^{0,7} + e^{0,7}} = \frac{1,517}{2,51} = 0,604$$

$$\hat{h}_{31} = \tanh(\hat{C}_{31}) = \frac{e^{\hat{C}_{31}} - e^{\hat{C}_{31}}}{e^{\hat{C}_{31}} + e^{\hat{C}_{31}}} = \frac{e^{-1,1} - e^{-1,1}}{e^{-1,1} + e^{-1,1}} = \frac{-2,671}{3.337} = -0,8$$

$$\hat{h}_{41} = \tanh(\hat{C}_{41}) = \frac{e^{\hat{C}_{41}} - e^{\hat{C}_{41}}}{e^{\hat{C}_{41}} + e^{\hat{C}_{41}}} = \frac{e^{-1,3} - e^{-1,3}}{e^{-1,3} + e^{-1,3}} = \frac{-3,397}{3.942} = -0,862$$

$$\hat{h}_{51} = \tanh(\hat{C}_{51}) = \frac{e^{\hat{C}_{51}} - e^{\hat{C}_{51}}}{e^{\hat{C}_{51}} + e^{\hat{C}_{51}}} = \frac{e^{0,5} - e^{0,5}}{e^{0,5} + e^{0,5}} = \frac{1,042}{2,255} = -0,462$$

$$\hat{h}_{61} = \tanh(\hat{C}_{61}) = \frac{e^{\hat{C}_{61}} - e^{\hat{C}_{61}}}{e^{\hat{C}_{61}} + e^{\hat{C}_{61}}} = \frac{e^{-1,6} - e^{-1,6}}{e^{-1,6} + e^{-1,6}} = \frac{-4,751}{5,155} = -0,922$$

Setelah setiap *entry* diterapkan kedalam fungsi aktivasi *tanh*, diperoleh hasil *output* $\hat{C}_1$ sebagai berikut:

$$\hat{C}_1 = \begin{bmatrix} 0,537 \\ 0,604 \\ -0,8 \\ -0,862 \\ 0,462 \\ -0,922 \end{bmatrix}$$

Lakukan perhitungan yang sama untuk menghitung *output* ($\hat{C}_t$) untuk $t = 1,2,3,4,5,6$ sehingga diperoleh hasil sebagai berikut.

$$\hat{C}_2 = \begin{bmatrix} -0{,}1 \\ 1 \\ -0{,}996 \\ 0{,}604 \\ -0{,}197 \\ 0 \end{bmatrix}, \hat{C}_3 = \begin{bmatrix} -0{,}462 \\ -0{,}984 \\ 0{,}885 \\ -0{,}987 \\ 0{,}999 \\ -1 \end{bmatrix}, \hat{C}_4 = \begin{bmatrix} -0{,}8 \\ -0{,}462 \\ 0{,}197 \\ -0{,}462 \\ -0{,}197 \\ 0{,}905 \end{bmatrix},$$

$$\hat{C}_5 = \begin{bmatrix} 0{,}834 \\ 0 \\ 0{,}991 \\ 0{,}922 \\ -1 \\ 0{,}980 \end{bmatrix}, \hat{C}_6 = \begin{bmatrix} 0{,}905 \\ 0{,}380 \\ -0{,}8 \\ -0{,}762 \\ -0{,}970 \\ 0{,}604 \end{bmatrix}$$

d. Menghitung matriks *output* untuk *cell state* baru C_1 untuk memperbarui matriks *cell state* lama C_0 dengan menggunakan Persamaan (2.13). Hasil perhitungan *cell state* baru sebagai berikut:

$$C_1 = F_1 \odot C_0 + I_1 \odot \hat{C}_1$$

$$= \begin{bmatrix} 0{,}846 \\ 0{,}622 \\ 0{,}250 \\ 0{,}9 \\ 0{,}881 \\ 0{,}599 \end{bmatrix} \odot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0{,}378 \\ 0{,}198 \\ 0{,}953 \\ 0{,}668 \\ 0{,}818 \\ 0{,}063 \end{bmatrix} \odot \begin{bmatrix} 0{,}537 \\ 0{,}604 \\ -0{,}8 \\ -0{,}862 \\ 0{,}462 \\ -0{,}922 \end{bmatrix}$$

$$= \begin{bmatrix} (0{,}846)(0) \\ (0{,}622)(0) \\ (0{,}250)(0) \\ (0{,}9)(0) \\ (0{,}881)(0) \\ (0{,}599)(0) \end{bmatrix} + \begin{bmatrix} (0{,}378)(0{,}537) \\ (0{,}198)(0{,}604) \\ (0{,}953)(-0{,}8) \\ (0{,}668)(-0{,}862) \\ (0{,}818)(0{,}462) \\ (0{,}063)(-0{,}922) \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} (0{,}203) \\ 0{,}12 \\ -0{,}763 \\ -0{,}576 \\ 0{,}378 \\ -0{,}058 \end{bmatrix}$$

$$= \begin{bmatrix} (0{,}203) \\ 0{,}12 \\ -0{,}763 \\ -0{,}576 \\ 0{,}378 \\ -0{,}058 \end{bmatrix}$$

c. Menghitung *output* untuk *output gate* O_t menggunakan Persamaan (2.14). Untuk menghitung hasil O_t, ditentukan matriks bobot untuk masing-masing *state*, yakni W_{XO} sebagai matriks bobot *input state* dan W_{HO} sebagai matriks bobot *hidden state*. Nilai untuk setiap entri matriks W_{XO} dan W_{HO} dibangkitkan secara acak oleh komputer, seperti berikut:

$$W_{XO} = \begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

$$W_{HO} = \begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix}$$

Kemudian, ditentukan bias O_t, H_0, dan C_0 yang telah diinisialisasi sebelumnya. Setelah itu, hitung *output* O_t menggunakan Persamaan (2.14).

$$O_1 = sigmoid\left(W_{XO}\, X_1 + W_{HO}\, H_{t-1} + B_O\right)$$

$$= sigmoid\left(\begin{bmatrix} 0 & 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 \end{bmatrix}\begin{bmatrix} 0{,}6 \\ 1{,}9 \\ 0{,}8 \\ 2{,}3 \\ 1{,}3 \\ 2 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 \end{bmatrix}\begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \\ 0{,}1 \end{bmatrix}\right)$$

$$= \begin{bmatrix} 0{,}931 \\ 0{,}196 \\ 0{,}690 \\ 0{,}892 \\ 0{,}690 \\ 0{,}832 \end{bmatrix}$$

Lakukan perhitungan yang sama untuk menghitung *output gate* (O_t) untuk $t =$ 1,2,3,4,5,6 sehingga diperoleh hasil sebagai berikut.

$$O_2 = \begin{bmatrix} 0{,}98 \\ 0{,}022 \\ 0{,}971 \\ 0{,}9 \\ 0{,}020 \\ 0{,}982 \end{bmatrix}, O_3 = \begin{bmatrix} 0{,}020 \\ 0{,}401 \\ 0{,}354 \\ 0{,}238 \\ 0{,}968 \\ 0{,}646 \end{bmatrix}, O_4 = \begin{bmatrix} 0{,}953 \\ 0{,}9 \\ 0{,}769 \\ 0{,}729 \\ 0{,}401 \\ 0{,}119 \end{bmatrix}, O_5 = \begin{bmatrix} 0{,}978 \\ 0{,}968 \\ 0{,}027 \\ 0{,}925 \\ 0{,}909 \\ 0{,}039 \end{bmatrix}, O_6 = \begin{bmatrix} 0{,}964 \\ 0{,}646 \\ 0{,}475 \\ 0{,}917 \\ 0{,}802 \\ 0{,}401 \end{bmatrix}$$

f. Menghitung *output* dari *hidden state* H_t dimana $t = 1,2,3,4,5,6$ menggunakan Persamaan (2.15). Hasil perhitungan *output* untuk *hidden state* H_t adalah sebagai berikut:

$$H_1 = O_1 \odot tanh(C_1)$$

Untuk memperoleh hasil dari $tanh(C_1)$ dilakukan perhitungan tiap entri C_1 menggunakan fungsi aktivasi tanh dengan proses yang sama seperti langkah (c) sehingga diperoleh $tanh(C_1) = \begin{bmatrix} 0{,}2 \\ 0{,}119 \\ -0{,}643 \\ -0{,}520 \\ 0{,}361 \\ -0{,}058 \end{bmatrix}$.

$$H_1 = O_1 \odot tanh(C_1)$$

$$= \begin{bmatrix} 0{,}931 \\ 0{,}196 \\ 0{,}690 \\ 0{,}892 \\ 0{,}690 \\ 0{,}832 \end{bmatrix} \odot \begin{bmatrix} 0{,}2 \\ 0{,}119 \\ -0{,}643 \\ -0{,}520 \\ 0{,}361 \\ -0{,}058 \end{bmatrix}$$

$$= \begin{bmatrix} (0{,}931)(0{,}2) \\ (0{,}196)(0{,}119) \\ (0{,}690)(-0{,}643) \\ (0{,}892)(-0{,}520) \\ (0{,}690)(0{,}361) \\ (0{,}832)(-0{,}058) \end{bmatrix}$$

$$= \begin{bmatrix} 0{,}186 \\ 0{,}023 \\ -0{,}443 \\ -0{,}464 \\ 0{,}249 \\ -0{,}048 \end{bmatrix}$$

Lakukan perhitungan yang sama untuk menghitung *hidden state* (H_t) untuk $t = 1,2,3,4,5,6$ sehingga diperoleh hasil sebagai berikut.

$$H_2 = \begin{bmatrix} -0{,}001 \\ 0 \\ -0{,}727 \\ 0{,}482 \\ -0{,}001 \\ 0 \end{bmatrix}, H_3 = \begin{bmatrix} -0{,}008 \\ -0{,}151 \\ 0{,}128 \\ -0{,}02 \\ 0{,}71 \\ -0{,}107 \end{bmatrix}, H_4 = \begin{bmatrix} -0{,}466 \\ -0{,}354 \\ 0{,}141 \\ -0{,}126 \\ -0{,}053 \\ 0{,}001 \end{bmatrix},$$

$$H_5 = \begin{bmatrix} 0{,}419 \\ 0 \\ 0{,}01 \\ 0{,}398 \\ -0{,}177 \\ 0{,}009 \end{bmatrix}, H_6 = \begin{bmatrix} 0{,}373 \\ 0{,}155 \\ -0{,}255 \\ -0{,}392 \\ -0{,}507 \\ 0{,}018 \end{bmatrix}$$

Sehingga diperoleh matriks H_t sebagai berikut.

$$H_t = \begin{bmatrix} 0{,}186 & -0{,}001 & -0{,}008 & -0{,}466 & 0{,}419 & 0{,}373 \\ 0{,}024 & 0 & -0{,}151 & -0{,}354 & 0 & 0{,}155 \\ -0{,}443 & 0{,}727 & 0{,}128 & 0{,}141 & 0{,}010 & -0{,}255 \\ -0{,}464 & 0{,}482 & -0{,}020 & -0{,}126 & 0{,}398 & -0{,}392 \\ 0{,}249 & -0{,}001 & 0{,}710 & -0{,}053 & -0{,}177 & -0{,}507 \\ -0{,}048 & 0 & -0{,}107 & 0{,}001 & 0{,}009 & 0{,}018 \end{bmatrix}$$

3. *Self Attention*

Proses pada *self attention* dilakukan menggunakan Persamaan (2.5), Persamaan (2.6), Persamaan (2.7), dan Persamaan (2.8).

Contoh 4. *Self Attention*

Ambil matriks H_t sebagai 6 *input* matriks baris pada proses *self attention* yang disimpan pada matriks Z_i dengan nilai $i = 1, 2, 3, 4, 5, 6$, sebagai berikut.

