TRANSLATOR BAHASA INDONESIA KE BAHASA JAWA NGOKO

Disusun Oleh:

Nama : Dian Dwi Ramadhan

NIM : 17523228

PROGRAM STUDI INFORMATIKA – PROGRAM SARJANA

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

2023

HALAMAN PENGESAHAN DOSEN PEMBIMBING

TRANSLATOR BAHASA INDONESIA KE BAHASA JAWA NGOKO

TUGAS AKHIR

Disusun Oleh:

Nama : Dian Dwi Ramadhan
NIM : 17523228

Yogyakarta, 10 April 2023

Pembimbing,

(Chanifah Indah Ratnasari, S.Kom., M.Kom.)

HALAMAN PENGESAHAN DOSEN PENGUJI

TRANSLATOR BAHASA INDONESIA KE BAHASA JAWA NGOKO

TUGAS AKHIR

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk memperoleh gelar Sarjana Komputer dari Program Studi Informatika – Program Sarjana di Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, 10 April 2023

Tim Penguji

Chanifah Indah Ratnasari, S.Kom., M.Kom. _____

Anggota 1

Zainudin Zukhri, S.T., M.I.T. _____

Anggota 2

Kurniawan Dwi Irianto, S.T., M.Sc. _____

Mengetahui,

Ketua Program Studi Informatika – Program Sarjana

Fakultas Teknologi Industri

Universitas Islam Indonesia

(Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D)

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Dian Dwi Ramadhan

NIM : 17523228

Tugas akhir dengan judul:

TRANSLATOR BAHASA INDONESIA KE BAHASA JAWA NGOKO

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila di kemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung risiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 5 Mei 2023

(Dian Dwi Ramadhan)

HALAMAN PERSEMBAHAN

Sang pencipta

Aku bersyukur dan berterima kasih karena Allah telah memberikan begitu banyak anugerah selama mengerjakan tugas akhir dalam memberikan kesehatan, ilmu yang bermanfaat, akal cerdas dan banyak karunia-karunia-Nya yang telah diberikan kepadaku.

Keluarga

Ibu, ayah serta kakak dan adik saya yang senantiasa memberikan dorongan doa, semangat, dan dukungan baik berupa materi maupun nonmateri.

Sahabat

Yang selalu bertukar cerita bagaimana beratnya menyelesaikan tugas akhir ini, malasnya menyelesaikan ini, bertukar cerita pusingnya mengerjakan ini, bertukar cerita capeknya mengerjakan ini, bahkan suka dan duka dalam melakukan prosesnya. Sehingga rasa kesepian pun tidak ada dan akhirnya kita saling memberi dukungan berupa doa, serta saling menguatkan satu dengan yang lainnya.

Kekasih

Terimakasih senantiasa mendukung, mendengarkan keluh kesah, juga membantu untuk selalu memberikan semangat dan menemani dalam kondisi suka duka pada proses mengerjakan tugas akhir.

Pelanggan setia *canubis store*

Sebagai pelanggan setia yang selalu menanyakan kabar walaupun hanya sekedar itu dari sisi lain mereka juga selalu memberikan dukungan dalam mengerjakan tugas akhir dan selalu memberikan beberapa motivasi semangat.

Musisi

Yang senantiasa menghadirkan lagu untuk didengar dalam menemani proses saat berjalannya pengerjaan tugas akhir.

Diri sendiri

Yang senantiasa terus berjuang dengan segala keterbatasan fasilitas, pengetahuan, dan wawasan yang luas dalam proses penyelesaian tugas akhir.

HALAMAN MOTO

Allah SWT berfirman dalam Qur'an Surat Al Baqarah ayat 286 yang berbunyi:

لَا يُكَلِّفُ اللَّهُ نَفْسًا إِلَّا وُسْعَهَا ۚ لَهَا مَا كَسَبَتْ وَعَلَيْهَا مَا اكْتَسَبَتْ ۗ رَبَّنَا لَا تُؤَاخِذْنَا إِنْ نَسِينَا أَوْ أَخْطَأْنَا ۚ رَبَّنَا وَلَا تَحْمِلْ عَلَيْنَا إِصْرًا كَمَا حَمَلْتَهُ عَلَى الَّذِينَ مِنْ قَبْلِنَا ۚ رَبَّنَا وَلَا تُحَمِّلْنَا مَا لَا طَاقَةَ لَنَا بِهِ ۖ وَاعْفُ عَنَّا وَاغْفِرْ لَنَا وَارْحَمْنَا ۚ أَنْتَ مَوْلَانَا فَانْصُرْنَا عَلَى الْقَوْمِ الْكَافِرِينَ

Artinya:

Allah tidak membebani seseorang melainkan sesuai dengan kesanggupannya. Ia mendapat pahala (dari kebaikan) yang diusahakannya dan ia mendapat siksa (dari kekejian) yang dikerjakannya. (Mereka berdo'a): "Ya Tuhan kami, janganlah Engkau hukum kami jika kami lupa atau kami tersalah. Ya Tuhan kami, janganlah Engkau berikan beban kepada kami beban yang sangat berat sebagaimana Engkau berikan beban kepada orang-orang sebelum kami. Ya Tuhan kami, janganlah Engkau pikulkan kepada kami apa yang tak sanggup kami memikulnya. Berikan maaf kepada kami; ampunilah kami; dan berikanlah keselamatan bagi kami. Engkaulah Penolong kami, maka berikanlah pertolongan kepada kami terhadap kaum-kaum yang kafir".

KATA PENGANTAR

Assalamu alaikum wa rahmatullahi wa barakatuh,

Alhamdulillah, puji serta syukur kehadirat Allah SWT yang telah melimpahkan nikmat, rahmat, dan hidayah-Nya, sehingga tugas akhir dengan judul "*Translator* dari Bahasa Indonesia ke Bahasa Jawa Ngoko" ini dapat diselesaikan. Selawat serta salam kepada Nabi Muhammad SAW beserta keluarga, sahabat serta pengikut beliau hingga akhir zaman.

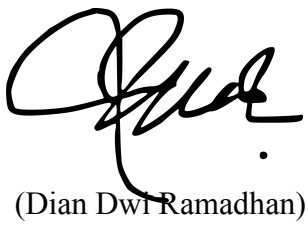
Tugas akhir merupakan salah satu syarat yang harus dipenuhi dalam menyelesaikan jenjang strata satu di Jurusan Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Penulis menyadari dalam penyelesaian tugas akhir ini banyak pihak yang telah memberikan dukungan, arahan, bimbingan, dan bantuan baik materi maupun nonmateri, oleh karena itu penulis ingin mengucapkan ucapan terima kasih kepada:

1. Hari Purnomo, Prof., Dr., Ir., M.T., IPU, ASEAN.Eng selaku Dekan Fakultas Teknologi Industri, Universitas Islam Indonesia, beserta seluruh jajarannya.
2. Dr. Raden Teduh Dirgahayu, S.T., M.Sc. selaku Ketua Jurusan Informatika beserta seluruh jajarannya.
3. Dhomas Hatta Fudholi, S.T., M.Eng., Ph.D. selaku Ketua Program Studi Informatika beserta seluruh jajarannya.
4. Chanifah Indah Ratnasari, S.Kom., M.Kom. selaku dosen pembimbing yang sangat berjasa dalam penyelesaian tugas akhir ini hingga terselesaikan.
5. Seluruh dosen pengajar dan staf prodi Informatika yang telah memberikan bekal ilmu dan bantuannya dalam proses belajar, semoga menjadi amal kebaikan Bapak/Ibu.
6. Semua yang terlibat dalam pembuatan tugas akhir dari mulai hingga selesai.

Penulis menyadari bahwa dengan keterbatasan wawasan serta pengalaman dalam membuat tugas akhir ini masih jauh dari kesempurnaan dan mengharapkan kritik dan saran yang bersifat membangun. Semoga tugas akhir ini dapat memberikan manfaat. Akhir kata, semoga kita semua mendapatkan rahmat dan selalu ada dalam lindungan Allah SWT.

Wassalamu alaikum wa rahmatullahi wa barakaatuh.

Yogyakarta, 10 April 2023



(Dian Dwi Ramadhan)

SARI

Bahasa Jawa merupakan bahasa daerah yang paling banyak digunakan sebagai bahasa dalam berinteraksi dalam kehidupan sehari-hari di Indonesia. Perbedaan bahasa di Indonesia termasuk bahasa Jawa dapat menyebabkan terhambatnya komunikasi antara orang yang satu dengan orang yang lain. Penelitian ini bertujuan sebagai media pembelajaran untuk membantu memudahkan orang dalam berkomunikasi dalam bahasa Jawa termasuk orang yang tidak bisa berbahasa Jawa, dalam hal ini bahasa Jawa ngoko. Pendekatan mesin *translator* menerapkan *Long-short Term Memory* (LSTM) pada *Recurrent Neural Network* (RNN) untuk translasi Bahasa Indonesia ke Bahasa Jawa Ngoko. Hasil translasi yang diperoleh dari penerapan RNN dan LSTM masih jauh dari hasil seharusnya. Model yang dirancang ini memiliki nilai *Bilingual Evaluation Understudy* (BLEU) sebesar 28% pada 1-*grams*, kemudian mengalami penurunan sesuai dengan pertambahan *n-grams* hingga 4-*grams* memiliki nilai *Bilingual Evaluation Understudy* (BLEU) sebesar 5%. Rata-rata nilai *Bilingual Evaluation Understudy* (BLEU) yang dihasilkan sebesar 15%. Model yang dibuat ini merupakan hasil pada dan awalnya belum mampu untuk melakukan translasi secara akurat, sehingga dilakukan remodel terbaru dengan hasil perhitungan nilai (BLEU) memiliki peningkatan hingga 66% dari hasil yang telah dilakukan menggunakan model sebelumnya. Peningkatan n-*gram*s yang didapatkan yaitu 87% pada 1-*grams*, 83% pada 2-*grams*, 81% pada 3-*grams*, dan 74% pada 4-*grams*. Rata-rata nilai BLEU yang diperoleh sebesar 81%. Model terbaru memiliki peluang 66% untuk mendapatkan translasi dengan lebih baik.

Kata kunci: Mesin translasi, *Recurrent Neural Network* (RNN), *Long-short Term Memory* (LSTM), Bahasa Jawa Ngoko, Bahasa Indonesia, *Bilingual Evaluation Understudy* (BLEU).

GLOSARIUM

TensorFlow	*Library open source* dalam python yang dikembangkan oleh *Google* untuk melakukan perhitungan numerik yang dapat memudahkan dan mempercepat dalam pengimplementasian *Machine Learning*.
Keras	*Interface library* dalam python yang bertujuan menyederhanakan penerapan algoritma *Deep Learning* pada *TensorFlow* dan digunakan untuk tujuan pengembangan dan pengevaluasian model *Deep Learning*.
Korpus	Kumpulan perkataan yang lisan atau berupa tulisan yang digunakan untuk menguji atau mendukung hipotesis tentang struktur dalam suatu bahasa.
Integer	Tipe data bilangan bulat baik yang positif maupun negatif dengan *range* tertentu dalam pemrograman.
Preprocessing	Suatu teknik sebelum melakukan pemrosesan data yang ditugaskan dalam pengubahan data asli yang telah dikumpulkan dari bermacam sumber menjadi suatu data yang lebih bersih agar dapat digunakan untuk pengolahan pada tahap selanjutnya.
Encoder	Merupakan sebuah proses mengubah data menjadi suatu bentuk tertentu.
Decoder	Merupakan proses pengembalian bentuk dari data yang telah diubah melalui proses *encoding*.
GloVe	Algoritma *Unsupervised* untuk memperoleh representasi vektor untuk kata-kata.
Model	Pemrograman komputer yang sering digunakan untuk membuat sebuah prediksi atau mengenali suatu pola dalam data.
Batch size	*Hyperparameter* dalam *Machine Learning* yg mengacu pada jumlah contoh *training* yang digunakan dalam satu iterasi.
Epoch	*Hyperparameter* dalam *Machine Learning* yang menentukan berapa kali algoritma *Deep Learning* bekerja dan belajar melewati seluruh dataset.
Training	Proses pelaksanaan *Machine Learning* dalam mengolah dataset untuk mengoptimalkan algoritma dalam mendapatkan pola atau output tertentu.

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1 Latar Belakang

Bahasa merupakan alat interaksi yang digunakan oleh manusia untuk berkomunikasi satu dengan yang lainnya pada kehidupan sehari-hari. Dalam Kamus Besar Bahasa Indonesia (KBBI), bahasa merupakan sistem simbol atau kata yang memungkinkan digunakan oleh anggota suatu masyarakat untuk mengidentifikasikan diri, berinteraksi, dan bekerja sama (Badan Pengembangan dan Pembinaan Bahasa, 2016). Oleh karena itu, bahasa sebagai alat komunikasi bagi manusia sehingga dapat bersosialisasi dan berinteraksi satu dengan yang lainnya.

Indonesia merupakan salah satu negara yang memiliki berbagai keanekaragaman mulai dari suku dan bahasanya yang sangat bermacam-macam. Indonesia telah memiliki bahasa persatuan, yaitu bahasa Indonesia. Namun, kebanyakan penduduk asal Indonesia masih selalu menggunakan bahasa daerah untuk berinteraksi dan berkomunikasi sehari-hari di lingkungannya. Menurut Na'im dan Syaputra (2010), mendefinisikan pada Sensus Penduduk Indonesia pada tahun 2010 bahasa setiap hari yang digunakan adalah bahasa yang sering digunakan dalam berkomunikasi dan berinteraksi di rumah antar sesama anggota keluarga. Bahasa sehari-hari yang digunakan seseorang tidak selalu dilandaskan pada keturunannya, namun dapat terbentuk karena proses yang ada di lingkungan dan interaksi sosial sekitarnya. Contoh yang terjadi yaitu seseorang dengan garis keturunan asalnya dari kelompok Suku Melayu dapat dianggap sebagai golongan kelompok masyarakat Jawa apabila dalam kesehariannya dalam berinteraksi dan bersosialisasi menggunakan bahasa Jawa, terutama apabila di lingkungan sekitar dan di rumahnya mereka menggunakan bahasa Jawa.

Hasil Sensus Penduduk tahun 2010 diperoleh hasil bahwa 79,45% dari semua populasi penduduk dengan umur lima tahun ke atas berkomunikasi dan berinteraksi sehari-hari di lingkungan rumah tangga menggunakan bahasa daerah, 19,94% berkomunikasi dan berinteraksi menggunakan bahasa persatuan yaitu bahasa Indonesia, 0,35% berkomunikasi dan berinteraksi menggunakan bahasa asing, dan 0,26% responden tidak memberi jawaban (Na'im et al., 2010). Variasi penggunaan bahasa ini dapat disebut sebagai keanekaragaman bahasa. Ragam bahasa dapat timbul karena masyarakat yang beraneka ragam dengan keseharian dan lingkungan yang bermacam-macam juga adanya kebutuhan dalam pemakaian

bahasa untuk bekerja sama, bersosialisasi, berkomunikasi, dan berinteraksi selaras dengan kebutuhan sosialnya (Devianty, 2017). Dengan demikian, Indonesia memiliki banyak variasi dalam penggunaan bahasa untuk berkomunikasi sehari-hari di lingkungan rumah.

Sebaran wilayah penduduk yang memakai bahasa Jawa dalam percakapan sehari-hari terdapat pada 33 provinsi di Indonesia, mulai dari Provinsi Aceh di Pulau Sumatera sampai dengan Provinsi Papua. Namun, proporsi penduduk berumur lima tahun ke atas terbanyak yang memakai bahasa Jawa ini tinggal di Provinsi Jawa Tengah dibandingkan Provinsi Jawa Timur (Na'im et al., 2011). Dengan demikian, bahasa Jawa menjadi bahasa daerah yang sangat umum dipakai di Indonesia.

Bahasa Jawa merupakan bahasa daerah yang memiliki tingkat tutur yang menunjukkan hubungan antara orang yang melakukan komunikasi. Menurut Poedjosoedarmo et al. (2013), bahasa Jawa merupakan bahasa yang memiliki gaya tertentu dalam menunjukkan tingkah laku hubungan yang bermacam-macam karena terdapat perbedaan status sosial di masyarakat. Adanya golongan tertentu di masyarakat yang harus dihormati dan ada suatu kelompok yang dapat dihadapi dengan lazim. Oleh karena itu, percakapan yang menggunakan bahasa Jawa perlu menyesuaikan dengan lawan bicara.