$Z_1 = [0{,}186 \quad -0{,}001 \quad -0{,}008 \quad -0{,}466 \quad 0{,}419 \quad 0{,}373]$

$Z_2 = [0{,}024 \quad 0 \quad -0{,}151 \quad -0{,}354 \quad 0 \quad 0{,}155]$

$Z_3 = [-0{,}443 \quad -0{,}727 \quad 0{,}128 \quad 0{,}141 \quad 0{,}01 \quad -0{,}255]$

$Z_4 = [-0{,}464 \quad 0{,}482 \quad -0{,}02 \quad -0{,}126 \quad 0{,}398 \quad -0{,}392]$

$Z_5 = [0{,}249 \quad -0{,}001 \quad 0{,}71 \quad -0{,}053 \quad -0{,}177 \quad -0{,}507]$

$Z_6 = [-0{,}048 \quad 0 \quad -0{,}107 \quad 0{,}001 \quad 0{,}009 \quad 0{,}018]$

Kemudian menginisiasi matriks bobot untuk merepresentasikan *input* sebagai *key* (K_i), *query* (Q_i), dan *value* (V_i). Pada contoh ini, matriks bobot memiliki ukuran matriks bobot 6×6. Matriks bobot untuk *key* (W_K), *query* (W_Q), dan *value* (W_V) dibangkitkan secara acak oleh komputer sebagai berikut.

$$W_K = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

$$W_Q = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$W_V = \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

Selanjutnya menghitung nilai masing-masing bobot untuk menjadi *key* (K_i), *query* (Q_i), dan *value* (V_i) menggunakan dengan Persamaan (2.5), Persamaan (2.6), dan

Persamaan (2.7). Hal ini dilakukan dengan cara melakukan perkalian masing-masing *input* dengan matriks bobot.

Perkalian *input* dengan matriks bobot menjadi *key* (K)

$$K_1 = Z_1 W_k$$

$$= [0{,}186 \quad -0{,}001 \quad -0{,}008 \quad -0{,}466 \quad 0{,}419 \quad 0{,}373] \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

$$= [-0{,}233 \quad 0{,}465 \quad -0{,}381 \quad -0{,}652 \quad 0{,}427 \quad 0{,}374]$$

$$K_2 = Z_2 W_k$$

$$= [0{,}024 \quad 0 \quad -0{,}151 \quad -0{,}354 \quad 0 \quad 0{,}155] \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

$$= [0{,}024 \quad 0{,}355 \quad -0{,}330 \quad -0{,}378 \quad 0{,}151 \quad 0{,}481]$$

$$K_3 = Z_3 W_k$$

$$= [-0{,}443 \quad -0{,}727 \quad 0{,}128 \quad 0{,}141 \quad 0{,}01 \quad -0{,}255] \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

$$= [-0{,}453 \quad -0{,}867 \quad 0{,}383 \quad 0{,}584 \quad -0{,}118 \quad 0{,}255]$$

$$K_4 = Z_4 W_k$$

$$= [-0{,}464 \quad 0{,}482 \quad -0{,}02 \quad -0{,}126 \quad 0{,}398 \quad -0{,}392] \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

$$= [-0{,}862 \quad 0{,}608 \quad 0{,}372 \quad 0{,}338 \quad 0{,}418 \quad 0{,}289]$$

$$K_5 = Z_5 W_k$$

$$= [0{,}249 \quad -0{,}001 \quad 0{,}71 \quad -0{,}053 \quad -0{,}177 \quad -0{,}507] \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

$$= [0{,}426 \quad 0{,}051 \quad 1{,}367 \quad -0{,}302 \quad -0{,}887 \quad 0{,}434]$$

$$K_6 = Z_6 W_k$$

$$= [-0{,}048 \quad 0 \quad -0{,}107 \quad 0{,}001 \quad 0{,}009 \quad 0{,}018] \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 1 & 0 & 0 & -1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

$$= [-0{,}058 \quad 0-0{,}001 \quad -0{,}217 \quad 0{,}049 \quad 0{,}117 \quad -0{,}266]$$

Setelah didapatkan matriks K_i selanjutnya menggabungkan matriks K_i sehingga diperoleh matriks *key* (K) sebagai berikut.

$$K = \begin{bmatrix} -0.233 & 0{,}465 & -0{,}381 & -0{,}652 & 0{,}427 & 0{,}374 \\ 0{,}024 & 0{,}355 & 0{,}330 & -0{,}378 & 0{,}151 & 0{,}481 \\ -0{,}453 & -0{,}867 & 0{,}383 & 0{,}584 & -0{,}118 & 0{,}255 \\ -0{,}862 & 0{,}608 & 0{,}372 & 0{,}338 & 0{,}418 & 0{,}289 \\ 0{,}426 & 0{,}051 & 1{,}367 & -0{,}302 & -0{,}887 & 0{,}434 \\ -0{,}058 & -0{,}001 & -0{,}217 & 0{,}049 & 0{,}117 & -0{,}266 \end{bmatrix}$$

Perkalian *input* dengan matriks bobot menjadi *query* (Q)

$$Q_1 = Z_1 W_Q$$

$$= [0{,}186 \quad -0{,}001 \quad -0{,}008 \quad -0{,}466 \quad 0{,}419 \quad 0{,}373] \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$= [-0{,}465 \quad 0{,}194 \quad 0{,}652 \quad 0{,}411 \quad 0{,}373 \quad -0{,}421]$$

$$Q_2 = Z_2 W_Q$$

$$= [0{,}024 \quad 0 \quad -0{,}151 \quad -0{,}354 \quad 0 \quad 0{,}155] \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$= [-0{,}355 \quad 0{,}174 \quad 0{,}378 \quad -0{,}151 \quad 0{,}155 \quad 0]$$

$$Q_3 = Z_3 W_Q$$

$$= [-0{,}443 \quad -0{,}727 \quad 0{,}128 \quad 0{,}141 \quad 0{,}01 \quad -0{,}255] \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$= [0{,}867 \quad -0{,}571 \quad -0{,}584 \quad 0{,}137 \quad -0{,}255 \quad -0{,}736]$$

$$Q_4 = Z_4 W_Q$$

$$= [-0{,}464 \quad 0{,}482 \quad -0{,}02 \quad -0{,}126 \quad 0{,}398 \quad -0{,}392] \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$= [-0{,}608 \quad -0{,}444 \quad -0{,}338 \quad 0{,}379 \quad -0{,}392 \quad 0{,}084]$$

$$Q_5 = Z_5 W_Q$$

$$= [0{,}249 \quad -0{,}001 \quad 0{,}71 \quad -0{,}053 \quad -0{,}177 \quad -0{,}507] \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$= [-0{,}051 \quad -0{,}461 \quad 0{,}302 \quad 0{,}532 \quad -0{,}507 \quad 0{,}176]$$

$$Q_6 = Z_6 W_Q$$

$$= [-0{,}048 \quad 0 \quad -0{,}107 \quad 0{,}001 \quad 0{,}009 \quad 0{,}018] \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 & 0 \\ 1 & 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$= [0{,}001 \quad 0{,}059 \quad -0{,}049 \quad -0{,}098 \quad 0{,}018 \quad -0{,}009]$$

Setelah didapatkan matriks Q_i, selanjutnya menggabungkan matriks Q_i sehingga diperoleh matriks *query* (Q) sebagai berikut.

$$Q = \begin{bmatrix} -0.465 & 0{,}194 & 0{,}652 & 0{,}411 & 0{,}373 & -0{,}421 \\ -0{,}355 & 0{,}174 & 0{,}378 & -0{,}151 & 0{,}155 & 0 \\ 0{,}867 & -0{,}571 & -0{,}584 & 0{,}137 & -0{,}255 & -0{,}736 \\ -0{,}608 & -0{,}444 & -0{,}338 & 0{,}379 & -0{,}392 & 0{,}084 \\ -0{,}051 & -0{,}461 & 0{,}302 & 0{,}532 & -0{,}507 & 0{,}176 \\ 0{,}001 & 0{,}059 & -0{,}049 & -0{,}098 & 0{,}018 & -0{,}009 \end{bmatrix}$$

Perkalian *input* dengan matriks bobot menjadi *value* (V)

$$V_1 = Z_1 W_V$$

$$= [0{,}186 \quad -0{,}001 \quad -0{,}008 \quad -0{,}466 \quad 0{,}419 \quad 0{,}373] \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

$$= [0{,}599 \quad -1{,}031 \quad -0{,}193 \quad -0{,}551 \quad -0{,}978 \quad -0{,}978]$$

$$V_2 = Z_2 W_V$$

$$= [0{,}024 \quad 0 \quad -0{,}151 \quad -0{,}354 \quad 0 \quad 0{,}155] \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

$$= [-0{,}019 \quad -0{,}0684 \quad -0{,}283 \quad -0{,}028 \quad -0{,}179 \quad -0{,}179]$$

$$V_3 = Z_3 W_V$$

$$= [-0{,}443 \quad -0{,}727 \quad 0{,}128 \quad 0{,}141 \quad 0{,}01 \quad -0{,}255] \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

$$= [1{,}052 \quad 1{,}693 \quad 0{,}666 \quad 0{,}571 \quad 0{,}689 \quad 0{,}689]$$

$$V_4 = Z_4 W_V$$

$$= [-0{,}464 \quad 0{,}482 \quad -0{,}02 \quad -0{,}126 \quad 0{,}398 \quad -0{,}392] \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

$$= [-0{,}031 \quad 0{,}228 \quad -0{,}574 \quad 0{,}875 \quad 0{,}457 \quad 0{,}457]$$

$$V_5 = Z_5 W_V$$

$$= [0{,}249 \quad -0{,}001 \quad 0{,}71 \quad -0{,}053 \quad -0{,}177 \quad -0{,}507] \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

$$= [-0{,}223 \quad 0{,}917 \quad 1{,}468 \quad -0{,}451 \quad 0{,}436 \quad 0{,}436]$$

$$V_6 = Z_6 W_V$$

$$= [-0{,}048 \quad 0 \quad -0{,}107 \quad 0{,}001 \quad 0{,}009 \quad 0{,}018] \begin{bmatrix} -1 & -1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & 0 & 0 & 0 \\ 1 & 1 & 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 & -1 \end{bmatrix}$$

$$= [-0{,}031 \quad -0{,}077 \quad -0{,}174 \quad 0{,}137 \quad 0{,}02 \quad 0{,}02]$$

Setelah didapatkan matriks V_i, selanjutnya menggabungkan matriks V_i sehingga diperoleh matriks *value* (V) sebagai berikut.

$$V = \begin{bmatrix} 0,599 & -1,031 & -0,193 & -0,551 & -0,978 & -0,978 \\ -0,019 & -0,684 & -0,283 & -0,028 & -0,179 & -0,179 \\ 1,052 & 1,693 & 0,666 & 0,571 & 0,689 & 0,689 \\ -0,031 & 0,228 & -0,574 & 0,875 & 0,457 & 0,457 \\ -0,223 & 0,917 & 1,468 & -0,451 & 0,436 & 0,436 \\ -0,031 & -0,077 & -0,174 & 0,137 & 0,020 & 0,020 \end{bmatrix}$$

Menghitung *attention score*, dimana *attention score* digunakan untuk menentukan kata mana yang memiliki nilai atau skor yang lebih berat, *attention score* (A) untuk dihitung menggunakan Persamaan (2.8). Sebelum melakukan perhitungan *attention score* (A), matriks K akan di transpose sebagai berikut.

$$K = \begin{bmatrix} -0.233 & 0,465 & -0,381 & -0,652 & 0,427 & 0,374 \\ 0,024 & 0,355 & 0,330 & -0,378 & 0,151 & 0,481 \\ -0,453 & -0,867 & 0,383 & 0,584 & -0,118 & 0,255 \\ -0,862 & 0,608 & 0,372 & 0,338 & 0,418 & 0,289 \\ 0,426 & 0,051 & 1,367 & -0,302 & -0,887 & 0,434 \\ -0,058 & -0,001 & -0,217 & 0,049 & 0,117 & -0,266 \end{bmatrix}$$

$$K^T = \begin{bmatrix} -0,233 & 0,024 & -0,453 & -0,862 & 0,426 & -0,058 \\ 0,465 & 0,355 & -0,867 & 0,608 & 0,051 & -0,001 \\ -0,381 & -0,330 & 0,383 & 0,372 & 1,367 & -0,217 \\ -0,652 & -0,378 & 0,584 & 0,338 & -0,302 & 0,049 \\ 0,427 & 0,151 & -0,118 & 0,418 & -0,887 & 0,117 \\ 0,374 & 0,481 & 0,255 & 0,289 & 0,434 & -0,266 \end{bmatrix}$$

Setelah menentukan matriks K^T selanjutnya menentukan *attention score* menggunakan dimensi d=6. *Attention score* A_i dihitung menggunakan Persamaan (2.8). Tahapan menghitung nilai *attention score* A_i sebagai berikut.

a. Menggunakan Persamaan (2.11) untuk menghitung nilai *attention score* A_i.

$$A_i = \frac{1}{\sqrt{d}}(Q_1 K^T)$$

$$A_1 = \frac{1}{\sqrt{6}}\left(\begin{bmatrix} -0,465 & 0,194 & 0,652 & 0,411 & 0,373 & -0,421 \end{bmatrix} \begin{bmatrix} -0,233 \\ 0,465 \\ -0,381 \\ -0,652 \\ 0,427 \\ 0,374 \end{bmatrix} \right)$$

$$A_1 = [-0{,}14 \quad -0{,}036 \quad 0{,}001 \quad 0{,}146 \quad -0{,}047 \quad 0{,}298]$$

b. Menerapkan fungsi aktivasi softmax $[s(a)_i]$ pada setiap elemen matriks A_1 yang telah diperoleh pada langkah sebelumnya, sebagai berikut

$$softmax(a_i) = \frac{e^{a_i}}{\sum_{i=0}^{n} e^{a_i}}$$

Diketahui $A_1 = [-0{,}14 \quad -0{,}036 \quad 0{,}001 \quad 0{,}146 \quad -0{,}047 \quad 0{,}298]$

$$\hat{S}_{1(11)} = softmax(-0{,}14) = \frac{e^{-0{,}14}}{e^{-0{,}14} + e^{-0{,}036} + e^{0{,}001} + e^{0{,}146} + e^{-0{,}047} + e^{0{,}298}} = 0{,}138$$

$$\hat{S}_{1(12)} = softmax(-0{,}036) = \frac{e^{-0{,}036}}{e^{-0{,}14} + e^{-0{,}036} + e^{0{,}001} + e^{0{,}146} + e^{-0{,}047} + e^{0{,}298}} = 0{,}153$$

$$\hat{S}_{1(13)} = softmax(0{,}001) = \frac{e^{0{,}001}}{e^{-0{,}14} + e^{-0{,}036} + e^{0{,}001} + e^{0{,}146} + e^{-0{,}047} + e^{0{,}298}} = 0{,}159$$

$$\hat{S}_{1(14)} = softmax(0{,}146) = \frac{e^{0{,}146}}{e^{-0{,}14} + e^{-0{,}036} + e^{0{,}001} + e^{0{,}146} + e^{-0{,}047} + e^{0{,}298}} = 0{,}184$$

$$\hat{S}_{1(15)} = softmax(-0{,}047) = \frac{e^{-0{,}047}}{e^{-0{,}14} + e^{-0{,}036} + e^{0{,}001} + e^{0{,}146} + e^{-0{,}047} + e^{0{,}298}} = 0{,}152$$

$$\hat{S}_{1(16)} = softmax(0{,}298) = \frac{e^{0{,}298}}{e^{-0{,}14} + e^{-0{,}036} + e^{0{,}001} + e^{0{,}146} + e^{-0{,}047} + e^{0{,}298}} = 0{,}214$$

Didapatkan *softmax* A_1 sebagai matriks S_1 sebagai berikut.

$$\hat{S}_1 = [0{,}138 \quad 0{,}152 \quad 0{,}159 \quad 0{,}184 \quad 0{,}152 \quad 0{,}214]$$

c. Menghitung nilai berbobot dilakukan dengan mengalikan setiap elemen pada matriks $\hat{S}_1$ yang diperoleh dari langkah sebelumnya dengan matriks *value* V_i. Berdasarkan hasil perhitungan sebelumnya, diketahui bahwa

$\hat{S}_1 = [0{,}138 \quad 0{,}152 \quad 0{,}159 \quad 0{,}184 \quad 0{,}152 \quad 0{,}214]$, sehingga proses perhitungannya adalah sebagai berikut:

$$\hat{S}_{1(1,1)}V_1 = 0{,}138 \times [0{,}6 \quad -1{,}03 \quad -0{,}19 \quad -0{,}55 \quad -0{,}98 \quad -0{,}98]$$

$$= [0{,}083 \quad -0{,}143 \quad -0{,}027 \quad -0{,}076 \quad -0{,}135 \quad -0{,}135]$$

$$\hat{S}_{1(1,2)}V_2 = 0{,}153 \times [-0{,}02 \quad -0{,}68 \quad -0{,}28 \quad -0{,}03 \quad -0{,}18 \quad -0{,}18]$$

$$= [-0{,}003 \quad -0{,}105 \quad -0{,}043 \quad -0{,}004 \quad -0{,}027 \quad -0{,}027]$$

$$\hat{S}_{1(1,3)}V_3 = 0{,}159 \times [1{,}05 \quad 1{,}69 \quad 0{,}67 \quad 0{,}57 \quad 0{,}69 \quad 0{,}69]$$

$$= [0{,}167 \quad 0{,}269 \quad 0{,}106 \quad 0{,}091 \quad 0{,}11 \quad 0{,}11]$$

$$\hat{S}_{1(1,4)}V_4 = 0{,}184 \times [-0{,}03 \quad 0{,}23 \quad -0{,}57 \quad 0{,}88 \quad 0{,}46 \quad 0{,}46]$$

$$= [-0{,}006 \quad 0{,}042 \quad -0{,}105 \quad 0{,}161 \quad 0{,}084 \quad 0{,}084]$$

$$\hat{S}_{1(1,5)}V_5 = 0{,}152 \times [-0{,}22 \quad 0{,}92 \quad 1{,}47 \quad -0{,}45 \quad 0{,}44 \quad 0{,}44]$$

$$[-0{,}034 \quad 0{,}139 \quad 0{,}223 \quad -0{,}068 \quad 0{,}066 \quad 0{,}066]$$

$$\hat{S}_{1(1,6)}V_6 = 0{,}214 \times [-0{,}03 \quad -0{,}08 \quad -0{,}17 \quad 0{,}14 \quad 0{,}02 \quad 0{,}02]$$

$$= [-0{,}077 \quad -0{,}016 \quad -0{,}037 \quad 0{,}029 \quad 0{,}004 \quad 0{,}004]$$

Melakukan penjumlahan seluruh matriks *value* V_i yang sudah diberi bobot menjadi matriks $S_{i(i,j)}V_i$ untuk mendapatkan *output* matriks P_i sebagai berikut.

$$P_1 = [S_{1(1,1)}V_1] + [S_{1(1,2)}V_2] + [S_{1(1,3)}V_3] + [S_{1(1,4)}V_4] + [S_{1(1,5)}V_5] + [S_{1(1,6)}V_6]$$

$$P_1 = [0{,}201 \quad 0{,}186 \quad 0{,}116 \quad 0{,}132 \quad 0{,}102 \quad 0{,}102]$$

d. Dilakukan perhitungan yang sama untuk menghitung matriks P_i dimana $i = 1,2,3,4,5,6$ sehingga didapatkan hasil sebagai berikut.

$$P_2 = [0{,}216 \quad 0{,}367 \quad 0{,}218 \quad 0{,}115 \quad 0{,}135 \quad 0{,}135]$$

$$P_3 = [0{,}284 \quad 0{,}239 \quad 0{,}197 \quad 0{,}083 \quad 0{,}076 \quad 0{,}076]$$

$$P_4 = [0{,}226 \quad 0{,}199 \quad 0{,}152 \quad 0{,}116 \quad 0{,}094 \quad 0{,}094]$$

$$P_5 = [0{,}233 \quad 0{,}164 \quad 0{,}106 \quad 0{,}117 \quad 0{,}068 \quad 0{,}068]$$

$$P_6 = [0{,}221 \quad 0{,}170 \quad 0{,}153 \quad 0{,}088 \quad 0{,}072 \quad 0{,}072]$$

4. *Multi-head Attention*

Contoh 5. ***Multi-head Attention***

Hasil dari proses *self attention* dilanjutkan ke proses *multi-head attention.* Sebelum dihitung nilai *multi-head attention,* diketahui bobot untuk *multi head attention* yang dibangkitkan secara acak oleh komputer sebagai berikut.

$$W_O = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 & 1 & 1 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Selanjutnya, dihitung nilai *multi-head attention* menggunakan *concate* nilai P_t.

$$M = Concat(P_1, P_2, P_3, P_4, P_5, P_6)W_O$$

$$= \begin{bmatrix} 0{,}201 & 0{,}186 & 0{,}116 & 0{,}132 & 0{,}102 & 0{,}102 \\ 0{,}216 & 0{,}367 & 0{,}218 & 0{,}115 & 0{,}135 & 0{,}135 \\ 0{,}284 & 0{,}239 & 0{,}197 & 0{,}083 & 0{,}076 & 0{,}076 \\ 0{,}226 & 0{,}199 & 0{,}152 & 0{,}116 & 0{,}094 & 0{,}094 \\ 0{,}233 & 0{,}164 & 0{,}106 & 0{,}117 & 0{,}068 & 0{,}068 \\ 0{,}221 & 0{,}170 & 0{,}153 & 0{,}088 & 0{,}072 & 0{,}072 \end{bmatrix} \begin{bmatrix} 0 & -0 & 1 & 0 & 0 & 1 \\ 0 & -1 & 1 & 1 & 1 & 1 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & -0 & 0 & 1 & 0 & 0 \\ 0 & -0 & 0 & 0 & 0 & 0 \\ 0 & -0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0{,}132 & 0{,}071 & 0{,}387 & 0{,}319 & 0{,}288 & 0{,}387 \\ 0{,}115 & 0{,}149 & 0{,}582 & 0{,}481 & 0{,}502 & 0{,}582 \\ 0{,}083 & 0{,}042 & 0{,}523 & 0{,}322 & 0{,}315 & 0{,}523 \\ 0{,}116 & 0{,}047 & 0{,}425 & 0{,}316 & 0{,}294 & 0{,}425 \\ 0{,}117 & 0{,}057 & 0{,}396 & 0{,}280 & 0{,}231 & 0{,}396 \\ 0{,}088 & 0{,}017 & 0{,}391 & 0{,}258 & 0{,}242 & 0{,}391 \end{bmatrix}$$

5. *Normalozation Layer*

Proses pada *normalization layer* dilakukan menggunakan Persamaan (2.16), Persamaan (2.17), dan Persamaan (2.18).