Bahasa Jawa memiliki keragaman tingkatan tutur bahasa yang menunjukkan sikap dan hubungan orang-orang yang berkomunikasi. Menurut Poedjosoedarmo et al. (2013), tiga tingkatan tutur bahasa Jawa meliputi, pertama, tingkat tutur Ngoko yang mencerminkan tidak ada jarak dalam hal status sosial antara orang yang melakukan percakapan, tidak ada rasa segan, dan umumnya percakapan penuh keakraban. Kedua, tingkat tutur Krama yang mencerminkan adanya sikap sopan santun, adanya rasa segan, atau melakukan percakapan dengan orang yang berpangkat, atau orang yang belum dikenal di antara orang-orang yang melakukan komunikasi. Ketiga, tutur Madya untuk tingkat kelompok tata bahasa menengah antara tutur Ngoko dan tutur Krama. pada tingkat ini, masih memperlihatkan adanya sikap sopan santun, namun sedang-sedang saja. Dengan demikian, keragaman tutur Bahasa Jawa ini dapat menunjukkan sikap, hubungan, dan perbedaan status sosial antara orang yang melakukan komunikasi.

Tutur Ngoko merupakan tingkatan bahasa Jawa yang paling sering dipakai dalam percakapan sehari-hari. Penggunaan tutur Ngoko menunjukkan adanya kesamaan derajat antara pelaku yang berkomunikasi, menunjukkan adanya keakraban, dan persahabatan yang setara. Meskipun demikian, tutur Ngoko ini layak digunakan dalam percakapan antara orang yang berstatus sosial atau memiliki pangkat/jabatan yang lebih tinggi kepada lawan bicara

yang berstatus lebih rendah, misalnya ayah kepada anak, guru kepada murid, pimpinan kepada anggota, dan lainnya.

Berdasarkan uraian di atas, bahasa Jawa menjadi bahasa daerah yang paling sering dipakai karena tersebar di semua provinsi di Indonesia. Bahasa Jawa tutur Ngoko menjadi tutur bahasa Jawa yang teramat banyak dan paling sering dipakai di kalangan penduduk. Hal ini karena tutur Ngoko identik dengan keakraban, persahabatan, dan tidak mengenal tingkatan jabatan dan pangkat pada orang-orang yang menggunakannya.

Sehubungan dengan banyaknya pengguna bahasa Jawa dan tutur Ngoko yang merupakan tingkatan bahasa yang paling umum dan banyak digunakan, maka perlu dilaksanakan penelitian untuk membuat mesin translasi dari bahasa Indonesia ke bahasa Jawa tutur Ngoko untuk menjembatani orang yang belum memahami bahasa Jawa sekaligus sebagai media pembelajaran. Bahasa Indonesia menjadi bahasa yang akan diterjemahkan, karena bahasa Indonesia adalah bahasa persatuan, dan bahasa kedua terbanyak yang digunakan di Indonesia setelah bahasa Jawa.

Pendekatan mesin translator dapat dilakukan dengan *Neural Machine Translation* (NMT) yang merupakan suatu pendekatan yang belum lama digunakan dalam mesin *translator* yang memakai arsitektur *Recurrent Neural Network* (RNN) dengan arsitektur *encoder* dan *decoder*-nya dan memakai rangkaian korpus paralel sebagai bahan untuk pelatihan data. Hal ini membuktikan penampilan NMT terbukti jauh lebih bagus dibandingkan dengan *Statistical Machine Translation* (SMT) (Abidin et al., 2018). RNN merupakan macam dari arsitektur jaringan saraf tiruan atau biasa disebut *Neural Network* yang rangkaian tindakannya dipanggil secara berkali-kali atau berulang untuk memproses data yang bersambung dan berurutan. RNN telah mengalami kemajuan pada bidang tertentu seperti pemrosesan bahasa alami atau biasa disebut *Natural Language Processing* (NLP), translasi bahasa, sistem pengenalan suara, dan pemrosesan data sekuensial.

1.2 Rumusan Masalah

Menurut latar belakang masalah yang telah dijabarkan di atas, penelitian ini memiliki rumusan masalah sebagai berikut:

a. Bagaimana model translasi bahasa Indonesia ke bahasa Jawa Ngoko dengan menggunakan *Recurrent Neural Network* (RNN) dan *Long Short-Term Memory* (LSTM)?

b. Bagaimana penerapan LSTM pada RNN untuk translasi bahasa Indonesia ke bahasa Jawa Ngoko?
c. Bagaimana nilai *Bilingual Evaluation Understudy* (BLEU) dari model translasi yang dikembangkan?

1.3 Batasan Lingkup Penelitian

Batasan masalah dalam lingkup penelitian ini ditetapkan untuk menjauhkan dari penyebaran pembahasan agar tujuan dari penelitian ini tercapai. Dalam pelaksanaan pembuatan mesin translasi ini lingkup penelitian tidak lebih dari dibatasi pada:

a. Translasi dilakukan hanya dengan satu arah yaitu bahasa Indonesia ke bahasa Jawa Ngoko.
b. Penggunaan data yang dipakai sebanyak 5.000 kalimat bahasa Indonesia dan 5.000 bahasa Jawa Ngoko.
c. Metode yang digunakan untuk translasi adalah *Recurrent Neural Network* (RNN) dan *Long Short-Term Memory* (LSTM) dan pendekatan mesin translator dengan *Neural Machine Translation* (NMT).

1.4 Tujuan Penelitian

Tujuan dari penelitian ini adalah sebagai berikut:

a. Dapat menerjemahkan kata dalam Indonesia ke bahasa Jawa Ngoko dengan menghasilkan terjemahan yang akurat dan menangani hasil urutan kata pada pemrosesan data yang lebih baik.
b. Menangani konteks jangka panjang pada mesin penerjemah, penanganan varian panjang urutan, dan dapat belajar mengenali dalam menerjemahkan kata dengan tepat.
c. Meningkatkan kualitas dan akurasi terjemahan bahasa pada *Bilingual Evaluation Understudy* (BLEU) dengan mempertimbangkan konteks data yang relevan dan mengatasi masalah memori jangka panjang dalam penerjemahan pada suatu kalimat.

1.5 Manfaat Penelitian

Manfaat dari hasil penelitian ini diharapkan dapat berguna sebagai berikut:

a. Sebagai media translasi bahasa Indonesia ke bahasa Jawa ke bahasa Jawa Ngoko, sekaligus sebagai media pembelajaran bahasa Jawa Ngoko.

b. Membuka ide penelitian baru tentang mesin translasi terutama untuk bahasa daerah lain di Indonesia untuk penelitian selanjutnya.

BAB II
TINJAUAN PUSTAKA

2.1 Bahasa Jawa

Bahasa Jawa merupakan bahasa asli bagi masyarakat jawa di daerah Jawa Timur, Jawa Tengah, dan Yogyakarta. Bahasa Jawa juga digunakan di daerah transmigrasi seperti Lampung dan Banten bagian utara. Bahasa Jawa memiliki dialek yang berbeda untuk daerah tertentu seperti Yogyakarta, Solo, Surabaya, dan Tegal. Bahasa Jawa memiliki jenis bahasa resmi dan tidak resmi dalam bentuk bidang bahasa yang menyelidiki bunyi dari setiap bahasa menurut fungsinya, cabang bahasa dengan bentuk katanya, cabang bahasa dengan susunan kalimatnya, maupun kosakata yang bermacam-macam. Tingkatan dalam perkataan atau tutur kata ialah ragam bahasa yang mempunyai perbedaan antara penutur satu dengan penutur lain yang telah dipastikan oleh perbedaan peradaban dari penutur kepada lawan tuturnya (Soepomo, 1975).

Berdasarkan Poedjasoedarma (1979) tingkatan tutur kata dapat dilihat dari perbedaan bentuknya. Penyebutan kata ganti, nomina, verba, dan adjektiva yang berbeda memperlihatkan perbedaan rasa hormat dari orang yang bertutur. Perbedaan pemakaian perkataan atau penuturan yang mempunyai makna sama menampakkan adanya perbedaan susunan tutur yang terdapat dalam bahasa Jawa yaitu susunan tutur lembut yang menyebabkan rasa kehormatan tinggi (susunan tutur Krama), susunan tutur menengah mencerminkan rasa kehormatan biasa (susunan tutur Madya), juga susunan tutur biasa yang menampakkan kehormatan yang rendah (susunan tutur Ngoko) (Poedjasoedarma, 1979). Dapat disimpulkan bahwa susunan dalam bahasa jawa memiliki susunan lembut yaitu tutur Krama, susunan tutur sedang yaitu tutur Madya, dan susunan tutur biasa yaitu tutur Ngoko. Dari setiap susunan bentuk Krama, Madya, dan Ngoko memiliki tiga bentuk tingkatan lagi. Bentuk Krama memiliki tiga susunan yaitu Wredha Krama, Kramantara, dan Muda Krama. Bentuk Madya memiliki dua susunan yaitu Ngoko dan Madya Krama Madyantara. Bentuk Ngoko juga memiliki tiga susunan yaitu Ngoko Lugu, Antya Basa, dan Basa Antya (Wilian, 2006).

Menurut (Sasangka & Wisnu, 2009) prinsip tata cara berbicara dan bertingkah laku dalam rangka menghormati dan menghargai dalam berbahasa Jawa disebut unggah-ungguh. Unggah-ungguh terbagi menjadi dua yaitu Ngoko dan Krama. Unggah-ungguh memiliki

perbedaan yang nyata yang nampak pada kosakata yang telah dirangkai. Unggah-ungguh dalam bahasa Jawa hanya ada dua yaitu Ngoko dan Krama jika didapati bentuk yang lain maka itu hanyalah variasi lain dari jenis Ngoko ataupun Krama (Indrayanto & Yuliastuti, 2015).

2.2 *Neural Machine Translation (*NMT*)*

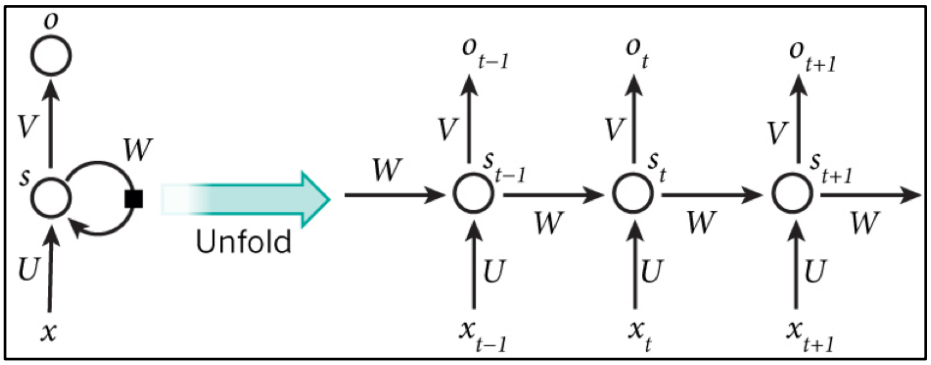
NMT merupakan pendekatan mesin penerjemah canggih yang menggunakan teknik *Neural Network* atau biasa disebut jaringan syaraf dan *Artificial Intelligence* (AI) untuk melatih model saraf dalam memprediksi suatu kumpulan kata pada urutan kalimat yang dapat berupa kalimat lengkap, teks, dan dokumen. NMT hanya menggunakan sebagian kecil dari memorinya untuk melatih semua model secara bersamaan agar dapat memaksimalkan kinerja terjemahannya akan tetapi dapat menghasilkan kualitas terjemahan yang kualitasnya sangat baik.

Neural Network digunakan dalam NMT merupakan metode kecerdasan buatan dalam mengajarkan mesin komputer memproses data yang terinspirasi seperti otak manusia. *Neural Network* dapat memecahkan masalah yang rumit dengan menciptakan mesin adaptif yang dapat memecahkan masalah dan memperbaikinya secara terus menerus. *Machine Learning* dan *Neural Network* memiliki hubungan layaknya manusia seperti *Machine Learning* sebagai tubuhnya dan *Neural Network* sebagai otaknya.

2.3 *Recurrent Neural Network* (RNN)

Jaringan saraf berulang atau biasa disebut RNN merupakan metode *Machine Learning* yang diciptakan sedemikian rupa untuk memproses data yang berurutan atau bersambung. RNN memanfaatkan informasi dari masa lalu untuk diproses sebagai bentuk pembelajaran. RNN juga sangat cocok untuk data yang sifatnya berurutan karena RNN merupakan *Supervised Deep Learning* yang dilatih dengan menggunakan *output* dari langkah sebelumnya digunakan atau dimasukkan sebagai *input* yang sedang berlangsung untuk mengurangi komplektifitas peningkatan parameter dan selalu mengingat *output* sebelumnya dengan cara selalu menggunakan *output* sebagai *input* ke *hidden layer* berikutnya. Karakteristik ini membuat RNN memiliki kemampuan menyimpan ingatan yang memberikan kesempatan suatu sistem untuk memprediksi data dengan akurat dan mengetahui pola data dengan baik. RNN memiliki arsitektur yang dapat melakukan *looping* atau perulangan dari hasil penyimpanan informasi masa lalunya agar informasi dari masa lalu yang digunakan

tetap tersimpan. RNN sederhana memiliki arsitektur yang terdiri dari tiga lapisan atau biasa disebut dengan *layer*, yaitu lapisan *input* berupa *input layer*, lapisan tersembunyi berupa *hidden layer*, dan lapisan *output* berupa *output layer* (Salehinejad et al., 2018).



Gambar 2.1 Proses RNN

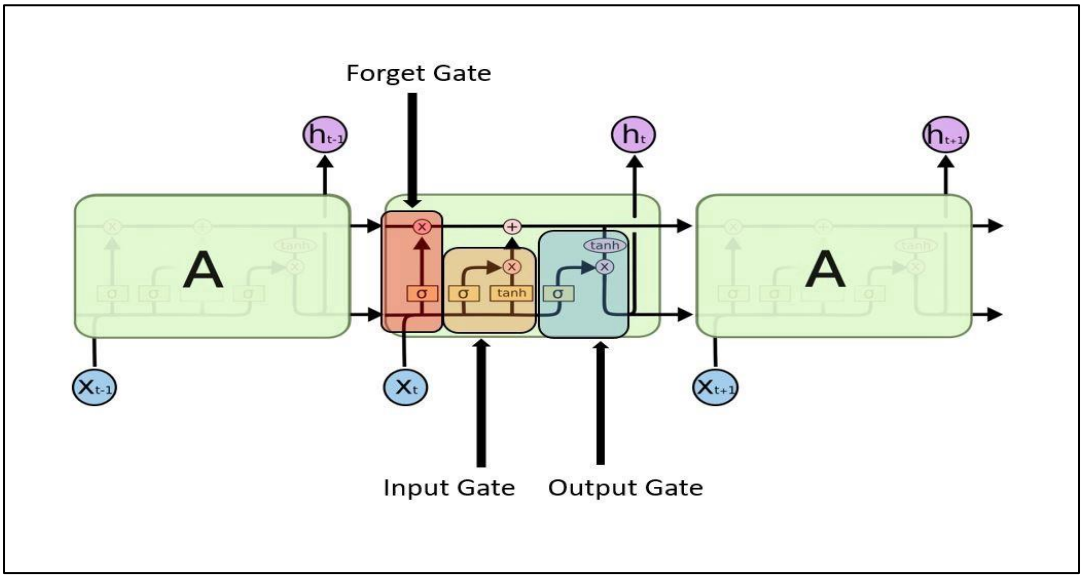
Pada Gambar 2.1 terdapat kotak hitam pada gambar bagian kiri yang menampilkan bahwa RNN pada posisi tertutup dan menampilkan *time delay* dari satu *time step*. Gambar bagian sebelah kanan menampilkan bahwa RNN dengan posisi terbuka menjadi jaringan penuh atau *full network* yang mengakibatkan urutan *sequence* berubah menjadi lebih lengkap. Jika dalam *sequence* memiliki lima kata, maka jaringan akan dibuka untuk setiap kata, sehingga memiliki 5-*layer neural network*. Keterangan simbol dari Gambar 2.1. sebagai berikut.

a. x_t adalah *input* pada *time step*.
b. s_t adalah *hidden state* (memori) pada setiap *time step* t. Perhitungan st menurut dari *input* pada *current state* ($s_t = f(Ux_t + Ws_{t-1})$) dan *hidden state* sebelumnya.
c. o_t adalah *output* pada *step* t.