Contoh 6. *Normalization Layer*

Hasil dari proses *multi-head attention* dilanjutkan ke proses *normalization layer.* Matriks M dianggap sebagai *input* untuk *normalization layer* yang berukuran 6×6. Pertama-tama ditentukan data dalam *mini batchs* (d) dan jumlah dari *mini batch* (t) berdasarkan ukuran matriks M yaitu 6×6 ,sehingga diperoleh $d = 6$ dan t= 6. Dihitung rata-rata untuk setiap pada matriks M dengan Persamaan (2.16).

$$\begin{aligned} u_1 &= \frac{1}{6}\sum_{i=1}^{6}(m)_{i,1} \\ &= \frac{1}{6}(0{,}132 + 0{,}115 + 0{,}083 + 0{,}116 + 0{,}117 + 0{,}088) \\ &= \frac{1}{6}(0{,}652) \\ &= 0{,}109 \end{aligned}$$

Dilakukan perhitungan yang sama untuk menghitung u_i dimana $i = 1,2,3,4,5,6$ sehingga didapatkan hasil sebagai berikut.

$$u_2 = 0{,}064 \quad u_3 = 0{,}451 \quad u_4 = 0{,}329 \quad u_5 = 0{,}312 \quad u_6 = 0{,}451$$

Setelah didapatkan nilai setiap u_i untuk $i = 1,2,3,4,5,6$, selanjutnya menghitung variansi σ_j^2 untuk setiap *mini batch* pada matriks M menggunakan Persamaan (2.17).

$$s_1^2 = \frac{1}{5}\sum_{i=1}^{6}\left((m)_{i,1} - u_1\right)^2$$

$$= \frac{1}{5}((0{,}132 - 0{,}109)^2 + (0{,}115 - 0{,}109)^2 + (0{,}083 - 0{,}109)^2 + (0{,}116 - 0{,}109)^2 + (0{,}117 - 0{,}109)^2 + (0{,}088 - 0{,}109)^2 + (0{,}088 - 0{,}109)^2)$$

$$= 0{,}0003$$

Dilakukan perhitungan yang sama untuk menghitung σ_j^2 dimana $j = 1,2,3,4,5,6$ sehingga didapatkan hasilsebagai berikut.

$$\sigma_2^2 = 0{,}0017 \quad \sigma_3^2 = 0{,}0056 \quad \sigma_4^2 = 0{,}0052 \quad \sigma_5^2 = 0{,}0081 \quad \sigma_6^2 = 0{,}0056$$

Dilakukan normalisasi setiap data pada matriks M menggunakan Persamaan (2.18) sebagai berikut.

$$n_{1,1} = \frac{(m)_{1,1} - u_1}{\sqrt{s_1^2}} = \frac{0{,}132 - 0{,}109}{\sqrt{0{,}0003}} = 0{,}1376$$

$$n_{2,1} = \frac{(m)_{2,1} - u_1}{\sqrt{s_1^2}} = \frac{0{,}115 - 0{,}109}{\sqrt{0{,}0003}} = 0{,}352$$

$$n_{3,1} = \frac{(m)_{3,1} - u_1}{\sqrt{s_1^2}} = \frac{0{,}083 - 0{,}109}{\sqrt{0{,}0003}} = -1{,}476$$

$$n_{4,1} = \frac{(m)_{4,1} - u_1}{\sqrt{s_1^2}} = \frac{0{,}116 - 0{,}109}{\sqrt{0{,}0003}} = 0{,}449$$

$$n_{5,1} = \frac{(m)_{5,1} - u_1}{\sqrt{s_1^2}} = \frac{0{,}117 - 0{,}109}{\sqrt{0{,}0003}} = 0{,}474$$

$$n_{6,1} = \frac{(m)_{6,1} - u_1}{\sqrt{s_1^2}} = \frac{0{,}088 - 0{,}109}{\sqrt{0{,}0003}} = -1{,}174$$

Dilakukan perhitungan yang sama untuk setiap entri matriks didapatkan matriks hasil *normalization layer* (N) sebagai berikut.

$$N = \begin{bmatrix} 1{,}376 & 0.163 & -0{,}849 & -0{,}151 & -0{,}268 & -0{,}849 \\ 0{,}352 & 2{,}058 & 1{,}753 & 2{,}117 & 2{,}116 & 1{,}753 \\ -1{,}476 & -0{,}524 & 0{,}962 & -0{,}101 & 0{,}030 & 0{,}962 \\ 0{,}449 & -0{,}412 & -0{,}341 & -0{,}192 & -0{,}204 & -0{,}341 \\ 0{,}474 & -0{,}160 & -0{,}727 & -0{,}684 & -0{,}898 & -0{,}727 \\ -1{,}174 & -1{,}126 & -0{,}797 & -0{,}990 & -0{,}776 & -0{,}797 \end{bmatrix}$$

6. *Feed Forward Layer*

Feed forward layer terdiri dari tiga lapisan utama, yaitu *input, hidden, dan output layer. Feed forward layer* dihitung menggunakan Persamaan (2.9).

Contoh 7. *Feed Forward Layer*

Matriks hasil *normalization layer* (*N*) digunakan sebagai *input* untuk *feed forward layer*, schingga dipcrolch cntri matriks scbagai bcrikut.

$$(n)_{1,1} = 1{,}376, \quad (n)_{1,2} = 0{,}163, \quad (n)_{1,3} = -0{,}849,$$

$$(n)_{1,4} = -0{,}151, \quad (n)_{1,5} = -0{,}268, \quad (n)_{1,6} = -0{,}849$$

Sclanjutnya komputer akan membangkitkan nilai bobot secara acak dengan 2 *hidden layer* seperti pada Tabel 4.4.

Tabel 4.4.Nilai bobot *hidden layer* dan *output*

Hidden Layer 1		*Hidden Layer* 2		*Output*	
Bobot	Nilai	Bobot	Nilai	Bobot	Nilai
$w_{1,1}$	0,21	$w_{2,1}$	0,4	$w_{3,1}$	0,34
$w_{1,2}$	0,33	$w_{2,2}$	0,2	$w_{3,2}$	0,55
$w_{1,3}$	0,45	$w_{2,3}$	0,4	$w_{3,3}$	0,24
$w_{1,4}$	0,51	$w_{2,4}$	0,55	$w_{3,4}$	0,04
$w_{1,5}$	0,03	$w_{2,5}$	0,01	$w_{3,5}$	0,99
$w_{1,6}$	0,49	$w_{2,6}$	0,47	$w_{3,6}$	0,01
$w_{1,7}$	0,44	$w_{2,7}$	0,35	$w_{3,7}$	0,01
$w_{1,8}$	0,01	$w_{2,8}$	0,43	$w_{3,8}$	0,24
$w_{1,9}$	0,45	$w_{2,9}$	0,65		
$w_{1,10}$	0,34	$w_{2,10}$	0,35		
$w_{1,11}$	0,05	$w_{2,11}$	0,21		
$w_{1,12}$	0,2	$w_{2,12}$	0,01		
$w_{1,13}$	0,2	$w_{2,13}$	0,34		

Tabel 4.4.Nilai bobot *hidden layer* dan *output*

Hidden Layer 1		*Hidden Layer* 2		*Output*	
Bobot	Nilai	Bobot	Nilai	Bobot	Nilai
$w_{1,14}$	0,3	$w_{2,14}$	0,04		
$w_{1,15}$	0,38	$w_{2,15}$	0,33		
$w_{1,16}$	0,12	$w_{2,16}$	0,01		
$w_{1,17}$	0,04	$w_{2,17}$	0,28		
$w_{1,18}$	0,05	$w_{2,18}$	0,08		
$w_{1,19}$	0,24	$w_{2,19}$	0,22		
$w_{1,20}$	0,01	$w_{2,20}$	0,39		
$w_{1,21}$	0,33	$w_{2,21}$	0,41		
$w_{1,22}$	0,97	$w_{2,22}$	0,45		
$w_{1,23}$	0,04	$w_{2,23}$	0,2		
$w_{1,24}$	0,72	$w_{2,24}$	0,35		
$w_{1,25}$	0,1				
$w_{1,26}$	0,01				
$w_{1,27}$	0,77				
$w_{1,28}$	0,24				
$w_{1,29}$	0,7				
$w_{1,30}$	0,01				
$w_{1,31}$	0,23				
$w_{1,32}$	0,01				
$w_{1,33}$	0,4				
$w_{1,34}$	0,49				
$w_{1,35}$	0,01				
$w_{1,36}$	0,01				

Tabel 4.4 merupakan nilai bobot dari *hidden layer* dan *output* yang digunakan pada *feed forward layer.* Dengan nilai bias yang digunakan sebagai berikut.

$$b_1 = 5 \quad b_2 = 3 \quad b_3 = 1$$

Kemudian dihitung nilai *output* untuk *hidden layer* 1 menggunakan Persamaan (2.9). Selanjutnya digunakan fungsi aktivasi ReLU.

$$Y_{1,1} = b_1 + \left(w_{1,1}(n)_{1,1}\right) + (w_{1,2}(n)_{1,2} + (w_{1,3}(n)_{1,3}) + \left(w_{1,4}(n)_{1,4}\right) + (w_{1,5}(n)_{1,5}) + (w_{1,6}(n)_{1,6})$$

$$Y_{1,1} = 5 + (0{,}21 \times 1{,}376) + (0{,}33 \times 0{,}163) + (0{,}45 \times -0{,}849) + (0{,}51 \times -0{,}151) + (0{,}03 \times -0{,}268) + (0{,}49 \times -0{,}849)$$

$$Y_{1,1} = 4{,}46$$

$$r_{1,1} = ReLU(z_{1,1})$$

$$r_{1,1} = ReLU(4{,}46)$$

$$r_{1,1} = 4{,}46$$

Dilakukan perhitungan yang sama untuk menghitung $h_{1,i}$ untuk $i = 1,2,3,4,5,6$ sehingga didapatkan hasil sebagai berikut.

$$r_{1,1} = 4{,}46 \quad r_{1,2} = 4{,}99 \quad r_{1,3} = 4{,}93 \quad r_{1,4} = 4{,}28 \quad r_{1,5} = 4{,}25 \quad r_{1,6} = 4{,}89$$

Kemudian hasil dari *hidden layer* 1 digunakan untuk *hidden layer* 2 sebagai berikut.

$$\hat{z}_{2,1} = b_2 + (w_{2,1}r_{1,1}) + (w_{2,2}r_{1,2} + (w_{2,3}r_{1,3}) + (w_{2,4}r_{1,4}) + (w_{2,5}r_{1,5}) + (w_{2,6}r_{1,6})$$

$$\hat{z}_{2,1} = 3 + (0{,}4 \times 4{,}46) + (0{,}2 \times 4{,}99) + (0{,}4 \times 4{,}93) + (0{,}55 \times 4{,}28) + (0{,}01 \times 4{,}25) + (0{,}47 \times 4{,}89)$$

$$\hat{z}_{2,1} = 12{,}45$$

$$r_{2,1} = ReLU(\hat{z}_{2,1})$$

$$r_{2,1} = ReLU(12{,}45)$$

$$r_{2,1} = 12{,}45$$

Dilakukan perhitungan yang sama untuk menghitung $h_{2,i}$ untuk $i = 1,2,3,4$ sehingga didapatkan hasil sebagai berikut.

$$r_{2,1} = 12{,}45 \quad r_{2,2} = 12{,}35 \quad r_{2,3} = 7{,}97 \quad r_{2,4} = 12{,}44$$

Kemudian hasil *hidden layer* 2 digunakan untuk *output* dengan perhitungan berikut.

$$output_{3,1} = b_3 + (w_{3,1}r_{2,1}) + (w_{3,2}r_{2,2} + (w_{3,3}r_{2,3}) + (w_{3,4}r_{2,4})$$

$$output_{3,1} = 1 + (0{,}34 \times 12{,}45) + (0{,}55 \times 12{,}35) + (0{,}24 \times 7{,}97) + (0{,}04 \times 12{,}44)$$

$$r_{3,1} = 13{,}44$$

$$r = ReLU(z_{3,1})$$

$$r_{3,1} = ReLU(13{,}44)$$

$$r_{3,1} = 13{,}44$$

Dilakukan perhitungan yang sama untuk menghitung $r_{3,i}$ untuk $i = 1{,}2$ sehingga didapatkan hasil sebagai berikut.

$$r_{3,1} = 13{,}44$$

$$r_{3,2} = 15{,}52$$

Sehingga diperoleh matriks *output* dari *feed forward layer* sebagai matriks R sebagai berikut.

$$R_1 = \begin{bmatrix} 13{,}44 \\ 15{,}52 \end{bmatrix}$$

Dengan langkah yang sama dilakukan perhitungan untuk mendapatkan matriks *output* pada baris ke-i untuk $i = 1{,}2{,}3{,}4{,}5{,}6$

$$R_2 = \begin{bmatrix} 20{,}07 \\ 23{,}45 \end{bmatrix} \quad R_3 = \begin{bmatrix} 14{,}60 \\ 16{,}90 \end{bmatrix} \quad R_4 = \begin{bmatrix} 13{,}52 \\ 15{,}63 \end{bmatrix} \quad R_5 = \begin{bmatrix} 12{,}43 \\ 14{,}30 \end{bmatrix} \quad R_6 = \begin{bmatrix} 10{,}91 \\ 12{,}46 \end{bmatrix}$$

7. *Average Pooling*

average pooling merupakan teknik yang digunakan untuk mengurangi jumlah parameter untuk mengurangi *overfitting*.