2.4 *Long Short-Term Memory* (LSTM)

Memori jangka panjang-pendek atau biasa disebut sebagai LSTM yaitu sebuah algoritma *Unsupervised Deep Learning* yang cocok digunakan dalam mesin translasi dengan sistem penyimpanan data yang berguna untuk memprediksi, memproses, dan mengklasifikasikan informasi yang telah lama disimpan dan mampu menghasilkan prediksi lebih akurat dengan informasi yang terbaru. Prediskси merupakan arsitektur dalam LSTM yang bekerja dengan *encoder* dan *decoder* berguna sebagai memungkinkan model untuk mendukung urutan input panjang variabel dan untuk memprediksi atau menampilkan urutan

output panjang varibel. Hasil pengembangan pertama kali dari algoritma LSTM yang dilakukan oleh Hochreiter dan Schmidhuber adalah algoritma dalam LSTM ini mampu menyimpan dan mempertahankan informasi untuk jangka waktu yang panjang. Tantangan dari RNN yaitu permasalahan dalam pemodelan yang memiliki ketergantungan dalam jangka waktu yang panjang, yaitu permasalahan efektivitas gradien (Salehinejad et al., 2018). LSTM merupakan modifikasi dari RNN sehingga dapat menutupi salah satu kekurangan RNN yaitu tidak dapat menyimpan informasi yang sudah berlalu dalam jangka waktu yang lama. LSTM ditemukan dengan tujuan untuk mengatasi permasalahan hilangnya efektivitas gradien (Sherstinsky, 2020). LSTM dapat digunakan untuk memproses, melakukan klasifikasi dan memprediksi suatu kumpulan data yang teratur oleh urutan waktu.



Gambar 2.2 Arsitektur LSTM

Pada Gambar 2.2 di atas, LSTM memiliki tiga *gate* sebagai gerbang dalam LSTM yang merupakan hasil modifikasi RNN, yaitu gerbang input berupa *input gate*, gerbang lupakan berupa *forget gate*, dan gerbang keluaran berupa *output gate*. Penjelasan dari tiap *gate* sebagai berikut.

a. *Input Gate* merupakan gerbang dalam LSTM untuk mencari nilai informasi dari *input* yang akan digunakan untuk dimodifikasi ke dalam memori. Penentuan nilai menerapkan fungsi *sigmoid* dan fungsi *tanh*. Fungsi *tanh* memberi bobot pada tiap nilai berdasarkan tingkat kepentingan nilai informasi tersebut dengan rentang nilai -1 sampai dengan 1.

$$i_t = \sigma(W_i \,.[h_{t-1}, x_t] + b_i) \tag{2.1}$$

$$\tilde{C}_t = \tanh(W_C \,.\, [h_{t-1}, x_t] + b_C) \tag{2.2}$$

b. *Forget Gate* merupakan gerbang dalam LSTM untuk mencari nilai informasi yang harus disimpan dan yang harus dibuang. Penentuan nilai menerapkan fungsi *sigmoid* dengan melihat *state* sebelumnya (h_{t-1}) dan *input* (x_t). Hasil dari perhitungan tersebut akan menghasilkan nilai 0 untuk membuang informasi dan nilai 1 untuk tetap menyimpan informasi.

$$f_t = \sigma(W_f \,.\, [h_{t-1}\,, x_t] + b_f) \tag{2.3}$$

c. *Output Gate* merupakan gerbang dalam LSTM untuk menentukan *output* dan menghasilkan informasi baru berdasarkan informasi dari *input gate* dan *forget gate*. Penentuan nilai menerapkan fungsi *sigmoid* dan fungsi *tanh*. Fungsi *tanh* memberi bobot pada tiap nilai berdasarkan tingkat kepentingan nilai informasi tersebut dengan rentang nilai -1 sampai dengan 1, kemudian dikalikan dengan *output* dari perhitungan fungsi *sigmoid*.

$$o_t = \sigma(W_o \,.\, [h_{t-1}\,, x_t] + b_o) \tag{2.4}$$

$$h_t = o_t \,.\, \tanh(C_t) \tag{2.5}$$

2.5 *Word Embedding*

Word Embedding adalah teknik pengolahan bahasa alami atau biasa disebut *Natural Language Processing* (NLP) yang digunakan untuk merepresentasikan kata-kata dalam bentuk numerik, sehingga dapat diolah oleh model *machine learning*. Representasi numerik ini disebut dengan vektor *embedding*. Vektor *embedding* merepresentasikan kata-kata dalam ruang vektor multidimensi, di mana kedekatan antara vektor merepresentasikan kedekatan makna antara kata-kata tersebut. Semakin dekat jarak antara dua vektor, semakin mirip makna dari kata-kata yang direpresentasikan oleh kedua vektor tersebut.

Teknik *word embedding* menggunakan model *neural network*, seperti *Word2Vec*, *GloVe*, atau *fastText*, untuk mempelajari representasi vektor *embedding* kata-kata dari dataset teks besar. Model ini melakukan *training* dengan mengekstraksi pola dan relasi antar kata dari konteks kata tersebut dalam kalimat atau dokumen. *Word embedding* digunakan dalam berbagai aplikasi NLP seperti klasifikasi dokumen, pemrosesan bahasa alami, analisis sentimen, dan penerjemahan mesin. Keuntungan dari penggunaan *word embedding* adalah dapat merepresentasikan makna dan konteks kata dengan lebih baik daripada representasi kata-kata dalam bentuk *one-hot vector* yang hanya merepresentasikan keberadaan atau ketiadaan kata-kata pada suatu dokumen atau kalimat.

2.6 *Confusion Matrix*

Confusion matrix adalah sebuah metode untuk mengevaluasi kinerja sebuah model klasifikasi dengan membandingkan hasil prediksi model dengan nilai sebenarnya. *Confusion matrix* merupakan salah satu cabang dari ilmu kecerdasan buatan atau biasa disebut *Artificial Intelligence* (AI) sebagai acuan evaluasi performa algoritma dari *Machine Learning* (MT). *Confusion matrix* menghitung berapa banyak sampel yang diklasifikasikan dengan benar dan yang salah dalam masing-masing kelas. *Confusion matrix* terdiri dari empat jenis matriks evaluasi, yaitu *true positive* (TP), *false positive* (FP), *true negative* (TN), dan *false negative* (FN).

TP adalah jumlah data positif yang diprediksi benar, sedangkan FP adalah jumlah data negatif yang diprediksi salah. TN adalah jumlah data negatif yang diprediksi benar, sedangkan FN adalah jumlah data positif yang diprediksi salah. Dari confusion matrix ini, peneliti dapat menghitung berbagai matriks evaluasi klasifikasi seperti akurasi, presisi, *recall*, dan F1-*score*.

Akurasi mengukur seberapa akurat model dalam memprediksi kelas yang benar, sedangkan presisi mengukur seberapa banyak data positif yang diprediksi benar. *Recall* atau *sensitivity* mengukur seberapa banyak data positif yang diidentifikasi secara benar dari total data positif yang seharusnya teridentifikasi, sedangkan F1-*score* adalah rata-rata harmonik antara presisi dan *recall*. *Confusion matrix* berguna untuk membantu peneliti memahami kinerja model klasifikasi dan dapat membantu peneliti untuk meningkatkan model tersebut. Rumus yang digunakan dalam matriks evaluasi klasifikasi dari *confusion matrix* sebagai berikut.

$$\text{Akurasi} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}} \tag{2.6}$$

$$\text{Presisi} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.7}$$

$$Recall = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.8}$$

$$\text{F1}score = \frac{(2\,.\,Recall\,.\,\text{Presisi})}{Recall + \text{Presisi}} \tag{2.9}$$

2.7 *Bilingual Evaluation Understudy* (BLEU)

Pengujian sebuah mesin penerjemah banyak dilakukan menghitung nilai dari BLEU. Metode BLEU berguna untuk mengevaluasi mesin translasi secara cepat, murah, dan tidak bergantung pada sebuah bahasa, yang memiliki korelasi tinggi dengan hasil evaluasi dari manusia (Papineni et al., 2002). BLEU banyak digunakan sebagai parameter evaluasi di bidang mesin penerjemah, pemberian teks singkat pada gambar, peringkas teks, dan pengenalan suara. Perhitungan dilakukan dengan cara mencocokkan panjang sebuah kalimat *output* dari mesin penerjemah dengan sumber acuan dari terjemahan. Rumus perhitungan BLEU membutuhkan perhitungan *brevity penalty* yang dapat dilihat pada perbandigan sebagai berikut.

$$\text{BLEU} = \text{BP}\,.\exp\left(\sum_{n=0}^{N} w_n \log P_n\right) \tag{2.10}$$

$$\text{BP} = \begin{cases} 1, & \text{c} > \text{r} \\ \exp\left(1 - \frac{r}{c}\right), & \text{c} \leq \text{r} \end{cases} \tag{2.11}$$

Variabel BP adalah *brevity penalty*, w_n merupakan bobot pasti yang dijumlah menjadi satu, P_n adalah perhitungan *n-gram* dengan N, r yang paling tinggi adalah panjang korpus referensi, dan c adalah panjang terjemahan.

2.8 Penelitian Terdahulu

Penelitian mengenai mesin penerjemah untuk bahasa Indonesia–bahasa Sunda yang dilakukan oleh Fauziyah et al. (2022) menggunakan 3.496 kumpulan kalimat berupa korpus bahasa Sunda dan bahasa Indonesia. Metode yang digunakan adalah *Recurrent Neural Network* dengan melalui 4 pengujian. Pengujian pertama diperoleh hasil nilai optimal oleh GRU akurasi sebesar 99,17% dengan arsitektur model variasi dari *Recurrent Neural Network* (RNN). Pengujian kedua memperoleh nilai optimal dengan nilai akurasi 99,94% oleh model dengan memakai *Attention.* Pengujian ketiga memperoleh hasil optimal dengan nilai akurasi 99.35% dengan perbandingan model optimasi Adam. Pengujian terakhir yaitu menggunakan *Bilingual Evaluation Understudy* (BLEU) skor *brevity penalty* 0,929 dan mendapatkan hasil nilai yang optimal dengan skor BLEU 92,63%.

Penelitian yang dilakukan Abidin et al. (2018) mengenai mesin penerjemah untuk kalimat Bahasa Lampung dan Bahasa Indonesia menggunakan sebanyak 3.000 paralel korpus Bahasa Lampung dengan variasi bahasa *api* dan Bahasa Indonesia. Pengujian dengan pendekatan *Neural Machine Translation* (NMT) dengan metode RNN memakai 25 korpus majemuk dengan OOV, 25 korpus majemuk tanpa OOV, 25 korpus utuh dengan OOV, dan 25 korpus utuh tanpa *out-of-vocabulary* (OOV). Hasil pengujian metode yang digunakan menunjukkan nilai rata-rata BLEU sebesar 51,96%.

Hermanto et al. (2015) melakukan sebuah penelitian tentang mesin translasi untuk Bahasa Inggris–Bahasa Indonesia menggunakan data yang didapat dari BPPT dengan jumlah data 13.076 kalimat, kosakata Bahasa Inggris sebanyak 17.366 kata, dan kosakata Bahasa Indonesia sebanyak 18.371 kata. Metode yang digunakan adalah RNN dengan hasil BLEU mengalami peningkatan nilai sebesar 1,1 dari hasil BLEU sebelumnya yang menggunakan mesin penerjemah statistik menjadi sebesar 24,5 setelah menggunakan RNN.

Bintang et al. (2022) melakukan penelitian terkait *Machine Translation* (MT) berbasis *Recurrent Neural Network* (RNN) yang lebih baik daripada MT berbasis statistik dalam kasus bahasa dengan sumber daya terbatas dan pengguna aktif yang sedikit. Hasil penelitian menunjukkan bahwa MT berbasis RNN menghasilkan hasil terbaik dan dapat diimplementasikan untuk bahasa Kawi yang juga merupakan bahasa dengan sumber daya terbatas ke dalam Bahasa Indonesia. Penelitian yang dilakukan juga menghasilkan dataset korpus paralel Kawi-Indonesia yang digunakan dalam pengembangan MT berbasis neural dengan menggunakan *microframework Flask*. Tujuan penelitian yang dilakukan adalah merancang model MT berbasis RNN seperti *Simple* RNN dan *Bidirectional* RNN untuk

menerjemahkan teks dari bahasa Kawi ke bahasa Indonesia. Berdasarkan hasil penelitian ini model RNN menggunakan dataset Kawi-Indonesia dengan paralel korpus sebanyak 1.086 baris kalimat dapat digunakan untuk pengembangan sistem MT kedepannya, namun dengan jumlah yang masih terbatas masih perlu bantuan ahli untuk menambah korpus dataset tersebut. *Bilingual Evaluation Understudy* (BLEU) *score* yang diperoleh memiliki nilai yang berada pada kisaran 4,6% pada 1-*grams* dan bertambah sesuai penambahan *n-grams* hingga 4-*grams* di angka 32,1% dengan rata-rata skor BLEU sebesar 20,43%. Model *bidirectional* RNN memiliki nilai yang berada pada kisaran 2,8% pada 1-*grams* dan bertambah sesuai penambahan n-*grams* hingga 4-*grams* di angka 29,6% dengan rata-rata skor BLEU sebesar 17,6%. Meskipun masih di bawah tingkat ahli manusia dan mesin translasi berbasis sumber daya tinggi seperti Google (dengan skor BLEU minimal 60), penggunaan *microframework Flask* mempercepat proses pengembangan MT untuk bahasa Kawi ke bahasa Indonesia. Model RNN memiliki performa yang sedikit lebih baik dibandingkan dengan model *bidirectional* RNN. Namun, keduanya masih jauh dari skor BLEU yang dapat dicapai oleh translator manusia maupun mesin seperti Google *Translate* agar dapat memahami konteksnya.

Aristyanto dan Kurniawan (2021) melakukan sebuah penelitian tentang mesin penerjemah untuk membantu memahami berbagai bahasa di seluruh dunia dengan pendekatan yang dapat digunakan dalam pembuatan mesin penerjemah yaitu *Neural Machine Translation* (NMT). NMT menggunakan metode baru dalam proses penerjemahan mesin yang melibatkan penggunaan arsitektur *Recurrent Neural Network* (RNN) pada *encoder* dan *decoder*, serta metode NMT berbasis *attention* yang menggunakan dua lapisan *Long-short Term Memory* (LSTM). Penelitian yang dilakukan adalah untuk memodifikasi arsitektur model NMT dengan menambahkan lapisan agar dapat meningkatkan kinerja model, terutama dalam mengatasi *overfitting* dan juga melibatkan simulasi pada beberapa *hyperparameter Neural Network*, seperti ukuran *batch*, *epoch*, *optimizer*, fungsi aktivasi, dan tingkat *dropout*. Model yang dihasilkan kemudian diuji menggunakan dataset yang berbeda, termasuk dalam penyelesaian masalah *overfitting* yang menghambat generalisasi model terhadap data uji lainnya. Terdapat banyak faktor yang mempengaruhi kinerja dalam mengatasi *overfitting* pada NMT, termasuk ukuran *hyperparameter* dan arsitektur model yang digunakan. Pada penelitian ini, digunakan 20.000 sampel data dari bahasa Inggris-Jerman untuk melatih model dan mencari kombinasi parameter Neural Network terbaik, sedangkan bahasa Inggris-Spanyol digunakan sebagai data uji untuk menguji model dan parameter tersebut. Metode

NMT menggunakan korpus paralel sebagai *input*. Penelitian ini bertujuan untuk mengembangkan arsitektur model NMT dan melakukan simulasi pada masing-masing *hyperparameter Neural Network* dan ukuran arsitektur modelnya, termasuk ukuran *batch*, *epoch*, *optimizer*, *dropout rate*, dan penggunaan fungsi aktivasi *SoftMax*. Hasil penelitian menunjukkan bahwa model pengembangan dapat mengatasi masalah *overfitting* yang ada pada model sebelumnya, dengan tingkat akurasi sebesar 71,95% dan skor BLEU sebesar 43,11% pada data uji yang berbeda.

Penelitian yang dilakukan oleh Gunawan et al. (2021) membahas tentang penggunaan mesin penerjemah untuk menerjemahkan antara bahasa Lampung dan bahasa Indonesia. Data korpus yang diperoleh melalui metode observasi dengan mengumpulkan sekitar 3.000 pasang kalimat paralel dalam bahasa Lampung dengan variasi dialek dan bahasa Indonesia. Pendekatan yang digunakan dalam pengujian adalah *Neural Machine Translation* (NMT) dengan menggunakan metode *Recurrent Neural Network* (RNN) yang dibangun berdasarkan kerangka model *sequence to sequence* yang terdiri dua jaringan saraf tiruan, yaitu *encoder* dan *decoder*. Analisis dilakukan dengan membandingkan akurasi mesin penerjemah menggunakan dua mekanisme *attention* yang berbeda. Hasilnya dievaluasi berdasarkan pengujian otomatis menggunakan *matrix* BLEU dan juga melibatkan penilaian oleh ahli bahasa dengan menggunakan 25 korpus majemuk dengan dan tanpa kata yang tidak dikenal *out-of-vocabulary* (OOV), serta 25 korpus lengkap dengan dan tanpa kata OOV. Hasil pengujian menunjukkan bahwa rata-rata nilai BLEU adalah 51,96%, sedangkan penilaian manual oleh dua ahli bahasa mencapai 78,17% dan 72,53%. Dapat disimpulkan bahwa kualitas terjemahan manual memiliki kualitas yang setara dengan terjemahan yang dihasilkan oleh mesin penerjemah.

Tabel 2.1 Penelitian Sebelumnya

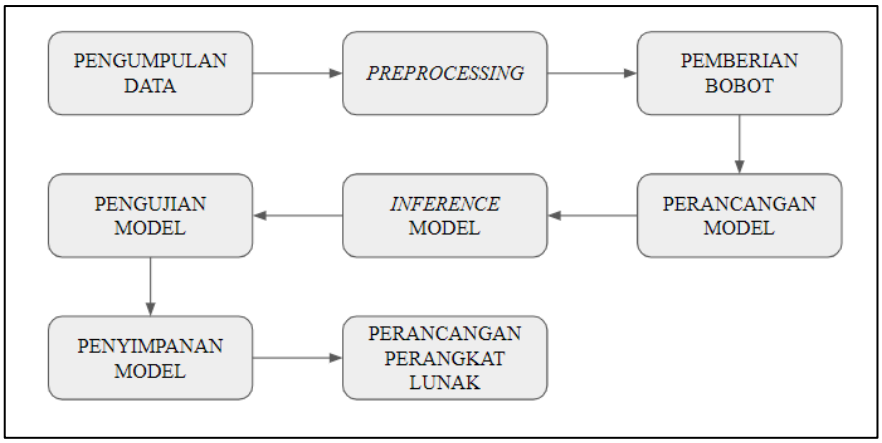
No	Nama Peneliti	Dataset	Metode Penelitian	Hasil
1	Fauziyah et al	3.496 kumpulan Bahasa Sunda dan Bahasa Indonesia	*Recurrent Neural Network*	Menghasilkan nilai optimal BLEU sebesar 92,63% dengan *brevity penalty* 0,929
2	Abidin et al	3.000 paralel korpus Bahasa Lampung dengan variasi bahasa *api* dan Bahasa Indonesia	*Recurrent Neural Network*	Menghasilkan nilai rata-rata BLEU sebesar 51,96%

3	Hermanto et al	BPPT dengan jumlah data 13.076 kalimat, 17.366 kosakata Bahasa Inggris, dan 18.371 kosakata Bahasa Indonesia	*Recurrent Neural Network*	Menghasilkan peningkatan nilai BLEU sebesar 1,1 menjadi 24,5 setelah menggunakan RNN
4	Bintang et al	Kawi-Indonesia dengan paralel korpus sebanyak 1.086 baris kalimat	*Recurrent Neural Network*	Menghasilkan rata-rata nilai BLEU sebesar 20,43% dari *Simple* RNN dan 17,6% nilai dari *Bidirectional* RNN
5	Aristyanto dan Kurniawan	20.000 sampel data dari bahasa Inggris-Jerman dan Inggris-Spanyol	*Recurrent Neural Network* dan *Long-short Term Memory*	Menghasilkan peningkatan akurasi sebesar 71,95% dan skor BLEU sebesar 43,11%
6	Gunawan et al	3.000 pasang kalimat paralel dalam bahasa Lampung dengan variasi dialek dan bahasa Indonesia.	*Recurrent Neural Network*	Menghasilkan rata-rata nilai BLEU adalah 51,96%, sedangkan penilaian manual oleh dua ahli bahasa mencapai 78,17% dan 72,53%

BAB III
METODOLOGI PENELITIAN

3.1 Tahapan Penelitian

Setelah pengumpulan data dan dilakukan kajian pustaka kemudian dibangun sistem pada penelitian ini. Penelitian ini memiliki enam tahapan yaitu tahap pengumpulan data, tahap *preprocessing*, tahap pemberian bobot, perancangan model, *inference* model, dan pengujian. Tahapan penelitian yang dibuat dapat dilihat pada Gambar 3.1.



Gambar 3.1 Tahapan penelitian

Tahap pertama yang dilaksanakan pada penelitian ini adalah pengumpulan data translasi dari bahasa Indonesia ke bahasa Jawa Ngoko. Perolehan data diambil dari forum komunitas penelitian *Natural Language Processing* (NLP) untuk bahasa Indonesia dan bahasa daerah. Data selanjutnya dilakukan *preprocessing* yang terdiri dari *case folding,* menghilangkan *punctuation* (tanda baca), penambahan *tag* (penanda) awal dan akhir dari suatu kalimat, *tokenizing*, dan *padding*. Data yang sudah melalui tahap *preprocessing* kemudian dilakukan pemberian bobot pada tiap kata menggunakan *word embeddings* untuk kata berbahasa Indonesia dan pembobotan menggunakan *One hot vector* untuk kata berbahasa Jawa Ngoko. Data yang sudah memiliki bobot tiap kata kemudian dilakukan perancangan model menggunakan *Recurrent Neural Network* (RNN) atau jaringan saraf berulang dan *Long Short-Term Memory* (LSTM) atau memori jangka panjang-pendek. Setelah perancangan model selesai kemudian dilakukan pelatihan dataset kemudian menguji dataset tersebut dengan sistem pengujian otomatis yaitu *Bilingual Evaluation Understudy*

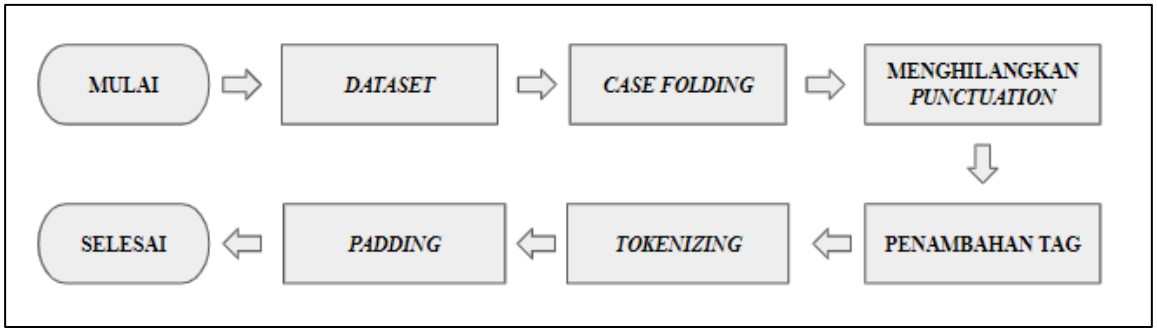
(BLEU) dan melakukan *inference encoder decoder model* yang kemudian model akan disimpan untuk digunakan output dalam tampilan web. Selanjutnya hasil klasifikasi tahap pengujian dengan menggunakan *confusion matrix* untuk melihat hasil performa dari sistem yang dibuat.

3.2 Pengumpulan Data

Perolehan data diambil dari forum komunitas penelitian *Natural Language Processing* (NLP) untuk bahasa Indonesia dan bahasa Jawa Ngoko. Data untuk translasi bahasa Indonesia ke bahasa Jawa Ngoko yang dipilih adalah data yang dibuat oleh Herry Sujaini dengan nama projek data tersebut adalah Korpus Nusantara. Data Korpus Nusantara memiliki banyak data translasi dari bahasa Indonesia ke bahasa daerah seperti bahasa Batak, Dayak, Melayu, Melayu, Bugis, dan bahasa Jawa. Jumlah data translasi bahasa Indonesia ke bahasa Jawa Ngoko yang tersedia terdapat 6.059 data. Pemakaian data yang akan digunakan pada penelitian ini tidak akan memakai seluruh data translasi bahasa Indonesia ke bahasa Jawa, akan tetapi hanya akan mengambil sebanyak 5.000 data pertama dari total 6.059 data. Alasan peneliti menggunakan data yang terbatas ini adalah menyesuaikan fasilitas yang dimiliki oleh peneliti agar program dapat berjalan dengan lancar.

3.3 *Preprocessing*

Data yang didapat kemudian tidak digunakan langsung, akan tetapi dilakukan *preprocessing* terlebih dahulu. Tampilan alur proses *preprocessing* yang berlangsung di tahap ini dapat dilihat pada Gambar 3.2.



Gambar 3.2 Tahapan *preprocessing*

Pada tahapan *preprocessing,* data akan diolah sehingga data siap digunakan dalam proses selanjutnya. Pelaksanaan *preprocessing* ini dilakukan agar memperoleh hasil dengan

semaksimal mungkin dan konsisten. Tahapan *preprocessing* pada penelitian ini disertakan pada rincian berikut.

a. *Case Folding*

Teknik *case folding* yaitu mengubah bentuk data mentah ke bentuk yang seragam. Dalam tahap ini, semua bentuk huruf dalam data akan dibuat menjadi huruf kecil untuk mempermudah mesin dalam melakukan *preprocessing*.

b. Menghilangkan *punctuation* (tanda baca)

Penghilangan tanda baca yang ada dalam data seperti dolar, pagar, petik, tanda seru, titik, koma, dan tanda baca lainnya dari data sehingga data bersih dari tanda baca.

c. Penambahan *tag* (penanda)

Penambahan *tag* pada tiap kalimat data sebagai penanda awal dan akhir dari kalimat. Penambahan *tag* berupa penambahan '<*sos*>' sebagai penanda awal dari kalimat dan '<*eos*>' sebagai penanda akhir dari kalimat. Tanda *tag* ini berguna pada tahap fungsi translasi atau ketika mesin translasi melakukan proses penerjemahan berlangsung dan ketika output penandaan *tag* ini tidak akan ditampilkan.

d. *Tokenizing*

Teknik *Tokenizing* dilakukan dalam *preprocessing* yang berguna untuk memisahkan kalimat menjadi tiap kata. Tiap kata yang sudah dipisah biasa disebut sebagai *token*. Tiap token kemudian diberi indeks berupa angka *integer* yang kemudian dilakukan vektorisasi dari kalimat yang terdiri dari kata menjadi s*equence* yang berbentuk matriks.

e. *Padding*

Padding adalah proses penyamaan panjang data dari seluruh dataset. *Padding* dilakukan agar panjang tiap data menjadi sama panjang dengan data yang terpanjang. *Padding* dapat dilakukan dengan penambahan angka nol (0) pada data *sequence*. Data *sequence* untuk *encoder* maka akan di*padding* pada bagian awal *sequence*, sedangkan data *sequence* untuk *decoder* akan di*padding* pada bagian akhir *sequence*.

3.4 *Word Embeddings*

Data yang sudah dilakukan *preprocessing* kemudian dilakukan pembobotan dengan *word embeddings*. *Word embeddings* adalah metode *Natural Language Processing* (NLP) untuk memetakan tiap kata dalam kamus hasil tokenisasi ke dalam nilai vektor. Nilai vektor tersebut kemudian dapat digunakan untuk melakukan prediksi kata, semantik kata, dan menentukan kemiripan kata. Sudah terdapat penyedia nilai vektor tiap kata pada word embeddings. Penyedia tersebut menghitung nilai vektor dari setiap kata, sehingga dalam penelitian ini cukup menggunakan data tersebut. Penyedia nilai vektor *word embeddings* adalah *GloVe* (*Global Vectors*) dan word2vec. Pada tahap ini, data Bahasa Indonesia akan dilakukan *word embeddings* dengan menyematkan nilai vektor dari penyedia *word embeddings* ke kata dalam kamus hasil tokenisasi.

3.5 *One Hot Vector Decoder*

One hot vector decoder adalah sebuah model *machine learning* yang digunakan untuk memprediksi label atau kelas dari data. Model ini bekerja dengan mengubah data input ke dalam bentuk vektor yang terdiri dari nilai 0 dan 1 (*one-hot vector*), di mana setiap nilai merepresentasikan keberadaan atau ketiadaan dari suatu fitur atau atribut.

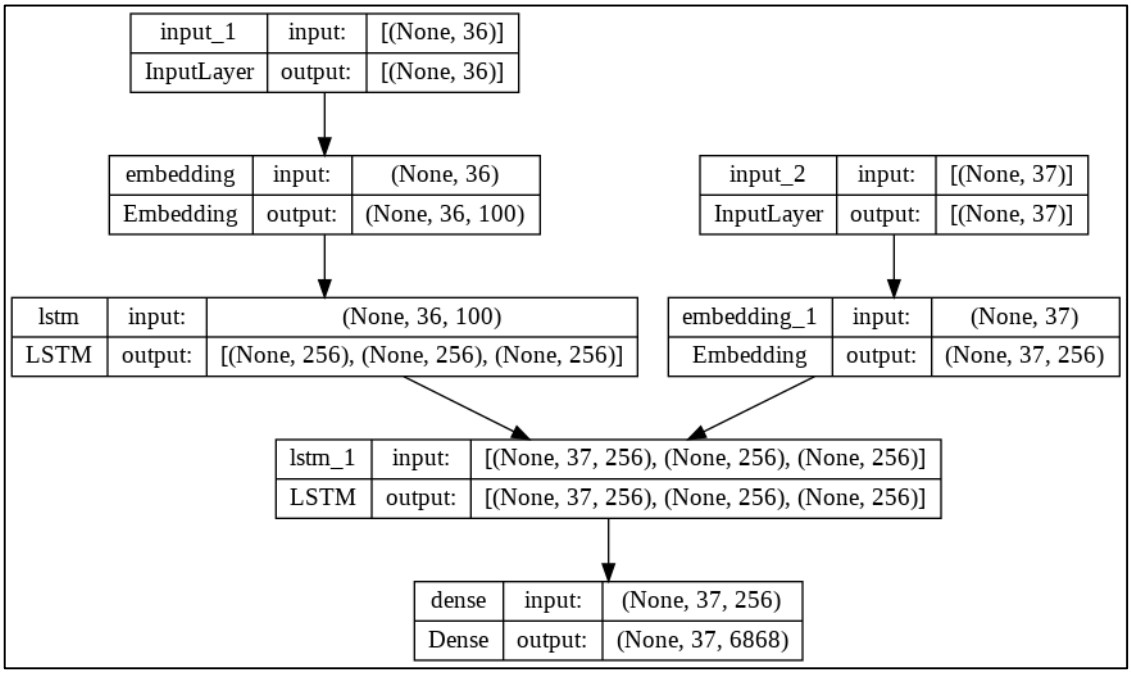
Setelah data input diubah menjadi *one-hot vector*, model akan mempelajari hubungan antara setiap fitur dengan label atau kelas yang diinginkan. Model ini menggunakan algoritma *supervised learning* dan memanfaatkan teknik *softmax* untuk memperoleh probabilitas kelas yang paling mungkin berdasarkan vektor *input*.

Dalam penggunaannya, *One hot vector decoder* banyak digunakan pada berbagai aplikasi *machine learning* seperti klasifikasi gambar, teks, dan pengenalan suara. Kekurangan dari model ini adalah bahwa ia tidak efektif untuk data yang memiliki dimensi yang sangat besar, data yang kurang terstruktur dan data kurangnya data yang dimiliki (banyak nilai nol).