Contoh 8. *Average Pooling*

Average pooling dihitung menggunakan nilai dari R_1, R_2, R_3, R_4, R_5, dan R_6 sebagai *input*. Setiap entri dari matriks R_i untuk $i = 1,2,3,4,5,6$ dihitung rata-ratanya sebaagi berikut.

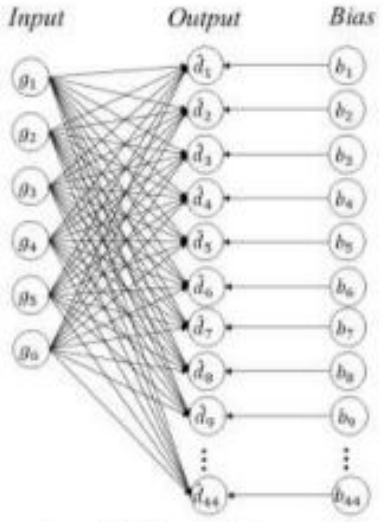
$g_1 = 14{,}48 \quad g_2 = 21{,}76 \quad g_3 = 15{,}75 \quad g_4 = 14{,}57 \quad g_5 = 13{,}36 \quad g_6 = 11{,}68$

Nilai *average pooling* dilakukan *softmax* schingga mcnghasilkan matriks *G* scbagai berikut.

$$G = \begin{bmatrix} 0{,}00068 \\ 0{,}99584 \\ 0{,}00245 \\ 0{,}00075 \\ 0{,}00023 \\ 0{,}00004 \end{bmatrix}$$

8. *Dense Layer*

Proses *dense layer* dapat dihitung menggunakan Persamaan (2.19) pada contoh 9 berikut. Ilustrasi *dense layer* menggunakan 6 *input* yang menghasilkan 44 *output* dapat dilihat pada Gambar 4.9.



Gambar 4.9. Ilustrasi *dense layer*

Contoh 9. *Dense Layer*

Untuk menghitung *dense layer*, ditentukan matriks bobot W_f dan bias yang dibangkitkan secara acak oleh komputer, seperti dibawah ini:

$$W_f = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}, b_f = \begin{bmatrix} 0{,}01 \\ 0{,}01 \end{bmatrix}$$

Selanjutnya menghitung *dense layer* dapat dihitung menggunakan Persamaan (2.19).

$$\bar{D} = softmax\left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & -1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0{,}00068 \\ 0{,}99584 \\ 0{,}00245 \\ 0{,}00075 \\ 0{,}00023 \\ 0{,}00004 \end{bmatrix} + \begin{bmatrix} 0{,}01 \\ 0{,}01 \end{bmatrix}\right)$$

$$= softmax \begin{bmatrix} 2{,}746 \\ 1{,}014 \\ 1{,}014 \\ 1{,}014 \\ 1{,}010 \\ 0{,}375 \\ 1{,}010 \\ 0{,}375 \\ 0{,}375 \\ 1{,}010 \\ 0{,}372 \\ 0{,}375 \\ 0{,}375 \\ 1{,}010 \\ 1{,}010 \\ 0{,}375 \\ 1{,}010 \\ 0{,}375 \\ 1{,}010 \\ 0{,}375 \\ 0{,}375 \\ 1{,}010 \\ 0{,}372 \\ 0{,}375 \\ 1{,}014 \\ 1{,}010 \\ 1{,}010 \\ 1{,}014 \\ 1{,}010 \\ 0{,}375 \\ 1{,}010 \\ 0{,}375 \\ 1{,}014 \\ 1{,}010 \\ 0{,}372 \\ 0{,}375 \\ 1{,}014 \\ 1{,}010 \\ 1{,}010 \\ 0{,}375 \\ 1{,}010 \\ 0{,}375 \\ 1{,}010 \\ 1{,}014 \end{bmatrix}$$

Menentukan nilai *softmax* dari masing-masing *entry* matriks sebagai berikut.

$$softmax(1{,}010) = \frac{2{,}746}{43{,}899} = 0{,}0625$$

Melakukan perhitungan dengan cara yang sama sampai *entry* ke 44, sehingga diperoleh matriks *dense layer* sebagai berikut.

$$\bar{D} = \begin{bmatrix} 0{,}079 \\ 0{,}029 \\ 0{,}029 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \end{bmatrix}$$

9. *Loss Function*

Loss function merupakan fungsi yang digunakan untuk mengukur kinerja model dalam memprediksi data. Proses pada *loss function* dapat dilihat pada Contoh 10.

Contoh 10. *Loss Function*

Loss function dihitung dengan melibatkan nilai hasil prediksi dan nilai sebenarnya dari klasifikasi. Nilai prediksi dimisalkan $\bar{D}$ dari proses *dense layer* dan nilai

sebenarnya klasifikasi dimisalkan sebagai matriks $\hat{Y}$ dengan ukuran yang sama seperti matriks $\hat{D}$ yaitu 44×1 sebagai berikut.

$$\hat{D} = \begin{bmatrix} 0{,}079 \\ 0{,}029 \\ 0{,}029 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \end{bmatrix} \hat{Y} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

Nilai *loss fuction* dihitung menggunakan Persamaan (2.24) sebagai berikut.

$$l(\hat{y}, \hat{d}) = -\big((1 \ln 0{,}079) + (0 \ln 0{,}029) + (0 \ln 0{,}029) + (1 \ln 0{,}029) + (0 \ln 0{,}029) + (0 \ln 0{,}011) + (0 \ln 0{,}029) + (0 \ln 0{,}011) + (0 \ln 0{,}011) + (0 \ln 0{,}029) + (0 \ln 0{,}011) + (0 \ln 0{,}011) +$$

$$(0 \ln 0{,}011) + (0 \ln 0{,}029) + (0 \ln 0{,}029) + (0 \ln 0{,}011) +$$
$$(0 \ln 0{,}029) + (0 \ln 0{,}011) + (0 \ln 0{,}029) + (0 \ln 0{,}011) +$$
$$(0 \ln 0{,}011) + (0 \ln 0{,}029) + (0 \ln 0{,}011) + (0 \ln 0{,}011) +$$
$$(0 \ln 0{,}029) + (0 \ln 0{,}029) + (0 \ln 0{,}029) + (0 \ln 0{,}029) +$$
$$(0 \ln 0{,}029) + (0 \ln 0{,}011) + (0 \ln 0{,}029) + (0 \ln 0{,}029) +$$
$$(0 \ln 0{,}011) + (0 \ln 0{,}029) + (0 \ln 0{,}011) + (0 \ln 0{,}011) +$$
$$s(0 \ln 0{,}029) + (0 \ln 0{,}029) + (0 \ln 0{,}029) + (0 \ln 0{,}029) +$$
$$(0 \ln 0{,}011) + (0 \ln 0{,}011) + (0 \ln 0{,}029) + (0 \ln 0{,}029))$$
$$= 2{,}54$$

Sehingga,didapatkan nilai *loss* dari hasil prediksi menggunakan *categorical cross entropy* adalah 2,54 yang menunjukkan kesalahan model dalam memprediksi label klasifikasi pada kelas 1.

10. *Adam Optimizer*

Adam optimizer menggunakan nilai momen pertama, momen kedua, dari gradien untuk memperbarui nilai pada bobot. Proses pada *adam optimizer* dilakukan menggunakan Persamaan (2.27), Persamaan (2.28), Persamaan (2.29), Persamaan (2.30), dan Persamaan (2.31).

Misalkan diambil matriks $\tilde{D}$, matriks G dan matriks $\hat{Y}$ sebagai berikut.

$$\tilde{D} = \begin{bmatrix} 0{,}079 \\ 0{,}029 \\ 0{,}029 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}011 \\ 0{,}029 \\ 0{,}029 \end{bmatrix}, \tilde{V} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, G = \begin{bmatrix} 0{,}00068 \\ 0{,}99584 \\ 0{,}00245 \\ 0{,}00075 \\ 0{,}00023 \\ 0{,}00004 \end{bmatrix}$$

Langkah-langkah untuk memperbarui bobot menggunakn *adam optimizer* dengan adalah sebagai berikut.

a. Menginisialisasi nilai awal pada momen pertama $r_0 = 0$, nilai awal pada momen kedua $a_0 = 0$, *decay rate* untuk momen pertama $\beta_1 = 0{,}9$, pada momen kedua $\beta_2 = 0{,}99$, *epoch* sebanyak 30, dan *learning rate* $c = 0{,}0001$.

b. Dihitung nilai gradien dari *loss function* menggunakan *sparse categorical cross entropy* sebagai berikut.

$$\hat{G} = (\hat{D} - \hat{Y})G^T$$

$$\hat{G} = \begin{bmatrix} 0{,}079 - 1 \\ 0{,}029 - 0 \\ 0{,}029 - 0 \\ 0{,}029 - 0 \\ 0{,}029 - 0 \\ 0{,}011 - 0 \\ 0{,}029 - 0 \\ 0{,}011 - 0 \\ 0{,}011 - 0 \\ 0{,}029 - 0 \\ 0{,}011 - 0 \\ 0{,}011 - 0 \\ 0{,}011 - 0 \\ 0{,}029 - 0 \\ 0{,}029 - 0 \\ 0{,}011 - 0 \\ 0{,}029 - 0 \\ 0{,}011 - 0 \\ 0{,}029 - 0 \\ 0{,}011 - 0 \\ 0{,}011 - 0 \\ 0{,}029 - 0 \\ 0{,}011 - 0 \\ 0{,}011 - 0 \\ 0{,}029 - 0 \\ 0{,}029 - 0 \\ 0{,}011 - 0 \\ 0{,}029 - 0 \\ 0{,}011 - 0 \\ 0{,}029 - 0 \\ 0{,}029 - 0 \\ 0{,}011 - 0 \\ 0{,}011 - 0 \\ 0{,}029 - 0 \\ 0{,}029 - 0 \\ 0{,}029 - 0 \\ 0{,}011 - 0 \\ 0{,}029 - 0 \\ 0{,}011 - 0 \\ 0{,}029 - 0 \\ 0{,}029 - 0 \end{bmatrix} \begin{bmatrix} 0{,}00068 & 0{,}99584 & 0{,}00245 & 0{,}00075 & 0{,}00023 & 0{,}00004 \end{bmatrix}$$

$$\hat{G} = \begin{bmatrix} 0 & -0{,}917 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}011 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \\ 0 & 0{,}029 & 0 & 0 & 0 & 0 \end{bmatrix}$$

c. Diperbarui nilai momen pertama dan momen kedua pada *epoch* ke-1 dengan melakukan operasi entri matriks menggunakan Persamaan (2.25) dan Persamaan (2.26).

$$\begin{aligned} r_{11} &= \beta_1 r_{1-1} + (1-\beta_1)\hat{g}_{11} \\ &= (1-0{,}9) \times 0 \end{aligned}$$

$$= 0$$

$$a_{11} = \beta_2 a_{1-1} + (1-\beta_2)\hat{g}_{11}{}^2$$

$$= (1-0{,}99) \times 0^2$$

$$= 0$$

d. Dihitung bias *correction* pada momen pertama menggunakan Persamaan (2.27) dan momen kedua menggunakan Persamaan (2.28).

$$\hat{r}_{11} = \frac{r_{11}}{(1-\beta_1{}^1)}$$

$$= \frac{0}{(1-0{,}9^1)}$$

$$= 0$$

$$\hat{a}_{11} = \frac{a_{11}}{(1-\beta_2{}^1)}$$

$$= \frac{0}{(1-0{,}99^1)}$$

$$= 0$$

e. Langkah selanjutnya adalah memperbarui bobot *epoch* ke-1 W_1 menggunakan Persamaan (2.29).

$$w_{11} = w_f - c\frac{\widehat{r_{11}}}{\sqrt{\widehat{a_{11}}} + \varepsilon}$$

$$= 1 - 0{,}0001\frac{0}{\sqrt{0} + 0{,}00000001}$$

$$= 1-(-0{,}1)$$

$$= 1{,}1$$

Dengan cara yang sama dilakukan perhitungan untuk semua entri pada matriks bobot W_f sehingga diperoleh matriks baru sebagai berikut.