3.6 Perancangan Model

Penelitian ini menggunakan model *Recurrent Neural Network* (RNN) dan *Long Short-Term Memory* (LSTM). Model yang dibuat memakai arsitektur *seq2seq* sehingga membutuhkan *encoder* dan *decoder*. Pembuatan *encoder* terdiri dari tiga lapisan yaitu *input layer* sebagai lapisan input, *embedding layer* sebagai lapisan embedding, dan LSTM *layer* sebagai lapisan LSTM. Pembuatan *decoder* terdiri dari empat lapisan yaitu *input layer* sebagai lapisan input, *embedding layer* sebagai lapisan *embedding,* LSTM *layer* sebagai

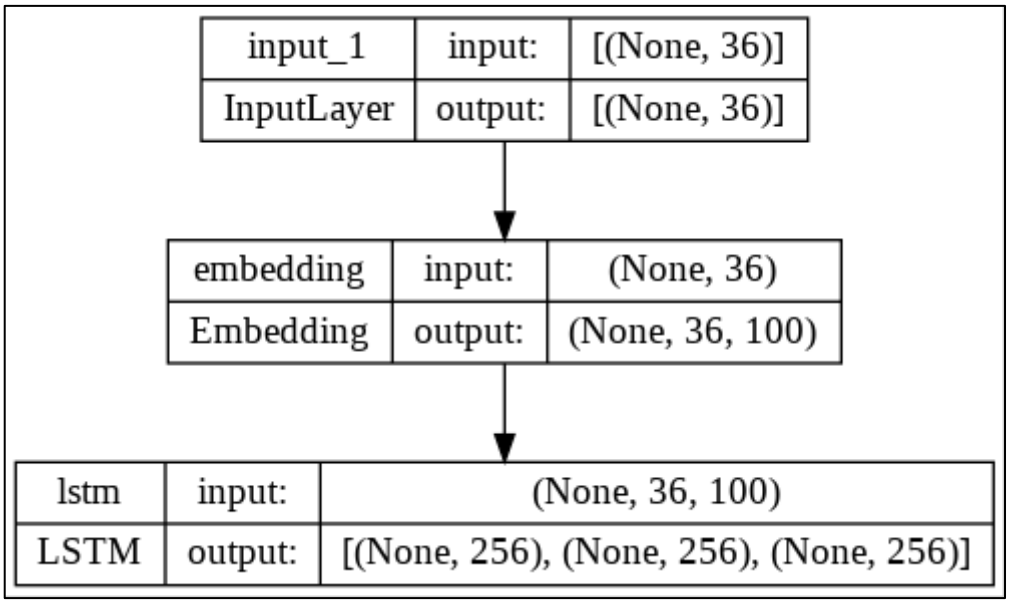
lapisan LSTM, dan *dense layer* sebagai lapisan padat. Rancangan model dapat dilihat pada Gambar 3.3.



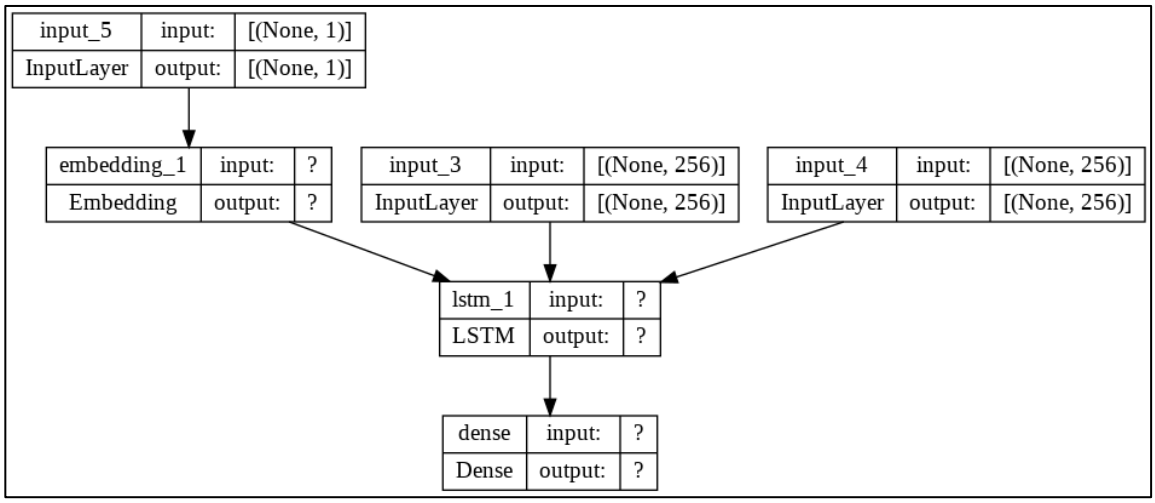
Gambar 3.3 Rancangan Model

3.7 *Inference Model*

Inference model adalah pembuatan model untuk menyimpulkan atau mencari makna atau informasi agar mesin dapat memprediksi kalimat yang diinputkan ke dalam mesin translasi pada saat proses penerjemahan berlangsung. Penggunaan *Inference model* ini mencegah adanya salah dalam pemaknaan kalimat dalam output yang dihasilkan. Tahap ini akan melaksanakan *Inference model* untuk *encoder* dan *decoder*. *Inference encoder model* terdiri dari tiga lapisan yaitu *input layer* sebagai lapisan input, *embedding layer* sebagai lapisan embedding, dan LSTM *layer* sebagai lapisan LSTM. *Inference decoder model* terdiri dari empat lapisan yaitu *input layer* sebagai lapisan input, *embedding layer* sebagai lapisan embedding, LSTM *layer* sebagai lapisan LSTM, dan *dense layer* sebagai lapisan padat. Rancangan *Inference model* untuk *encoder* dan *decoder* dapat dilihat pada Gambar 3.4 dan Gambar 3.5.



Gambar 3.4 *Inference Encoder Model*



Gambar 3.5 *Inference Decoder Model*

3.8 Penyimpanan Model

Model yang sudah dilatih dan dilakukan *Inference* kemudian perlu disimpan agar dapat digunakan lagi dan tidak perlu melatih ulang data. Model disimpan menggunakan *library* dari *Python* yaitu *TensorFlow Keras*.

3.9 Pengujian

Pengujian nilai dari hasil translasi akan dilaksanakan dengan tiga metode, pertama dengan melakukan pengujian pada sistem yang dibangun, kedua dengan melakukan

perhitungan nilai dengan *Bilingual Evaluation Understudy* (BLEU), dan ketiga pengujian dilakukan oleh responden.

3.10 Evaluasi Otomatis

Sistem pengujian *Bilingual Evaluation Understudy* (BLEU) merupakan algoritma sederhana guna untuk mengevaluasi kualitas hasil dari sebuah terjemahan mesin translasi terhadap terjemahan manusia. Sistem ini bekerja secara otomatis dengan mengukur modifikasi skor presisi *n-gram* antara hasil terjemahan mesin translasi terhadap terjemahan manusia dengan *brevity penalty* sebagai tetapannya.

BLEU dihitung menggunakan beberapa presisi *n-gram* yang dimodifikasi secara khusus, skor BLEU diperoleh dari hasil perkalian antara *brevity penalty* dengan rata-rata geometri dari skor presisi yang dimodifikasi. Apabila nilai BLEU yang diperoleh semakin tinggi maka nilai terjemahan semakin akurat dengan sumbernya. Perlu dimengerti bahwasanya semakin banyak terjemahan sumber per kalimatnya maka akan semakin tinggi pula dalam memperoleh nilainya. Dalam rangka memperoleh hasil nilai BLEU yang maksimal, panjang kalimat hasil translasi harus mendekati panjang dari kalimat sumber dan kalimat hasil translasi harus memiliki kata dan urutan yang presisi dengan kalimat sumber.

$$\text{BLEU} = \text{BP} \,.\, \exp\left(\sum_{n=0}^{N} w_n \log P_n\right) \tag{3.1}$$

$$\text{BP} = \begin{cases} 1, & \text{c} > \text{r} \\ \exp\left(1 - \dfrac{r}{c}\right), & \text{c} \leq \text{r} \end{cases} \tag{3.2}$$

Keterangan:

BP = *Brevity Penalty*

c = Jumlah kata dari hasil translasi otomatis

r = Jumlah kata sumber

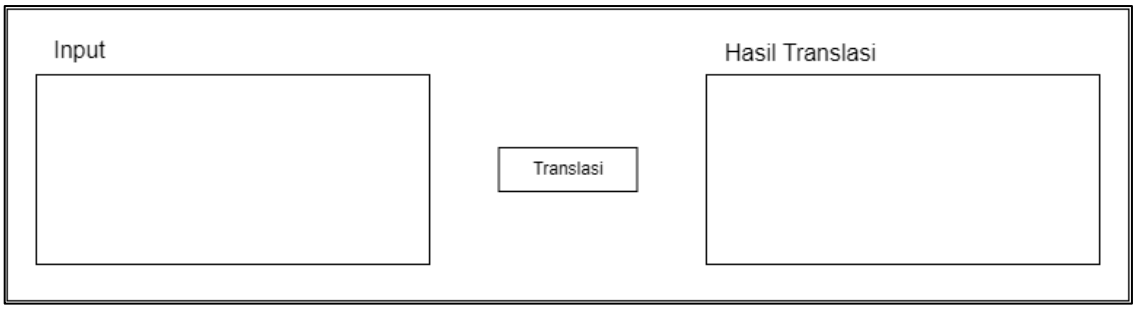
Pn = Skor presisi yang dimodifikasi

wn = 1/N (BLEU memiliki standar nilai N sebanyak 4)

pn = Jumlah *n-gram* (hasil translasi sesuai sumber dibagi jumlah *n-gram* hasil translasi)

3.11 Perancangan Perangkat Lunak

Perancangan perangkat dilakukan untuk memberikan tampilan sehingga lebih mudah untuk digunakan. Perangkat lunak akan dibuat berbasis *website* menggunakan bahasa pemrograman *Python* dan sebagai menjembatani *software*nya dengan menggunakan *Framework flask*. Perangkat yang akan digunakan masih dibuat dengan lokal dan belum bisa di akses secara umum. Perangkat lunak sebagai bentuk komunikasi antara pengguna dengan mesin. Rancangan perangkat lunak terdiri dari *form input* dan *textfield* hasil translasi. Rancangan antarmuka dari pengimplementasian program mesin translasi dapat dilihat pada **Error! Reference source not found.**



Gambar 3.6 Implementasi Rancangan Antarmuka

BAB IV
IMPLEMENTASI DAN PENGUJIAN

4.1 Pengumpulan Data

Pencarian data translasi bahasa Indonesia ke bahasa Jawa Ngoko diambil dari forum komunitas penelitian *Natural Language Processing* (NLP) untuk bahasa Jawa Ngoko dan bahasa Indonesia. Data untuk translasi bahasa Indonesia ke bahasa Jawa Ngoko yang dipilih adalah data yang dibuat oleh Herry Sujaini dengan nama projek data tersebut adalah Korpus Nusantara. Jumlah data translasi bahasa Indonesia ke bahasa Jawa Ngoko yang tersedia terdapat 6059 data dengan kalimat terpanjang yaitu 37 kata pada kalimat di dalam dataset. Untuk dataset lengkapnya dari dataset A terdapat pada lampiran dataset. Pemakaian data yang akan digunakan pada penelitian ini hanya akan mengambil sebanyak 5000 data pertama dari total 6059 data. Data kemudian dipisah dengan ketentuan bahasa Indonesia sebagai *input* dan bahasa Jawa Ngoko sebagai *output*. Implementasi pengolahan data dapat dilihat pada Gambar 4.1.

```
def load_data(path_dataset, jumlah_data = 20000):
  input_sentences = []
  output_sentences = []

  count = 1

  for line in open(path_dataset, encoding="latin-1"):
      if count > jumlah_data:
          break

      if '\t' not in line:
          continue

      output_sentence, input_sentence = line.rstrip().split('\t')

      input_sentences.append(input_sentence)
      output_sentences.append(output_sentence)

      count += 1

  return input_sentences, output_sentences

path_dataset = 'C:\\Users\\ASUS\\Desktop\\pendadaran\\DATASET\\dataset_ver5.txt'
jumlah_data = 5000
input_sentences, output_sentences = load_data(path_dataset, jumlah_data)
```

Gambar 4.1 Implementasi pengolahan pada dataset

Contoh isi dari dataset yang digunakan dapat dilihat pada Tabel 4.1.

Tabel 4.1 Contoh isi dataset

Bahasa Indonesia	Bahasa Jawa Ngoko
"Manusia membutuhkan makanan dan air supaya menjadi kuat dan sehat, begitu juga tanaman"	"Manungsa butuh pangan lan banyu dadi kuwat lan sehat, supaya tetanduran"
Tanaman hijau menggunakan air untuk membuat makanannya	Tetanduran ijo nggunakake banyu kanggo nggawe panganan
Tanaman yang tidak mendapat air akan layu dan menjadi kering	Tanduran sing ora bisa banyu bakal garing lan garing
Keberhasilan belajar murid tidak hanya bergantung pada jumlah dan mutu guru yang memadai	Sukses sinau siswa ora mung gumantung marang nomer lan kualitas guru
"Namun, di sekolah yang jumlah dan mutu gurunya kurang, diduga potensi belajar murid akan sulit mencapai hasil maksimal"	"Nanging, ing sekolah-sekolah ing ngendi jumlah lan kualitas guru kurang, disangka sing potensial learning siswa bakal angel entuk asil maksimal"

4.2 *Preprocessing*

Preprocessing salah satu tahapan dalam melakukan *data mining* yang dilakukan sebelum menuju tahap pemprosesan. Proses ini data akan diubah menjadi bentuk yang dapat lebih dipahami oleh sistem melalui cara membersihkan, mentransformasikan, mereduksi, serta mengeliminasi data yang tidak sesuai. Hal ini dilakukan untuk menghilangkan beberapa permasalahan yang sering mengganggu proses berlangsungnya pengolahan data. Tahapan *preprocessing* sebagai berikut.

a. *Case Folding*

Penggunaan *case folding* disertakan dalam melakukan proses *text preprocessing* agar data yang dimiliki menjadi terstruktur dan konsisten dalam penggunaan huruf kapital. Implementasi *case folding* dilakukan agar isi dari dataset menjadi sama. Pada implementasi ini, dataset akan dibuat seragam dengan cara *lowercase* untuk menyetarakan hurufnya. Penerapan dalam pelaksanaan *case folding* dapat dilihat pada Gambar 4.2 dan contoh implementasi *case folding* dapat dilihat pada Tabel 4.2.

```
sentences = [sentence.lower() for sentence in sentences]
```

Gambar 4.2 Implementasi *case folding*

Tabel 4.2 Contoh dataset setelah *case folding*

Bahasa Indonesia	Bahasa Jawa Ngoko
"manusia membutuhkan makanan dan air supaya menjadi kuat dan sehat, begitu juga tanaman"	"manungsa butuh pangan lan banyu dadi kuwat lan sehat, supaya tetanduran"
tanaman hijau menggunakan air untuk membuat makanannya	tetanduran ijo nggunakake banyu kanggo nggawe panganan
tanaman yang tidak mendapat air akan layu dan menjadi kering	tanduran sing ora bisa banyu bakal garing lan garing
keberhasilan belajar murid tidak hanya bergantung pada jumlah dan mutu guru yang memadai	sukses sinau siswa ora mung gumantung marang nomer lan kualitas guru
"namun, di sekolah yang jumlah dan mutu gurunya kurang, diduga potensi belajar murid akan sulit mencapai hasil maksimal"	"nanging, ing sekolah sekolah ing ngendi jumlah lan kualitas guru kurang, disangka sing potensial learning siswa bakal angel entuk asil maksimal"

b. Menghilangkan *punctuation* (tanda baca)

Penghilangan tanda baca bertujuan untuk membersihkan dataset dari tanda baca yang dianggap tidak relevan untuk ditranslasi. Tanda baca yang dihilangkan menggunakan library dari python dengan perintah *string.punctuation*. Gambar 4.3 merupakan isi dari perintah tersebut adalah sebagai berikut dan implementasi penghilangan *punctuation* dapat dilihat pada Gambar 4.4 dan hasil implementasi dari penghilangan *punctuation* pada dataset dapat dilihat pada Tabel 4.3.

```
!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~
```

Gambar 4.3 Isi perintah *string.punctuation*

```
  sentences = [sentence.translate
               (sentence.maketrans('', '', string.punctuation))
               for sentence in sentences]
  return sentences

input_sentences = lower_and_remove_punctuation(input_sentences)
output_sentences = lower_and_remove_punctuation(output_sentences)
```