$$W = \begin{bmatrix} 1{,}1 & 1{,}1 & 1{,}1 & 1{,}1 & 1{,}1 & 1{,}1 \\ 0{,}9 & -0{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 0{,}9 \\ 0{,}9 & -0{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 0{,}9 \\ 0{,}9 & -0{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 0{,}9 \\ -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ -1{,}1 & -1{,}1 & -1{,}1 & -1{,}1 & -1{,}1 & -1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ 0{,}9 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ -0{,}1 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ -1{,}1 & -1{,}1 & -1{,}1 & -1{,}1 & -1{,}1 & -1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ 0{,}9 & -0{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ 0{,}9 & -0{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ -0{,}1 & -0{,}1 & -0{,}1 & 0{,}1 & -0{,}1 & -0{,}1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ 0{,}9 & -0{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 0{,}9 \\ -0{,}1 & 0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ -1{,}1 & -1{,}1 & -1{,}1 & -1{,}1 & -1{,}1 & -1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ 0{,}9 & -0{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 0{,}9 \\ -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ -0{,}1 & 0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ 0{,}9 & -1{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 1 \\ -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 & -0{,}1 \\ 0{,}9 & -0{,}1 & 0{,}9 & 0{,}9 & 0{,}9 & 0{,}9 \end{bmatrix}$$

4.8 Hasil

4.8.1 *Training*

Pada tahap ini, dilakukan perlatihan pada model menggunakan kombinasi arsitektur LSTM dan *Transformer*. Proses *training* model dilakukan menggunakan data yang tekah diaugmentasi. Pada 52.676 data yang digunakan dalam penelitian

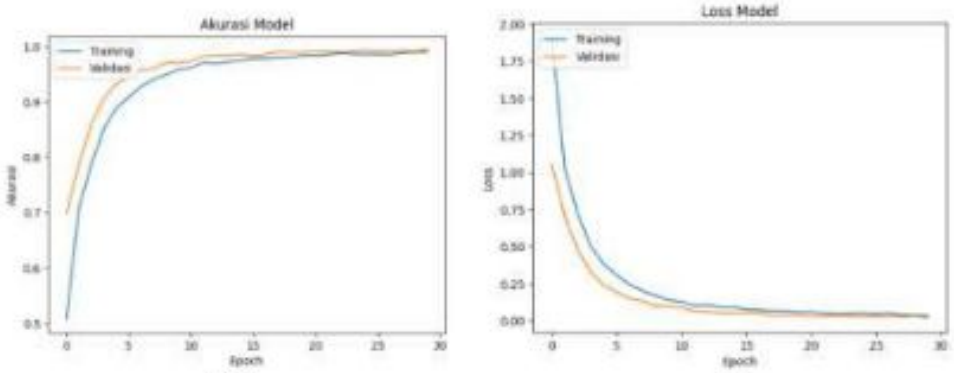
ini, dilakukan *split* data menjadi 80% data *training* dan 20% data *testing* sehingga digunakan 42.141 data secara acak yang digunakan untuk data *training* model. Dari 42.141 data, dilakukan kembali *split* data menjadi 80% data *training* dan 20% data validasi. Pada proses *training* model, terdapat parameter yang digunakan yaitu *epoch,* dan *bacth size.* Pada penelitian ini, umlah *epoch* yang digunakan adalah 30 *epoch,* dan *batch size* yang digunakan adalah 64. Hasil proses *training* model dapat dilihat pada Gambar 4.10.

```
Epoch 1/30
494/494 [──────────────────────────────] - 448s 877ms/step - loss:
1.7602 - accuracy: 0.5361 - val_loss: 0.9764 - val_accuracy: 0.5799
Epoch 2/30
494/494 [──────────────────────────────] - 390s 790ms/step - loss:
0.9570 - accuracy: 0.6820 - val_loss: 0.8350 - val_accuracy: 0.7003
Epoch 3/30
494/494 [==============================] - 410s 830ms/step - loss:
0.8170 - accuracy: 0.7010 - val_loss: 0.7897 - val_accuracy: 0.714
Epoch 4/30
494/494 [==============================] - 414s 837ms/step - loss:
0.7350 - accuracy: 0.7506 - val_loss: 0.6724 - val_accuracy: 0.7623
Epoch 5/30
494/494 [==============================] - 378s 765ms/step - loss:
0.6398 - accuracy: 0.7756 - val_loss: 0.6001 - val_accuracy: 0.8016
```

Gambar 4.10. Hasil *training* model kombinasi arsitektur LSTM dan *Transformer*

Pada Gambar 4.10, menunjukkan hasil proses *training* menggunakan kombinasi arsitektur LSTM-Transformer pada 2 *epoch* awal dari 30 *epoch.* Pada *epoch* 1, didapatkan nilai akurasi data *training* sebesar 0,5079 dengan nilai *loss* sebesar 1,916. Pada data validasi diperoleh akurasi sebesar 0,6989 dan *loss* sebesar 1,0556. Bobot yang diperoleh pada *epoch* 1 disimpan dan digunakan pada *epoch* berikutnya. Pada *epoch* 2, didapatkan nilai akurasi data *training* sebesar 0,7080 dengan nilai *loss* sebesar 1,0266. Pada data validasi diperoleh akurasi sebesar 0,7872 dan *loss* sebesar 0,7051. Nilai *loss* pada data validasi di *epoch* 2 lebih kecil

dari nilai *loss* pada data validasi di *epoch* 1, sehingga bobot pada *epoch* 2 disimpan dandigunakan untuk *epoch* selanjutnya. Akurasi model dan loss model dapat dilihat pada Gambar 4.11.



(a) (b)

Gambar 4.11. Grafik akurasi dan *loss* pada *training* dan data Validasi

Pada Gambar 4.11 (a), dapat dilihat bahwa grafik akurasi dan validasi tidak terjadi *overfitting* dan meningkat pada setiap *epoch*. Pada *epoch* pertama didapatkan nilai akurasi data *training* sebesar 0,7 lalu nilai akurasi mengalami peningkatan hingga diatas 0,85. Pada Gambar Gambar 4.11 (b), dapat dilihat bahwa proses *training* tidak terjadi *overfitting*. Pada setiap *epoch* data nilainya mendekati nol. Berdasarkan nilai akurasi dan *loss* yang didapatkan pada proses *training*, model memiliki kinerja yang baik karena akurasi model mencapai 85% dan nilai *loss* mendekati 0.

4.8.2 *Testing*

Data yang digunakan dalam proses *testing* berjumlah 10.536 data, yang merupakan 20% dari total data setelah proses pembagian. Proses *testing* menghasilkan *confussion matrix* yang disajikan pada Tabel 4.5.

Tabel 4.5. *Confussion matrix* data ujaran kebencian *multi label*

Kelas	H0	H1	…	H43
H0	4419	69	…	1
H1	103	1269	…	0
⋮	⋮	⋮	⋮	⋮
H43	0	0	…	9

Pada Tabel 4.5, diketahui bahwa kelas H0 memprediksi jumlah data yang berhasil diklasifikasikan dengan benar sebesar 4.419 data, sedangkan nilai lainnya adalah nilai yang diprediksi sebagai kelas lain. Untuk melihat *confussion matrix* lengkap dari *dataset* ujaran kebencian *multi label* dapat dilihat pada Lampiran 1.

4.9 Evaluasi

Evaluasi dilakukan setelah proses testing untuk mengukur hasil kinerja model yang dibangun menggunakan nilai-nilai yang didapatkan dari *confusion matrix*. Ukuran evaluasi kerja yang digunakan pada penelitian ini antara lain akurasi, presisi, *recall*, dan *f1-score* yang akan dihitung menggunakan Persamaan (2.30), Persamaan (2.31), Persamaan (2.32), dan Persamaan (2.33).

1. Akurasi

Akurasi digunakan untuk melihat seberapa banyak kelas yang diprediksi dengan benar dari keseluruhan data. Dilakukan perhitungan akuraasi untuk kelas 0 menggunakan persamaan (2.30) berikut:

$$Akurasi = \frac{TP + TN}{TP + FN + TN + FP} \times 100\%$$

$$= \frac{4419 + 5447}{4419 + 425 + 245 + 5447} \times 100\%$$

$$= 94{,}49\%$$

Nilai akurasi yang dihitung menggunakan Persamaan (2.30) untuk kelas 0 yaitu sebesar 94.49% yang memiliki kategori kinerja sangat baik berdasarkan kategori

nilai kinerja arsitektur pada Tabel 2.2. Untuk melihat akurasi pada kelas lainnya, dapat dilakukan perhitungan yang sama menggunakan Persamaan (2.30).

2. Presisi

Presisi digunakan untuk melihat seberapa banyak model dapat memprediksi data aktual yang bernilai positif di antara keseluruhan data yang prediksi positif. Dilakukan perhitungan presisi untuk kelas 0 menggunakan persamaan (2.31) berikut:

$$presisi = \frac{TP}{TP + FP} \times 100\%$$

$$= \frac{4419}{4419 + 425} \times 100\%$$

$$= 91{,}02\%$$

Nilai presisi yang dihitung menggunakan Persamaan (2.31) adalah sebesar 91,02% yang memiliki kategori kinerja sangat baik berdasarkan kategori nilai kinerja arsitektur pada Tabel 2.2. Untuk melihat presisi pada kelas lainnya, dapat dilakukan perhitungan yang sama menggunakan Persamaan (2.31).

3. *Recall*

Recall digunakan untuk mengetahui seberapa banyak kelas prediksi benar positif dibandingkan dengan keseluruhan data yang benar positif. Dilakukan perhitungan *recall* untuk kelas 0 menggunakan persamaan (2.32) berikut:

$$presisi = \frac{TP}{TP + FN} \times 100\%$$

$$= \frac{4419}{4419 + 245} \times 100\%$$

$$= 94{,}49\%$$

Nilai *recall* yang dihitung menggunakan Persamaan (2.32) adalah sebesar 94,49% yang memiliki kategori kinerja sangat baik berdasarkan kategori nilai kinerja arsitektur pada Tabel 2.2. Untuk melihat *recall* pada kelas lainnya, dapat dilakukan perhitungan yang sama menggunakan Persamaan (2.32).

4. *F1-score*

F1-score digunakan untuk menghitung nilai rata-rata presisi dan *recall*. Dilakukan perhitungan *f1-score* untuk kelas 0 menggunakan Persamaan (2.33) berikut:

$$F1 - score = 2 \times \frac{recall \times preisi}{recall + preisi} \times 100\%$$

$$= 2 \times \frac{91{,}02 \times 94{,}49}{91{,}02 + 94{,}49} \times 100\%$$

$$= 92{,}72\%$$

Nilai *f1-score* yang dihitung menggunakan Persamaan (2.33) adalah sebesar 92,72% yang memiliki kategori kinerja sangat baik berdasarkan kategori nilai kinerja arsitektur pada Tabel 2.2. Untuk melihat *f1-score* pada kelas lainnya, dapat dilakukan perhitungan yang sama menggunakan Persamaan (2.33).

Hasil evaluasi performa model yang diukur menggunakan metrik akurasi, presisi, recall, dan *f1-score* menunjukkan tingkat keberhasilan model dalam mengklasifikasikan data secara keseluruhan. Perbandingan nilai evaluasi performa pada masing-masing kelas dapat dilihat pada Tabel 4.6.