Gambar 4.4 Implementasi penghilangan *punctuation*

Tabel 4.3 Contoh dataset setelah tanda baca dihilangkan

Bahasa Indonesia	**Bahasa Jawa Ngoko**
manusia membutuhkan makanan dan air supaya menjadi kuat dan sehat begitu juga tanaman	manungsa butuh pangan lan banyu dadi kuwat lan sehat supaya tetanduran
tanaman hijau menggunakan air untuk membuat makanannya	tetanduran ijo nggunakake banyu kanggo nggawe panganan
tanaman yang tidak mendapat air akan layu dan menjadi kering	tanduran sing ora bisa banyu bakal garing lan garing
keberhasilan belajar murid tidak hanya bergantung pada jumlah dan mutu guru yang memadai	sukses sinau siswa ora mung gumantung marang nomer lan kualitas guru
namun di sekolah yang jumlah dan mutu gurunya kurang diduga potensi belajar murid akan sulit mencapai hasil maksimal	nanging ing sekolah sekolah ing ngendi jumlah lan kualitas guru kurang disangka sing potensial learning siswa bakal angel entuk asil maksimal

c. Penambahan *tag* (penanda)

Implementasi selanjutnya adalah pemberian tambahan *tag* (penanda). Penggunaan tanda *Tag* berfungsi untuk menandakan awal dan akhir kalimat yang akan di translansikan. *Tag* yang diberikan berupa '<*sos*>' sebagai tanda awal kalimat, dan '<*eos*>' sebagai akhir kalimat yang akan di translasi kan. Pemberian *tag* tambahan hanya diimplementasikan untuk data yang berperan sebagai *output*, maka dari itu bahasa Jawa Ngoko yang akan diimplementasikan dalam penambahan tanda *tag* ini. Implementasi penambahan *tag* dapat dilihat pada Gambar 4.5 dan contoh hasil penambahan *tag* pada dataset dapat dilihat pada Tabel 4.4.

```
def add_tag(sentences, tag):
  if (tag == '<eos>'):
    sentences = [sentence + ' <eos>' for sentence in sentences]
  elif(tag == '<sos>'):
    sentences = ['<sos> ' + sentence for sentence in sentences]

  return sentences

output_sentences_inputs = add_tag(output_sentences, '<sos>')
output_sentences = add_tag(output_sentences, '<eos>')
```

Gambar 4.5 Implementasi kode pemberian tambahan *tag* (penanda)

Tabel 4.4 Contoh implementasi penambahan *tag*

Bahasa Jawa Ngoko (dengan *tag* ‘’<*sos*>”)	**Bahasa Jawa Ngoko (dengan *tag* ‘’<*eos*>”)**
<*sos*> manungsa butuh pangan lan banyu dadi kuwat lan sehat supaya tetanduran	manungsa butuh pangan lan banyu dadi kuwat lan sehat supaya tetanduran <*eos*>
<*sos*> tetanduran ijo nggunakake banyu kanggo nggawe panganan	tetanduran ijo nggunakake banyu kanggo nggawe panganan <*eos*>
<*sos*> tanduran sing ora bisa banyu bakal garing lan garing	tanduran sing ora bisa banyu bakal garing lan garing <*eos*>
<*sos*> sukses sinau siswa ora mung gumantung marang nomer lan kualitas guru	sukses sinau siswa ora mung gumantung marang nomer lan kualitas guru <*eos*>
<*sos*> nanging ing sekolah sekolah ing ngendi jumlah lan kualitas guru kurang disangka sing potensial learning siswa bakal angel entuk asil maksimal	nanging ing sekolah sekolah ing ngendi jumlah lan kualitas guru kurang disangka sing potensial learning siswa bakal angel entuk asil maksimal <*eos*>

d. *Tokenizing*

Implementasi selanjutnya adalah *tokenizing*. *Tokenizing* berfungsi untuk memecah kalimat menjadi tiap kata yang kemudian disimpan dalam kamus dan diberi indeks berupa nilai *integer*. Kamus ini yang menyimpan kosakata dalam bahasa Indonesia dan bahasa Jawa Ngoko. Implementasi tokenizing menggunakan *library python* bernama *TensorFlow Keras* dengan *class Tokenizer* dengan maksimal jumlah kata sebanyak 20000. Implementasi *tokenizing* berlaku untuk semua bahasa, namun dengan *tokenizer* untuk bahasa itu sendiri. Implementasi *tokenizing* untuk bahasa Indonesia dapat dilihat pada Gambar 4.6.

```
input_tokenizer = Tokenizer(num_words=max_jml_kata)

input_tokenizer.fit_on_texts(input_sentences)
input_integer_seq = input_tokenizer.texts_to_sequences(input_sentences)

word2idx_inputs = input_tokenizer.word_index
max_input_len = max(len(sen) for sen in input_integer_seq)
```

Gambar 4.6 Implementasi kode untuk tokenisasi bahasa Indonesia

Hasil implementasi *tokenizing* bahasa indonesia memperoleh kata yang unik sebanyak 5677 dan panjang kalimat yang diperoleh sepanjang 36 kata. Contoh dataset dapat dilihat pada Tabel 4.5.

Tabel 4.5 Contoh implementasi *tokenizing* untuk bahasa Indonesia

Bahasa Indonesia	**Hasil *Tokenizing***
manusia membutuhkan makanan dan air supaya menjadi kuat dan sehat begitu juga tanaman	'manusia', 'membutuhkan', 'makanan', 'dan', 'air', 'supaya', 'menjadi', 'kuat', 'sehat', 'begitu', 'juga', 'tanaman'.
tanaman hijau menggunakan air untuk membuat makanannya	'tanaman', 'hijau', 'menggunakan', 'air', 'untuk' 'membuat', 'makanannya'.
tanaman yang tidak mendapat air akan layu dan menjadi kering	'tanaman', 'yang', 'tidak', 'mendapat', 'air', 'akan', 'layu', 'dan', 'menjadi', 'kering'.
keberhasilan belajar murid tidak hanya bergantung pada jumlah dan mutu guru yang memadai	'keberhasilan', 'belajar', 'murid', 'tidak', 'hanya', 'bergantung', 'pada', 'jumlah', 'dan', 'mutu', 'guru', 'yang', 'memadai'.
namun di sekolah yang jumlah dan mutu gurunya kurang diduga potensi belajar murid akan sulit mencapai hasil maksimal	'namun', 'di', 'sekolah', 'yang', 'jumlah', 'dan', 'mutu', 'gurunya', 'kurang', 'diduga', 'potensi', 'belajar', 'murid', 'akan', 'sulit', 'mencapai', 'hasil', 'maksimal'.

Implementasi *tokenizing* untuk bahasa Jawa Ngoko dapat dilihat pada Gambar 4.7

```
output_tokenizer = Tokenizer(num_words=max_jml_kata, filters='')

output_tokenizer.fit_on_texts(output_sentences + output_sentences_inputs)

output_integer_seq = output_tokenizer.texts_to_sequences(output_sentences)
output_input_integer_seq = output_tokenizer.texts_to_sequences(output_sentences_inputs)

word2idx_outputs = output_tokenizer.word_index
max_out_len = max(len(sen) for sen in output_integer_seq)
```

Gambar 4.7 Implementasi kode untuk tokenisasi bahasa Jawa Ngoko

Hasil implementasi *tokenizing* bahasa indonesia memperoleh kata yang unik sebanyak 5677 dan panjang kalimat yang diperoleh sepanjang 36 kata. Contoh dataset dapat dilihat pada Tabel 4.6.

Tabel 4.6 Contoh implementasi *tokenizing* bahasa Jawa Ngoko

Bahasa Jawa Ngoko	**Hasil *Tokenizing***
manungsa butuh pangan lan banyu dadi kuwat lan sehat supaya tetanduran <*eos*>	'manungsa', 'butuh', 'pangan', 'lan', 'banyu', 'dadi', 'kuwat', 'sehat', 'supaya', 'tetanduran', '<*eos*>'.
tetanduran ijo nggunakake banyu kanggo nggawe panganan <*eos*>	'tetanduran', 'ijo', 'nggunakake', 'banyu', 'kanggo', 'nggawe', 'panganan', '<*eos*>'.
tanduran sing ora bisa banyu bakal garing lan garing <*eos*>	'tanduran', 'sing', 'ora', 'bisa', 'banyu', 'bakal', 'garing', 'lan', 'garing', '<*eos*>'.
sukses sinau siswa ora mung gumantung marang nomer lan kualitas guru <*eos*>	'sukses', 'sinau', 'siswa', 'ora', 'mung', 'gumantung', 'marang', 'nomer', 'lan', 'kualitas', 'guru', '<*eos*>'.
nanging ing sekolah sekolah ing ngendi jumlah lan kualitas guru kurang disangka sing potensial learning siswa bakal angel entuk asil maksimal <*eos*>	'nanging', 'ing', 'sekolah', 'ngendi', 'jumlah', 'lan', 'kualitas', 'guru', 'kurang', 'disangka', 'sing', 'potensial', 'learning', 'siswa', 'bakal', 'angel', 'entuk' 'asil', 'maksimal', '<*eos*>'.

e. *Padding*

Padding adalah proses penyetaraan panjang data dengan memberi nilai nol (0) pada akhir data atau pada awal data. Teks yang dihasilkan dalam suatu kalimat memiliki panjang yang bervariasi. Penggunaan *Padding* ini digunakan untuk memenuhi persyaratan dalam penerapan *Long-short Term Memory* (LSTM) yang mengharapkan isi kalimat dengan

panjang yang sama. Pada implementasi ini, data *input* akan diberi *padding* pada bagian awal, sedangkan data *output* akan diberi *padding* pada bagian akhir. Implementasi *padding* pada data *input* dapat dilihat pada Gambar 4.8 dan Implementasi *padding* pada data *output* dapat dilihat pada Gambar 4.9. Contoh hasil dari implementasi *padding* dapat dilihat pada Tabel 4.7 dan Tabel 4.8 sebagai berikut.

```
encoder_input_sequences = pad_sequences(input_integer_seq, maxlen=max_input_len)
```

Gambar 4.8 Implementasi *padding* pada data *input*

Tabel 4.7 Contoh hasil dari implementasi *padding* pada data *input*

Bahasa Indonesia	**Bentuk *Sequence***	**Hasil *Padding***
manusia membutuhkan makanan dan air supaya menjadi kuat dan sehat begitu juga tanaman	[318, 1090, 298, 4, 362, 171, 186, 806, 4, 871, 40, 26, 547]	[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, 318, 1090, 298, 4, 362, 171, 186, 806, 4, 871, 40, 26, 547]
tanaman hijau menggunakan air untuk membuat makanannya	[547, 1758, 308, 362, 21, 152, 2210]	[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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 547, 1758, 308, 362, 21, 152, 2210]
tanaman yang tidak mendapat air akan layu dan menjadi kering	[547, 1, 5, 395, 362, 53, 3078, 4, 186, 609]	[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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 547, 1, 5, 395, 362, 53, 3078, 4, 186, 609]
keberhasilan belajar murid tidak hanya bergantung pada jumlah dan mutu guru yang memadai	[1236, 972, 743, 5, 64, 2211, 18, 135, 4, 1446, 417, 1, 2212]	[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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1236, 972, 743, 5, 64, 2211, 18, 135, 4, 1446, 417, 1, 2212]
namun di sekolah yang jumlah dan mutu gurunya kurang diduga potensi belajar murid akan sulit mencapai hasil maksimal	[396, 3, 232, 1, 135, 4, 1446, 1759, 279, 1760, 2213, 972, 743, 53, 807, 440, 493, 2214]	[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, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 396, 3, 232, 1, 135, 4, 1446, 1759, 279, 1760, 2213, 972, 743, 53, 807, 440, 493, 2214]

```
decoder_input_sequences = pad_sequences
(output_input_integer_seq, maxlen=max_out_len, padding='post')
```

Gambar 4.9 Implementasi *padding* pada data *output*

Tabel 4.8 Contoh hasil dari implementasi *padding* pada data *output*

Bahasa Jawa Ngoko	Bentuk *Sequence*	Hasil *Padding*
manungsa butuh pangan lan banyu dadi kuwat lan sehat supaya tetanduran <*eos*>	[338, 869, 786, 4, 370, 29, 870, 4, 2336, 164, 2337, 1]	[2, 338, 869, 786, 4, 370, 29, 870, 4, 2336, 164, 2337, 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]
tetanduran ijo nggunakake banyu kanggo nggawe panganan <*eos*>	[2337, 1264, 532, 370, 21, 1460, 305, 1]	[2, 2337, 1264, 532, 370, 21, 1460, 305, 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]
tanduran sing ora bisa banyu bakal garing lan garing <*eos*>	[617, 3, 6, 27, 370, 70, 496, 4, 496, 1]	[2, 617, 3, 6, 27, 370, 70, 496, 4, 496, 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]
sukses sinau siswa ora mung gumantung marang nomer lan kualitas guru	[2338, 1092, 957, 6, 49, 3381, 30, 1804, 4, 1805, 357, 1]	[2, 2338, 1092, 957, 6, 49 3381, 30 1804, 4, 1805, 357, 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]
nanging ing sekolah sekolah ing ngendi jumlah lan kualitas guru kurang disangka sing potensial learning siswa bakal angel entuk asil maksimal <*eos*>	[31, 7, 1806, 7, 174, 450, 4, 1805, 357, 223, 3382, 3, 3383, 3384, 957, 70, 714, 497, 715, 2339, 1]	[2, 31, 7, 1806, 7, 174, 450, 4, 1805, 357, 223 3382, 3, 3383, 3384, 957, 70, 714, 497, 715 2339, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

f. Pickle

Pickle merupakan modul yang disediakan dalam *standard library Python* yang berfungsi sebagai menyimpan dan membaca data variabel yang telah diolah ke dalam bentuk *file*. Penggunaan *Pickle* ini berguna untuk membaca kembali data yang disimpan dari *file* tersebut saat dibutuhkan. Implementasi penggunaan dalam tahap penyimpanan data pada *Pickle* dapat dilihat pada Gambar 4.10.

```
with open('C:\\Users\\ASUS\\Desktop\\pendadaran\\PICKLE\\data_objects_ver5.pickle', 'wb') as handle:
    pickle.dump({
        'input_tokenizer': input_tokenizer,
        'output_tokenizer': output_tokenizer,
        'max_input_len': max_input_len,
        'max_out_len': max_out_len,
      }, handle, protocol=pickle.HIGHEST_PROTOCOL
    )
```

Gambar 4.10 Implementasi tahap penyimpanan data pada *Pickle*

Implementasi penggunaan dalam tahap menggunakan dan membaca data yang sebelumnya sudah disimpan pada *file* menggunakan *Pickle* dapat dilihat pada Gambar 4.11.

```
with open("C:\\Users\\ASUS\\Desktop\\pendadaran\\PICKLE\\data_objects_ver5.pickle", 'rb') as f:
    data = pickle.load(f)
    input_tokenizer = data['input_tokenizer']
    output_tokenizer = data['output_tokenizer']
    max_input_len = data['max_input_len']
    max_out_len = data['max_out_len']

word2idx_inputs = input_tokenizer.word_index
word2idx_outputs = output_tokenizer.word_index
```

Gambar 4.11 Implementasi tahap pemakaian data pada *Pickle*

4.3 *Word Embeddings*

Data yang telah di *preprocessing* kemudian dilakukan pembobotan dengan *word embeddings*. *Word embeddings* merupakan metode *Natural Language Processing* (NLP) untuk memetakan tiap kata dalam kamus hasil tokenisasi ke dalam nilai vektor dengan menggunakan data GloVe (*Global Vectors*) sebagai penyedia nilai vektor *word embeddings* yang digunakan untuk memprediksi kata, semantik kata, dan kemiripan kata. Pada tahap ini, data Bahasa Indonesia akan dilakukan *word embeddings* dengan menyematkan nilai vektor dari penyedia *word embeddings* ke kata dalam kamus hasil tokenisasi dengan 100 dimensi. Implementasi *word embeddings* pada data dapat dilihat pada Gambar 4.12.

```
EMBEDDING_SIZE = 100
embeddings_dictionary = dict()

glove_file = open(r'C:\\Users\\ASUS\\Desktop\\pendadaran\\WORD EMBEDDING\\Model100dm.txt',
                  encoding="latin-1")

next(glove_file)

for line in glove_file:
    records = line.split()

    try:
      word = records[0]

      vector_dimensions = np.asarray(records[1:], dtype='float32')
    except:
        pass

    embeddings_dictionary[word] = vector_dimensions

glove_file.close()

num_words = min(max_jml_kata, len(word2idx_inputs) + 1)

embedding_matrix = np.zeros((num_words, EMBEDDING_SIZE))

for word, index in word2idx_inputs.items():
    if index > num_words:
      break

    embedding_vector = embeddings_dictionary.get(word)

    if embedding_vector is not None:
        embedding_matrix[index] = embedding_vector
```