Tabel 4.6 Nilai kinerja kombinasi arsitektur LSTM dan *Transformer* pada data ujaran kebencian *multi label*

Kelas	Akurasi (%)	Presisi (%)	*Recall* (%)	*f1-score* (%)
H0	94	92	94	93
H1	86	82	86	84

Tabel 4.6 Nilai kinerja kombinasi rsitektur LSTM dan *Transformer* pada data ujaran kebencian *multi label*

Kelas	Akurasi (%)	Presisi (%)	*Recall* (%)	*f1-score* (%)
H2	90	76	90	83
H3	61	80	61	70
H4	81	80	81	81
H5	76	80	76	78
H6	79	76	79	77
H7	77	75	77	76
H8	100	89	100	94
H9	58	95	58	72
H10	83	85	83	84
H11	85	84	85	85
H12	80	67	80	73
H13	60	100	60	75
H14	50	83	50	63
H15	70	78	70	74
H16	50	88	50	64
H17	67	66	67	67
H18	100	33	100	50
H19	43	75	43	55
H20	60	86	60	71
H21	71	94	71	78
H22	71	94	71	81
H23	52	52	52	52
H24	73	89	73	80
H25	75	75	75	75
H26	100	100	100	100
H27	62	89	62	73
H28	67	90	67	77
H29	85	92	85	88
H30	50	100	50	67
H31	67	67	67	67
H32	69	82	69	75
H33	78	81	78	80

Tabel 4.6 menunjukkan performa model klasifikasi ujaran kebencian *multi label* dalam bahasa Indonesia menggunakan kombinasi arsitektur LSTM dan *Transformer*. Model LSTM dan *Transformer* memiliki nilai akurasi rata-rata

sebesar 86%, yang menunjukkan bahwa model dapat memprediksi dengan benar sebagian besar data yang tersedia. Proses pelatihan menghasilkan nilai rata-rata presisi dan *recall* masing-masing sebesar 86% dan 85%. Nilai presisi dan *recall* menunjukkan bahwa model dapat mengenali kelas positif dan menghindari kesalahan prediksi dengan seimbang. Dengan demikian, hasil ini menunjukkan bahwa model mampu mengklasifikasikan sebagian besar kelas ujaran kebencian dengan kategori kinerja yang baik.

Beberapa kelas memiliki performa yang sangat baik, yaitu pada kelas H8 dan H26 yang memiliki nilai akurasi sebesar 100% dengan *f1-score* diatas 90%.. Hal tersebut menunjukkan bahwa model dapat mengenali sampel dari kelas tersebut dengan sangat baik. Model juga menunjukkan performa yang baik pada kelas H0, H1, H2, H10, H11, dan H29 dengan *f1-score* di atas 80%.

Beberapa kelas memiliki performa yang kurang optimal, seperti H37 dan H42. Kedua kelas ini memiliki akurasi rendah dengan *f1-score* di bawah 50%. Nilai ini menunjukkan bahwa model masih kesulitan dalam mengenali sampel dari kelas tersebut. Nilai presisi dan *recall* tidak seimbang terjadi pada beberapa kelas. Kelas H18 memiliki *recall* sebesar 100% tetapi presisi hanya mencapai 33%. Nilai ini menunjukkan bahwa model sering mengklasifikasikan sampel ke dalam kelas tersebut meskipun banyak prediksi yang tidak akurat. Ketidakseimbangan juga terjadi pada kelas H35 dan H42, di mana model memiliki presisi tinggi tetapi *recall* rendah. Hal ini menunjukkan bahwa meskipun model jarang salah dalam klasifikasi positif, banyak sampel yang seharusnya termasuk dalam kelas tersebut tidak

terdeteksi. Model bekerja dengan baik untuk sebagian besar kelas, tetapi masih terdapat beberapa kelemahan dalam menyeimbangkan nilai presisi dan *recall*.

4.10 Analisis dan Interpretasi Hasil

Pada tahap ini, dilakukan analisis hasil dengan membandingkan nilai evaluasi kinerja model dengan hasil dari penelitian sebelumnya. Perbandingan nilai evaluasi kinerja model dapat dilihat pada Tabel 4.7.

Tabel 4.7. Hasil kinerja model untuk klasifikasi teks ujaran kebencian

Metode	Jumlah Kelas	Hasil			
		Akurasi	Presisi	*Recall*	*F1-score*
Random Forest Decission Tree (Ibrohim & Budi., 2019)	3	77,36%	-	-	-
*Transformer (*Desiani *et al.*, 2023)	2	85,21%	83,09%	**88,79%**	**85,84%**
BiLSTM (Aurora et al., 2023)	12	80,25%	80,25%	80,25%	80,25%
LSTM dan *Transformer*	**44**	**86%**	**86%**	85%	85%

Tabel 4.5 menunjukkan perbandingan hasil kinerja model pada *dataset* ujaran kebencian dari *tweet*. Hasil perbandingan menunjukkan bahwa model LSTM dan *Transformer* memiliki kinerja terbaik dengan nilai akurasi 86%, presisi 86%, serta nilai *recall* dan *f1-score* yang seimbang dalam menangani 44 kelas. Model ini memiliki peningkatan akurasi sebesar 5,75% dibandingkan dengan BiLSTM, yang hanya melakukan klasifikasii pada 12 kelas. Hal ini menunjukkan bahwa kombinasi LSTM dan *Transformer* lebih efektif dalam menangani jumlah kelas yang lebih banyak dibandingkan dengan model BiLSTM. Metode *Transformer* pada penelitian sebelumnya, memiliki nilai *recaLl* lebih tinggi 3,79% dan *f1-score* lebih tinggi 0,84% dibandingkan dengan LSTM dan *Transformer*. Meskipun memiliki nilai *recall* dan *f1-score* lebih tinggi, *Transformer* hanya melakukan klasifikasi biner.

Metode *Random Forest Decision Tree* yang menangani 3 kelas memiliki akurasi lebih rendah 2,89% dibandingkan BiLSTM dan lebih rendah 8,64% dibandingkan LSTM dan *Transformer*. Perbedaan nilai model *Random Forest Decision Tree* menunjukkan bahwa pendekatan berbasis pohon keputusan kurang baik dalam melakukan klasifikasi ujaran kebencian. Perbandingan model menunjukkan bahwa kombinasi arsitektur LSTM dan *Transformer* memiliki kinerja yang lebih baik dalam klasifikasi *multi label* , karena mampu menangani jumlah kelas yang lebih besar dengan keseimbangan metrik evaluasi yang optimal.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

1. Penerapan teknik *back translation* dan augmentasi BERT berhasil meningkatkan jumlah data ujaran kebencian *multi label* pada *tweet* berbahasa Indonesia hingga tiga kali lipat dari data awal. Hal ini menunjukkan bahwa teknik augmentasi tersebut efektif dalam menambah data dan dapat memenuhi kebutuhan algoritma deep learning seperti LSTM dan *Transformer*.
2. Evaluasi kinerja arsitektur LSTM dan *Transformer* dalam klasifikasi teks ujaran kebencian *multi label* pada *tweet* berbahasa Indonesia yang telah mengalami augmentasi menunjukkan hasil kinerja rata-rata yang baik. Model memiliki akurasi rata-rata 86%, yang berarti model mampu mengklasifikasikan teks dengan benar terhadap seluruh data yang diuji. Presisi rata-rata sebesar 86% yang menunjukkan bahwa model dapat mengenali ujaran kebencian dengan tepat pada sebuah teks, sementara *recall* sebesar 85% yang menunjukkan bahwa model berhasil menemukan ujaran kebencian dalam data dengan kinerja yang baik. Kinerja model memiliki nilai *f1-score* sebesar 85% yang menunjukkan keseimbangan antara ketepatan dan cakupan deteksi, sehingga model memiliki performa yang stabil dalam mengidentifikasi ujaran kebencian. Model menunjukkan kinerja sangat baik dalam melakukan kalsifikasi kelas H0 dan H26, dengan nilai akurasi, presisi, *recall*, dan *f1-score* di atas 90%. Hal ini menunjukkan

bahwa model sangat baik dalam melakukan klasifikasi ujaran kebencian pada kelas H0 dan H26. Pada beberapa kelas dalam data, model masih mengalami kesulitan dalam mengenali pola ujaran kebencian terutama pada kelas H17, H23, dan H31. Nilai akurasi, presisi, *recall*, dan *f1-score* pada kelas tersebut berada di bawah 70% yang menunjukkan bahwa model kurang baik dalam mendeteksi kelas.

5.2. Saran

Penelitian ini menerapkan kombinasi arsitektur LSTM dan *Transformer* untuk klasifikasi teks serta teknik *back translation* dan BERT sebagai metode augmentasi data. Penelitian ini berhasil meningkatkan jumlah data sesuai kebutuhan arsitektur, namun belum memperhatikan keseimbangan data pada setiap kelas. Untuk penelitian selanjutnya disarankan agar menerapkan metode-metode keseimbangan data agar data lebih seimbang. Penelitian selanjutnya juga disarankan untuk menggunakan arsitektur yang lebih kompleks guna meningkatkan kinerja model dalam klasifikasi teks ujaran kebencian *multi label*.

DAFTAR PUSTAKA

Alkomah, F., & Ma, X. (2022). A literature review of textual hate speech detection methods and datasets. *Information (Switzerland)*, *13*(6), 1–22.

Antypas, D., & Camacho-Collados, J. (2023). Robust hate speech detection in social media: a cross-dataset empirical evaluation. *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, 231–242.

Apicella, A., Donnarumma, F., Isgrò, F., & Prevete, R. (2021). A survey on modern trainable activation functions. *Neural Networks*, *138*(6), 14–32.

Arbaatun, C. N., Nurjanah, D., & Nurrahmi, H. (2022). Hate speech detection on twitter through natural language processing using LSTM model. *Building of Informatics, Technology and Science (BITS)*, *4*(3), 1548–1557.

Aurora, F., Zahra, A., Sibaroni, Y., Sri, &, & Prasetyowati, S. (2023). Classification of Multi-Label of hate speech on twitter Indonesia using LSTM and BiLSTM Method. *JINAV: Journal of Information and Visualization*, *4*(2), 2746–1440.

Ayo, F. E., Folorunso, O., Ibharalu, F. T., & Osinuga, I. A. (2020). Machine learning techniques for hate speech classification of twitter data: State-of-The-Art, future challenges and research directions. *Computer Science Review*, *38*(9), 100311.

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer Normalization. *ArXiv*, *V1*(7), 1–14.

Beddiar, D. R., Jahan, M. S., & Oussalah, M. (2021). Data expansion using back translation and paraphrasing for hate speech detection. *Online Social Networks and Media*, *24*.

Beyhan, F., Çarık, B., Arın, İ., Terzioğlu, A., Yanıkoğlu, B., & Yeniterzi, R. (2022). A Turkish hate speech dataset and detection system. *2022 Language Resources and Evaluation Conference, LREC 2022*, 4177–4185.

Bilal, M., Khan, A., Jan, S., Musa, S., & Ali, S. (2023). Roman urdu hate speech detection using Transformer-based model for cyber security applications. *Sensors*, *23*(8), 1–26.

Chai, C. P. (2023). Comparison of text preprocessing methods. *Natural Language Engineering*, *29*(3), 509–553.

Chen, J., Zhou, D., Tang, Y., Yang, Z., Cao, Y., & Gu, Q. (2020). Closing the generalization gap of adaptive gradient methods in training deep neural networks. *IJCAI International Joint Conference on Artificial Intelligence*, *2021-Janua*, 3267–3275.

Chotirat, S., & Meesad, P. (2021). Part-of-speech tagging enhancement to natural language processing for Thai wh-question classification with deep learning. *Heliyon*, *7*(10), e08216.

Das, A. K., Al Asif, A., Paul, A., & Hossain, M. N. (2021). Bangla hate speech detection on social media using attention-based recurrent neural network. *Journal of Intelligent Systems*, *30*(1), 578–591.

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *1*(Mlm), 4171–4186.

Dogra, V., Verma, S., Kavita, Chatterjee, P., Shafi, J., Choi, J., & Ijaz, M. F. (2022). A complete process of text classification system using State-of-the-Art NLP models. *Computational Intelligence and Neuroscience*, *2022*(6), 26.

Domor, I., Sun, Y., & Ileberi, E. (2024). Machine Learning with Applications Artificial intelligence and sustainable development in Africa : A comprehensive review. *Machine Learning with Applications*, *18*(July),

Egger, R. (2022). Text Representations and Word Embeddings: Vectorizing Textual Data. *Tourism on the Verge*, *Part F1051*, 335–361.