Gambar 4.12 Penggunaan *Word embedding* pada *input*

Agar memastikan hasil dari prediksi kata, semantik kata, dan kemiripan kata dari penggunaan *Word embedding* dengan melakukan pengecekan kata pada Gambar 4.13 kemudian membandingkan apakah kata dengan indeks tersebut memiliki kemiripan dengan cara menampilkan vektornya pada Gambar 4.14 dan Gambar 4.15. Setelah melakukan beberapa pencocokan pada vektornya dapat disimpulkan bahwa kata tanaman memiliki kemiripan dengan indeks 547.

```
print(word2idx_inputs["tanaman"])
```

```
547
```

Gambar 4.13 Implementasi *Word embedding* dari kata tanaman

```
print(embeddings_dictionary["tanaman"])
```

```
[-0.144185 -0.392203  0.222141  0.237954 -0.520795  0.159329 -0.064351
  0.235271 -0.243593  0.243802  0.025506  0.40654  -0.369392 -0.714041
 -0.193339 -0.530592 -0.005106 -0.328899  0.656107  0.378511 -0.027514
  0.542274 -0.175885 -0.232672 -0.077764  0.038534 -1.085296 -0.087581
  0.109854 -0.062655  0.380769  0.143738 -0.063331  0.300602  0.125186
  0.033036  0.531766  0.056533 -0.133737  0.054702 -0.032955  0.964193
 -0.537204 -0.20089  -0.092723 -0.249207  0.519917  0.47198  -0.781405
 -0.302421  0.139777 -0.691386 -0.081192 -0.139992  0.080248 -0.610953
 -0.069127 -0.289361 -0.051387  0.162075  0.10959  -0.265222  0.321965
 -0.166955  0.11853   0.554304  0.344821 -0.38721   0.516805 -0.156922
 -0.003092  0.103363  0.355818  0.005843 -0.75808  -0.661649  0.066909
 -0.127244 -0.794461 -0.60969  -0.281256  0.406569  0.180257  0.600998
  0.277325 -0.0449   -0.048194 -0.402006  0.411247  0.662114  0.078224
  0.218132 -0.055036  0.58583  -0.598649 -0.018336 -0.013486 -0.713201
 -0.126561 -0.394893]
```

Gambar 4.14 Contoh vektor yang dihasilkan dari kata tanaman

```
print(embedding_matrix[547])
```

```
[-0.14418501 -0.392203    0.222141    0.23795401 -0.52079499  0.159329
 -0.064351    0.23527101 -0.24359301  0.243802    0.025506    0.40654001
 -0.36939201 -0.71404099 -0.19333901 -0.53059202 -0.005106   -0.328899
  0.65610701  0.37851101 -0.027514    0.542274   -0.17588501 -0.23267201
 -0.077764    0.038534   -1.08529603 -0.087581    0.109854   -0.062655
  0.38076901  0.143738   -0.063331    0.30060199  0.125186    0.033036
  0.531766    0.056533   -0.133737    0.054702   -0.032955    0.96419299
 -0.53720403 -0.20089    -0.092723   -0.249207    0.51991701  0.47198001
 -0.78140497 -0.302421    0.139777   -0.69138598 -0.081192   -0.139992
  0.080248   -0.61095297 -0.069127   -0.289361   -0.051387    0.162075
  0.10959    -0.26522201  0.32196501 -0.16695499  0.11853     0.554304
  0.34482101 -0.38721001  0.51680499 -0.156922   -0.003092    0.103363
  0.355818    0.005843   -0.75808001 -0.66164899  0.066909   -0.127244
 -0.79446101 -0.60969001 -0.28125599  0.406569    0.18025699  0.60099798
  0.277325   -0.0449     -0.048194   -0.402006    0.41124699  0.66211402
  0.078224    0.218132   -0.055036    0.58582997 -0.59864902 -0.018336
 -0.013486   -0.71320099 -0.126561   -0.39489299]
```

Gambar 4.15 Contoh vektor yang dihasilkan dari indeks 547

4.4 *One Hot Vector Decoder*

One hot vector berguna untuk membantu perhitungan bobot pada decoder dalam mengatasi keterbatasan data *word embeddings* untuk bahasa daerah termasuk bahasa Jawa Ngoko. *One hot vector* merupakan salah satu metode untuk mempresentasikan suatu data menjadi vektor yang terdiri dari angka biner yaitu 0 dan 1, dengan ketentuan semua nilai elemen akan bernilai 0 kecuali elemen yang memiliki kesesuaian dengan data akan bernilai 1. Kesesuaian data pada tahap ini adalah kesesuaian nilai indeks dalam *sequence* dengan nilai kolom pada *One hot vector*. Implementasi *One hot vector* pada data dapat dilihat pada Gambar 4.16.

```
num_words_output = len(word2idx_outputs) + 1

def init_decoder_targets_one_hot():
  decoder_targets_one_hot = np.zeros((
          len(input_sentences),
          max_out_len,
          num_words_output
      ),
      dtype='float32'
  )

  for i, d in enumerate(decoder_output_sequences):
    for t, word in enumerate(d):
        decoder_targets_one_hot[i, t, word] = 1

  return decoder_targets_one_hot

decoder_targets_one_hot = init_decoder_targets_one_hot()
```

Gambar 4.16 Implementasi *One hot vector* pada *output*

4.5 Data Latih dan Data Uji

Dataset dibagi menjadi dua bagian yaitu data latih dan data uji. Data yang diambil dari dataset kemudian digunakan untuk melaksanakan pelatihan model merupakan dari bagian data latih sedangkan data yang diambil dari dataset kemudian digunakan untuk melaksanakan pengujian dan melakukan validasi nilai hasil dari pelatihan model berdasarkan data latih itu merupakan bagian dari data uji. Persentase pembagian dalam data latih sebanyak 90% dari dataset dan data uji sebanyak 10% dari dataset.

4.6 Model

Penelitian ini menggunakan model *Recurrent Neural Network* (RNN) dan *Long Short-Term Memory* (LSTM). Arsitektur model yang diterapkan adalah *seq2seq* (*sequence-to-sequence*). *Seq2seq* merupakan model dalam pembelajaran mesin yang memiliki dua komponen yaitu *encoder* dan *decoder*. Pembuatan *encoder* dan *decoder* dilakukan dengan mendefinisikan *layer* yang akan diterapkan. Pada pembuatan *encoder*, pertama dilakukan pembuatan *input layer* dengan bentuk dan ukuran sesuai dengan maksimal kalimat terpanjang pada data input. Selanjutnya, pembuatan *embedding layer* dengan membawa nilai bobot dari *word embeddings* dan disambungkan dengan *input layer* yang sudah dibuat di awal. Inisialisasi berikutnya ada LSTM *layer* dengan mendefinisikan jumlah *nodes* yang akan diterapkan untuk digunakan dalam pengimplementasiannya. Pada pelaksanaan penelitian ini

jumlah *nodes* yang diterapkan untuk digunakan dalam pengimplementasiannya adalah sebanyak 256 *nodes*. LSTM layer yang dibuat juga diatur agar mengembalikan nilai *state*. Nilai *state* dari *encoder* ini kemudian akan dijadikan inputan oleh *decoder*. Implementasi dari pembuatan *encoder* dapat dilihat melalui kode pada Gambar 4.17.

```
LSTM_NODES = 256

encoder_inputs_placeholder = Input(shape=(max_input_len,))

x = embedding_layer(encoder_inputs_placeholder)

encoder = LSTM(LSTM_NODES, return_state=True)

encoder_outputs, h, c = encoder(x)

encoder_states = [h, c]
```

Gambar 4.17 Implementasi dari pembuatan *encoder*

Tahapan pembuatan *decoder*, *pertama* dilakukan pembuatan *input layer* dengan bentuk dan ukuran sesuai dengan maksimal kalimat terpanjang pada data output. *Kedua*, pembuatan *embedding layer* dengan membawa nilai bobot dari *One hot vector decoder* disambungkan dengan *input layer* yang sudah dibuat di awal. Inisialisasi berikutnya ada LSTM *layer* dengan mendefinisikan jumlah *nodes* yang akan diterapkan untuk digunakan dalam pengimplementasiannya. Pada penelitian ini, jumlah *nodes* yang diterapkan untuk digunakan dalam pengimplementasiannya adalah sebanyak 256 *nodes*. LSTM *layer* yang dibuat juga diatur agar menerima input dari hasil pengolahan *encoder* dan mengembalikan nilai *state* dan nilai *sequence*. Tahapan paling akhir adalah pembuatan *dense layer* pada *decoder* menerapkan perhitungan *activation softmax*. *Dense layer* berfungsi untuk melakukan prediksi dan menghasilkan output. Implementasi pembuatan *decoder* dapat dilihat melalui kode pada Gambar 4.18.

```
decoder_inputs_placeholder = Input(shape=(max_out_len,))

decoder_embedding = Embedding(num_words_output, LSTM_NODES)

decoder_inputs_x = decoder_embedding(decoder_inputs_placeholder)

decoder_lstm = LSTM(LSTM_NODES, return_sequences=True, return_state=True)

decoder_outputs, _, _ = decoder_lstm(decoder_inputs_x, initial_state=encoder_states)

decoder_dense = Dense(num_words_output, activation='softmax')

decoder_outputs = decoder_dense(decoder_outputs)
```

Gambar 4.18 Implementasi pembuatan *decoder*

Setelah selesai dalam pembuatan *encoder* dan *decoder* kemudian dilanjutkan dengan pembuatan model. Model dibuat dengan membawa nilai *input layer* dari *encoder* dan *decoder* sebagai *input* dari model, serta *dense layer* sebagai *output* dari model. Model kemudian diatur untuk *training* dengan ketentuan parameter *optimizer* menggunakan *rmsprop*, perhitungan *loss* dengan *categorical_crossentropy*, dan *metrics validasi accuracy*. Implementasi dari pembuatan model dapat dilihat melalui kode pada Gambar 4.19.

```
model = Model([encoder_inputs_placeholder, decoder_inputs_placeholder],
              decoder_outputs)

model.compile(
    optimizer='rmsprop',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Gambar 4.19 Implementasi dari pembuatan model

Model yang sudah dibuat kemudian akan dilakukan *training* sebanyak nilai *epochs*, dengan ukuran *batch size*, dan pembagian data validasi sebanyak 10%. Implementasi dari *training* model dapat dilihat melalui kode pada Gambar 4.20. Penentuan nilai *epochs* dan *batch size* dapat dilakukan dengan mempertimbangkan banyaknya jumlah dataset yang akan digunakan dalam *training* model. Penentuan nilai *epochs* dan *batch size* dalam penelitian ini dilakukan 18 kali percobaan untuk mendapatkan nilai validasi akurasi tertinggi dan konvergen. Rincian nilai *epochs* dan *batch size* yang digunakan dalam percobaan dapat dilihat pada Tabel 4.9.

```
BATCH_SIZE = 98
EPOCHS = 54

def train_model(model, batch_size=256, epochs=10, validation_split=0.1):
  model.fit(
      [encoder_input_sequences, decoder_input_sequences],
      decoder_targets_one_hot,
      batch_size=batch_size,
      epochs=epochs,
      validation_split=validation_split,
  )

train_model(model, BATCH_SIZE, EPOCHS)
```

Gambar 4.20 Implementasi dari *training* model

Tabel 4.9 Rincian percobaan untuk menentukan nilai *epochs* dan *batch size*

Batch Size	***Epochs***	**Validasi Akurasi**
64	8	75,26%
	16	83,78%
	20	95,68%
	32	97,16%
	48	97,82%
	54	97,83%
	72	97,83%
98	8	93,75%
	16	97,81%
	32	97,83%
	54	97,85%
	96	97,81%
128	8	75,81%
	16	78,57%
	32	94,08%
	64	95,66%
	96	97,82%
	128	97,84%

Kesimpulan yang dapat diperoleh dari hasil percobaan yang telah dilakukan dapat diketahui bahwa semakin besar nilai *epochs*, maka nilai validasi akurasi dari hasil evaluasi

dataset *training* akan meningkat dan konvergen. Nilai validasi akurasi dari hasil evaluasi dataset *training* terendah yang didapat sebesar 75,26% dan nilai tertinggi sebesar 97,85%. Berdasarkan nilai evaluasi dataset tertinggi, maka diambil nilai *epochs* sebesar 54 dan *batch size* sebesar 98 dengan proses validasi akurasi pada dataset *training* 0,7869 untuk menjadi acuan model dari hasil *training* yang akan digunakan.

4.7 *Inference Model*

Model yang sudah dilatih kemudian dilakukan *Inference model*. *Inference model* dilakukan dengan pembuatan model untuk *encoder* dan model untuk *decoder*. Pembuatan model untuk *encoder* dilakukan dengan inisialisasi model dengan *input layer* dari *encoder* model utama sebagai *input* dan nilai *state* dari *encoder* model utama dan nilai *hidden cell* dan *state cell* sebagai *output*. Implementasi pembuatan *Inference model* pada *encoder* dapat dilihat melalui kode pada Gambar 4.21.

```
encoder_model = Model(encoder_inputs_placeholder, encoder_states)
```

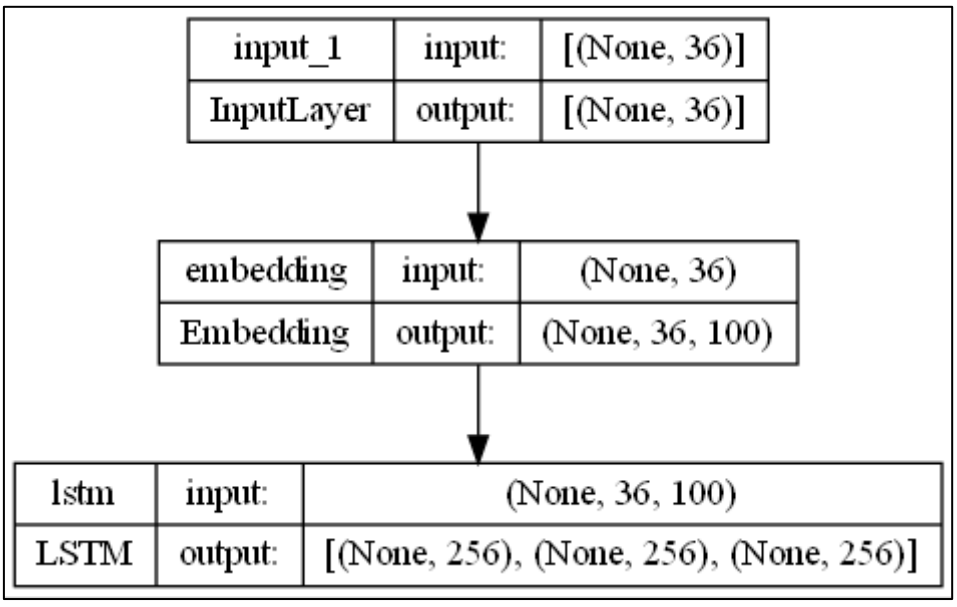
Gambar 4.21 Implementasi *Inference model* pada *encoder*

Pembuatan model untuk bagian *decoder* dilakukan dengan membentuk *state input decoder* yang terbentuk dari dua *input layer*, yaitu *input layer* untuk *state cell* dan *hidden cell*. Dilanjutkan dengan inisialisasi *input layer* dengan ukuran sepanjang satu, karena nilai bobot dari *decoder* berasal dari *One hot vector decoder* yang hanya bernilai 0 atau 1. Inisialisasi berikutnya adalah *embedding layer* yang disambungkan dengan *input layer*. Inisialisasi selanjutnya adalah LSTM *layer*, untuk diambil nilai *hidden cell* dan *state cell* dan dilanjutkan dengan inisialisasi *dense layer*. Nilai *state* tersebut kemudian akan digunakan pada *dense layer*. Setelah semua *layer* terbuat, kemudian dibuatlah model untuk *decoder* dengan *input layer* dan *embedding layer* sebagai *input* dan *output gate* dan nilai *hidden state* dan nilai *state cell* sebagai *output*. Implementasi *Inference model decoder* dapat dilihat melalui kode pada Gambar 4.22.