Elsafoury, F. (2023). Thesis Distillation: Investigating The Impact of Bias in NLP Models on Hate Speech Detection. *BigPicture 2023 - Big Picture Workshop, Proceedings*, *June*, 53–65.

Ezhilarasi, S., & Maheswari, D. P. U. (2021). Designing the Neural Model for POS Tag classification and prediction of words from ancient stone inscription script. *Int. J. of Aquatic Science*, *12*(3), 1718–1728.

Fazil, M., Khan, S., Albahlal, B. M., Alotaibi, R. M., Siddiqui, T., & Shah, M. A. (2023). Attentional Multi-Channel Convolution with Bidirectional LSTM cell toward hate speech prediction. *IEEE Access*, *11*(2), 16801–16811.

Gasparetto, A., Marcuzzo, M., Zangari, A., & Albarelli, A. (2022). Survey on Text Classification Algorithms: from text to predictions. *Information (Switzerland)*, *13*(2), 1–39.

Hana, K. M., Adiwijaya, Al Faraby, S., & Bramantoro, A. (2020). Multi-label Classification of Indonesian Hate Speech on Twitter Using Support Vector Machines. *2020 International Conference on Data Science and Its Applications, ICoDSA 2020*.

Hernández, A., & Amigó, J. M. (2021). Attention mechanisms and their

applications to complex systems. *Entropy*, *23*(3), 1–18.

Ibrohim, M. O., & Budi, I. (2019). Multi-label hate speech and abusive language detection in Indonesian twitter. *2019 Association for Computational Linguistics (ACL)*, 46–57.

Ibrohim, M. O., & Budi, I. (2023). Hate speech and abusive language detection in Indonesian social media: Progress and challenges. *Heliyon*, *9*(8), e18647.

Kapil, P., & Ekbal, A. (2022). A Transformer based Multi-Task Learning Approach Leveraging Translated and Transliterated Data to Hate Speech Detection in Hindi. *Computer Science & Informasion Technology (CS & IT)*, 191–207.

Khan, S., Naseer, M., Hayat, M., Zamir, S. W., Khan, F. S., & Shah, M. (2022). Transformers in Vision: A Survey. *ACM Computing Surveys*, *54*(10), 1–30.

Kolesnichenko, L., Velldal, E., & Øvrelid, L. (2023). Word substitution with Masked Language Models as data augmentation for sentiment analysis. *RESOURCEFUL. 2023 - Workshop on Resources and Representations for Under-Resourced Languages and Domains, Proceedings of the 2nd*, *5*(task 10), 42–47.

Kovaleva, O., Romanov, A., Rogers, A., & Rumshisky, A. (2019). Revealing the dark secrets of Bert. *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference*, *8*(2018), 4365–4374.

Kurniawan, S., & Budi, I. (2020). Indonesian tweets hate speech target classification using machine learning. *2020 5th International Conference on Informatics and Computing, ICIC 2020*, 1–5.

Li, J., Wang, X., Tu, Z., & Lyu, M. R. (2021). On the diversity of multi-head attention. *Neurocomputing*, *454*, 14–24.

Lin, M., Chen, Q., & Yan, S. (2014). Network in network. *2nd International Conference on Learning Representations, ICLR 2014 - Conference Track Proceedings*, 1–10.

Liu, L., Liu, X., Gao, J., Chen, W., & Han, J. (2020). Understanding the difficulty of training Transformers. *Conference on Empirical Methods in Neural Language Processing*, 5747–5763.

Malik, J. S., Qiao, H., Pang, G., & Hengel, A. van den. (2022). Deep Learning for Hate Speech Detection: A Comparative Study. *International Journal of Data Science and Analytics*, *09517v2*(12), 1–18.

Marpaung, A., Rismala, R., & Nurrahmi, H. (2021). Hate speech detection in

Indonesian twitter texts using Bidirectional Gated Recurrent Unit. *KST 2021 - 2021 13th International Conference Knowledge and Smart Technology*, 186–190.

Mehta, H., & Passi, K. (2022). Social media hate speech detection using explainable AI. *Algorithms*, *15*(8), 291.

Mohiuddin, K., Welke, P., Alam, M. A., Martin, M., Alam, M. M., Lehmann, J., & Vahdati, S. (2023). Retention Is All You Need. *International Conference on Information and Knowledge Management, Proceedings*, *Nips*, 4752–4758.

Mostafa, G., Ahmed, I., & Junayed, M. S. (2021). Investigation of different Machine Learning Algorithms to determine human sentiment using twitter data. *International Journal of Information Technology and Computer Science*, *13*(2), 38–48.

Mozafari, M., Farahbakhsh, R., & Crespi, N. (2020). A BERT-Based Transfer Learning Approach for Hate Speech Detection in Online Social Media. *Studies in Computational Intelligence*, *881 SCI*, 928–940.

Ohiri, E. (2024). *Feedforward neural networks: everything you need to know*. CudaCompute Blog. https://www.cudocompute.com/blog/feedforward-neural-networks-everything-you-need-to-know?

Or, C., Quintana, C., & Zilio, L. (2018). Challenges in Translation of Emotions in Multilingual User-Generated Content : Twitter as a Case Study. *ArXiv*, *1*(6).

Pereira-Kohatsu, J. C., Quijano-Sánchez, L., Liberatore, F., & Camacho-Collados, M. (2019). Detecting and monitoring hate speech in twitter. *Sensors (Switzerland)*, *19*(21), 1–37.

Powers, D. M. W. (2020). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *International Journal of Machine Learning Technology*, *VI*(10), 37–63.

Qiu, D., & Yang, B. (2022). Text summarization based on multi-head self-attention mechanism and pointer network. *Complex and Intelligent Systems*, *8*(1), 555–567.

Rahman, M. M., Balakrishnan, D., Murthy, D., Kutlu, M., & Lease, M. (2021). An information retrieval approach to building datasets for hate hpeech detection. *ArXiv Computer Science*, *11*(NeurIPS 2021), 1–28.

Rongbo, Z., XU, Z., & IWAIHARA, M. (2020). Multi-Label Text Classification Using Only Label Names. *DEIM Forum*, 1–10.

Sabty, C., Omar, I., Wasfalla, F., Islam, M., & Abdennadher, S. (2021). Data Augmentation Techniques on Arabic Data for Named Entity Recognition.

Procedia CIRP, 292–299.

Saeidi, M., Samuel, S. B., Milios, E., Zeh, N., & Berton, L. (2020). Categorizing Online Harassment on Twitter. In *Communications in Computer and Information Science: Vol. 1168 CCIS*.

Salau, A., & Yesufu, T. K. (2020). *Recent Trends in Image and Signal Processing in Computer Vision*.

Sarkar, D., Zampieri, M., Ranasinghe, T., & Ororbia, A. (2021). FBERT: A Neural Transformer for Identifying Offensive Content. *Findings of the Association for Computational Linguistics, Findings of ACL: EMNLP 2021*, *1*(9), 1792–1798.

Shorten, C., Khoshgoftaar, T. M., & Furht, B. (2021). Text Data Augmentation for Deep Learning. In *Journal of Big Data* (Vol. 8, Issue 1). Springer International Publishing.

Takawane, G., Phaltankar, A., Patwardhan, V., Patil, A., Joshi, R., & Takalikar, M. S. (2023). Language augmentation approach for code-mixed text classification. *Natural Language Processing Journal*, *5*(November), 100042.

Tra, V., Duong, B. P., & Kim, J. M. (2019). Improving diagnostic performance of a power transformer using an adaptive over-sampling method for imbalanced data. *IEEE Transactions on Dielectrics and Electrical Insulation*, *26*(4), 1325–1333.

Vasiu, M. A., & Potolea, R. (2020). Enhancing Tokenization by Embedding Romanian Language Specific Morphology. *Proceedings - 2020 IEEE 16th International Conference on Intelligent Computer Communication and Processing, ICCP 2020*, 243–250.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*, *2017-Decem*(Nips), 5999–6009.

Verma, A., Singh, A., Bihari, A., Tripathi, S., Agrawal, S., Pandey, S. K., & Verma, S. (2023). Identification of Hate Speech on Social Media using LSTM. *GMSARN International Journal*, *17*(4), 468–474.

Wang, C., Nulty, P., & Lillis, D. (2020). A Comparative Study on Word Embeddings in Deep Learning for Text Classification. *ACM International Conference Proceeding Series*, 37–46.

Wu, X., Lv, S., Zang, L., Han, J., & Hu, S. (2019). Conditional BERT contextual augmentation. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11539 LNCS*(12), 84–95.

Xia, M., Kong, X., Anastasopoulos, A., & Neubig, G. (2020). Generalized data augmentation for low-resource translation. *ACL 2019 - 57th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference*, 5786–5796.

Yi, D., Ahn, J., & Ji, S. (2020). An effective optimization method for machine learning based on ADAM. *Applied Sciences (Switzerland)*, *10*(3).

Zhang, Y., Wen, J., Yang, G., He, Z., & Wang, J. (2019). Path loss prediction based on machine learning: Principle, method, and data expansion. *Applied Sciences (Switzerland)*, *9*(9).

Lampiran 1. Hasil *testing* data ujaran kebencian *multi label*

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43
0	4419	69	1	13	1	9	1	17	0	0	4	60	1	0	0	0	0	3	0	0	0	11	2	0	0	3	0	0	4	2	0	0	1	0	28	1	9	0	3	1	0	0	0	1
1	103	1269	0	0	0	1	0	0	0	0	0	7	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	1	0	0	0	0	50	20	9	0	0	0	0	0	0	0
2	1	0	57	6	5	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	28	1	3	109	0	6	0	4	0	0	0	25	0	0	0	1	0	0	0	0	0	5	1	0	0	0	0	0	0	0	0	0	0	0	2	1	0	0	0	0	0	0	0	0
4	1	0	0	0	75	3	1	2	0	0	0	3	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	14	4	0	1	5	110	0	1	0	0	0	6	0	0	0	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
6	3	2	2	0	0	0	19	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	16	0	0	2	0	0	0	104	0	1	0	6	0	0	0	0	0	4	0	1	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	0	0	0	0	0	1	4	0	21	0	1	0	0	0	0	0	1	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	8	0	0	0	0	0	1	0	0	0	98	12	0	0	2	0	1	0	1	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	46	4	0	9	1	0	0	0	0	0	5	714	0	0	1	2	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	1	0	0	0	0	0
12	0	0	0	1	0	0	0	0	0	0	0	1	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13	1	0	0	0	0	2	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	1	0	0	8	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0
15	5	0	0	0	0	2	0	0	0	0	0	3	0	0	0	37	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	1	2	0	0	1	0	0	0	0	0	6	3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	15	0	0	2	0	0	1	2	0	0	0	7	0	0	0	0	0	35	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	6	2	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
21	38	18	0	0	0	1	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	410	1	0	0	1	0	1	3	0	0	0	0	0	38	0	1	0	1	1	1	0	0	0
22	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0
23	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	13	0	0	0	0	0	0	0	0	0	0	1	0	3	0	0	0	0	0	0	0
24	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	2	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	38	0	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

27	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	1	0	0	0	0	0	0	0	0	0	0	0	0
28	7	3	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	124	1	0	0	0	0	0	0	0	0	1	0	0	0
29	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	9	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	3	0	0	0	0	1	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0
32	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	19	0	0	0	0	0	0	0	0	0	0
33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	1	1	0	0	0	0	0	0	0
34	83	78	1	2	0	0	0	0	0	0	1	22	0	0	0	0	0	0	0	0	0	48	0	4	0	0	0	0	5	0	0	0	0	1	952	9	8	0	1	0	0	0	0
35	7	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	1	10	129	0	1	0	0	0	0
36	17	17	0	0	0	1	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	21	2	141	0	0	0	0	0	0
37	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	2	9	0	0	0	0	0
38	6	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	2	0	0	0	57	0	0	0	0
39	6	0	0	0	0	0	0	2	0	0	0	2	0	0	0	1	0	2	0	0	0	1	0	0	0	0	0	0	6	0	0	0	0	0	7	0	4	0	0	48	1	3	0	2
40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	4	0	0	0
41	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	5	0	0
42	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	2	0
43	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9