```
decoder_state_input_h = Input(shape=(LSTM_NODES,))
decoder_state_input_c = Input(shape=(LSTM_NODES,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
decoder_inputs_single = Input(shape=(1,))
decoder_inputs_single_x = decoder_embedding(decoder_inputs_single)
decoder_outputs, h, c = decoder_lstm(decoder_inputs_single_x, initial_state
                                     =decoder_states_inputs)
decoder_states = [h, c]
decoder_outputs = decoder_dense(decoder_outputs)

decoder_model = Model(
    [decoder_inputs_single] + decoder_states_inputs,
    [decoder_outputs] + decoder_states
)
```

Gambar 4.22 Implementasi *Inference model decoder*

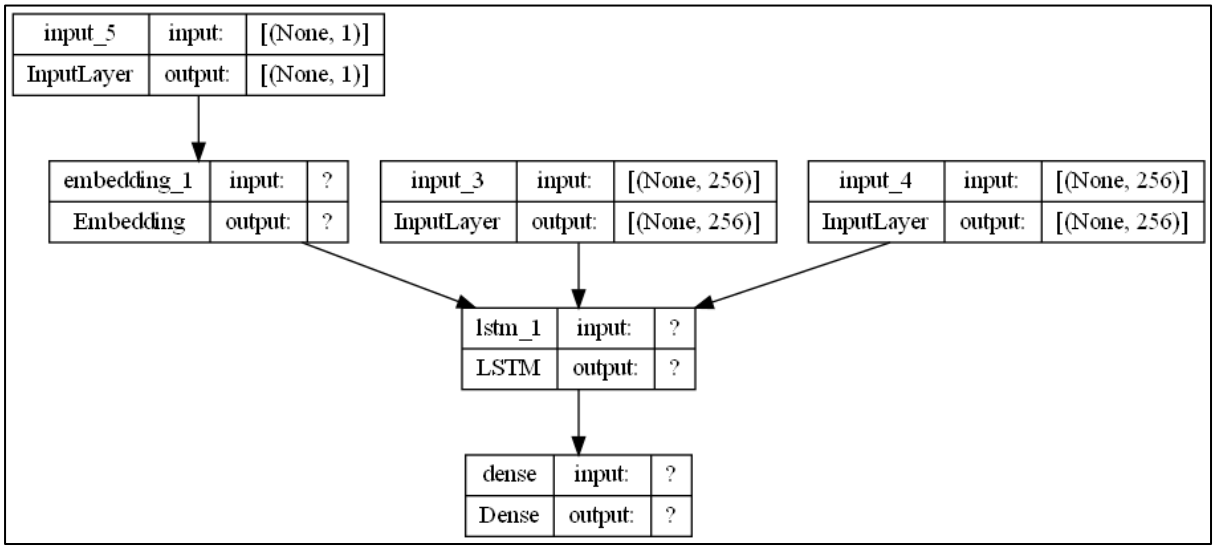
Inference encoder model yang dihasilkan memiliki tiga lapisan yaitu *input layer* sebagai lapisan input, *embedding layer* sebagai lapisan embedding, dan LSTM *layer* sebagai lapisan LSTM. Berikut hasil alur model dari *encoder* dihasilkan oleh inference model dapat dilihat pada Gambar 4.23.



Gambar 4.23 Alur *Inference Encoder Model*

Inference decoder model yang dihasilkan akan terdiri atas empat lapisan yaitu *input layer* sebagai lapisan input, *embedding layer* sebagai lapisan embedding, LSTM *layer*

sebagai lapisan LSTM, dan *dense layer* sebagai lapisan padat. Berikut hasil alur model dari *decoder* dihasilkan oleh inference model dapat dilihat pada Gambar 4.24.



Gambar 4.24 Alur *Inference Decoder Model*

Hal terakhir setelah pembuatan model, *training* model, dan *inference model* adalah menyimpan model. Model disimpan menggunakan *library Python* dari *TensorFlow keras* dengan *function save_model* untuk menyimpan model dan *load_model* untuk memuat model yang tersimpan. Implementasi penyimpanan model dapat dilihat melalui kode pada Gambar 4.25.

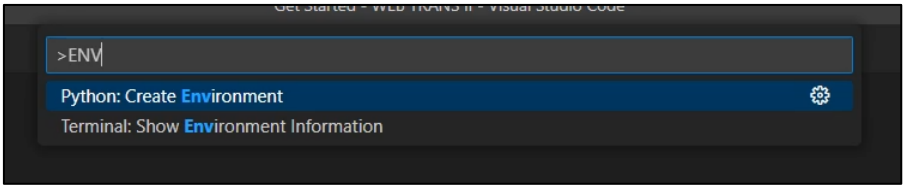
```
model_filename = 'trainmodel.rnn.lstm.indo_jawa_ngoko_epochs_54_bs_98_ver5.2022.h5'
path_model = 'C:\\Users\\ASUS\\Desktop\\pendadaran\\MODEL\\' + model_filename
model.save(path_model)
print("Model Saved")


model_encoder_filename = 'encoder.model.rnn.lstm.indo_jawa_ngoko_ver5.2023.h5'
path_encoder_model = 'C:\\Users\\ASUS\\Desktop\\pendadaran\\MODEL\\' + model_encoder_filename
encoder_model.save(path_encoder_model)
print("Encoder Model Saved")


model_decoder_filename = 'decoder.model.rnn.lstm.indo_jawa_ngoko_ver5.2023.h5'
path_decoder_model = 'C:\\Users\\ASUS\\Desktop\\pendadaran\\MODEL\\' + model_decoder_filename
decoder_model.save(path_decoder_model)
print("Decoder Model Saved")
```

Gambar 4.25 Implementasi penyimpanan model

4.8 Perancangan Antarmuka

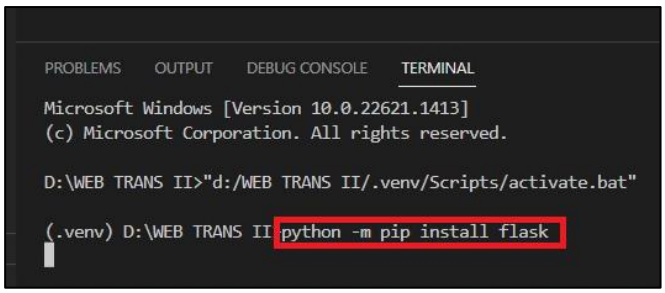
Perangkat lunak akan dibuat berbasis *website* menggunakan bahasa pemrograman *Python* dan sebagai menjembatani perangkat lunak dengan menggunakan *Framework flask*. Perancangan antarmuka yang dibuat akan terdiri dari *form input* dan *textfield* hasil translasi. Penggunaan *flask* bertujuan untuk memberikan tampilan sehingga lebih mudah untuk digunakan. Perancangan dalam pembuatan *page rendering* atau penjembatan yang menampilkan *form input* dari mesin translasi peneliti menggunakan perangkat lunak teks editor multiplatform yang handal dan komplit buatan *Microsoft* yaitu *Visual Studio Code* atau biasa disebut sebagai VS Code.

Langkah pertama unduh dan instal perangkat lunak *Visual Studio Code* kemudian instal ekstensi *Python* pada *Visual Studio Code.* Buat folder baru untuk menyimpan bahan yang dibutuhkan dalam proses *flask.* Buat *environment* yang dibutukan untuk *Python*, disarankan untuk menggunakan sesuai dengan versi yang sama pada versi *Pythonnya.*



Gambar 4.26 Buat *environment* untuk *Python*

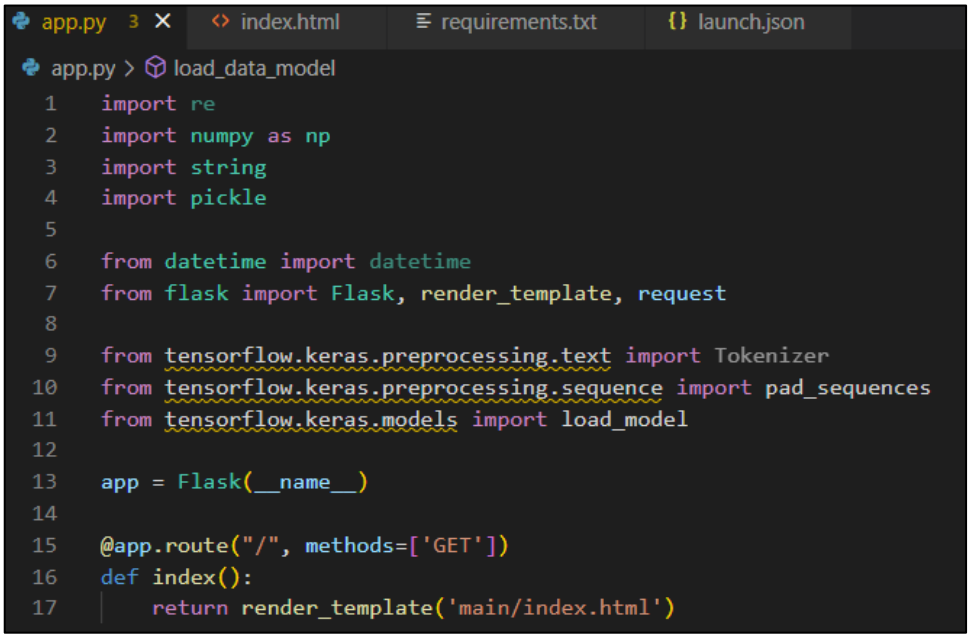
Kemudian instal *flask* di dalam *environment* dengan menggunakan perintah yang disediakan dalam Gambar 4.27 dan tunggu hingga proses instal selesai.



Gambar 4.27 Perintah untuk menginstal *flask*

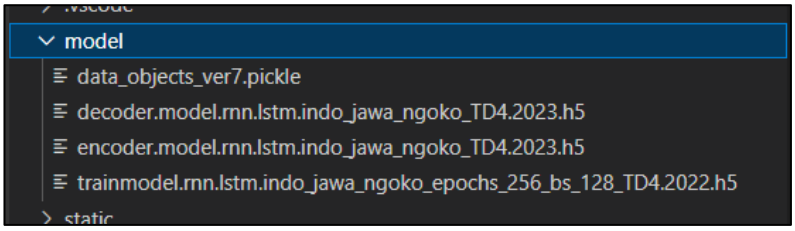
Pada VS Code, buat *file* baru di dalam folder yang telah digunakan dengan nama *app.py* kemudian buat library, *instance* obyek *flask*, dan tambahkan kode untuk mengimpor *flask*. Impor dataset yang telah disimpan setelah melalui pemrosesan data seperti impor

Playground, data *Pickle, training* model, *encoder model, decoder model*, dan fungsi translasi. Tampilan impor dapat dilihat pada Gambar 4.28.



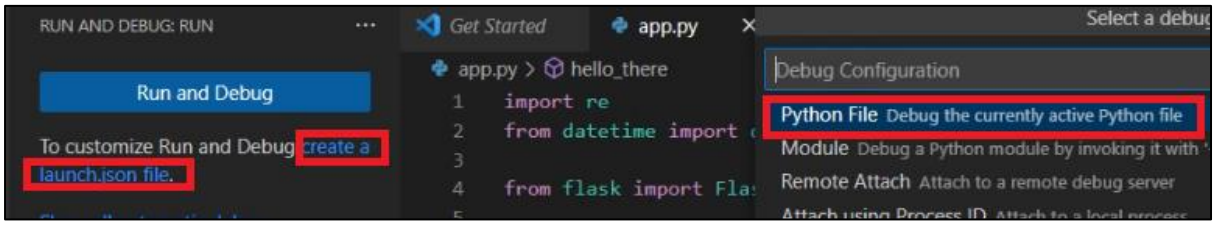
Gambar 4.28 Impor *instance* obyek *flask* dan library yang digunakan

Buat *file* dalam folder yang dibuat pada VS Code dengan berisikan data model yang disimpan dari hasil pemrosesan yang telah dilaksanakan sebelumnya.

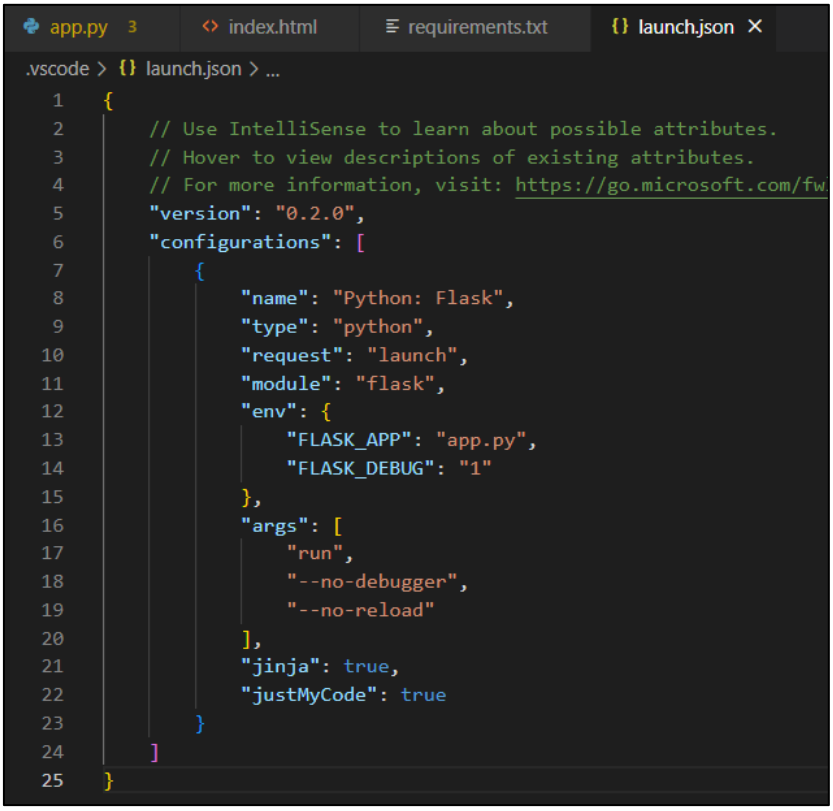


Gambar 4.29 Model hasil pemrosesan yang akan digunakan

Agar dapat mengakses ke halaman *website*nya perlu diperhatikan untuk melakukan *debugging* pada tampilan *run and debug* kemudian *create a launch.json file* seperti . Tampilan kode *launch.json file* dapat dilihat pada Gambar 4.30 dan Gambar 4.31.

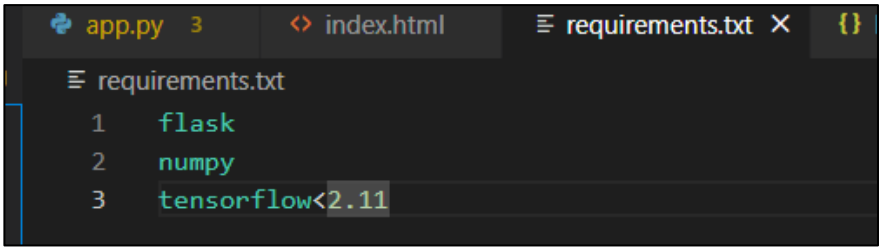


Gambar 4.30 *debugging* pembuatan *file launch.json*



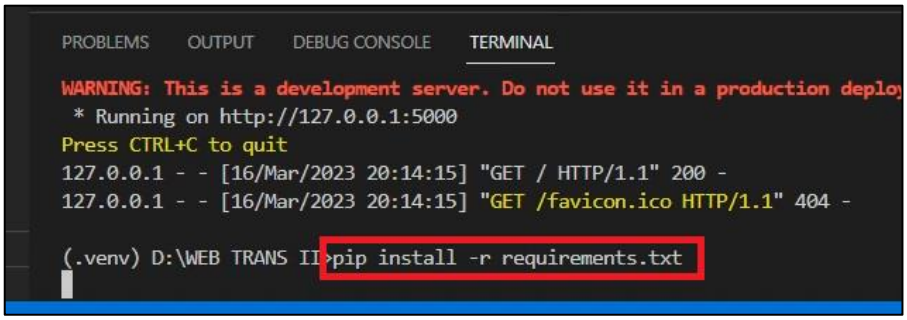
Gambar 4.31 Tampilan *file launch.json*

Selanjutnya dalam proses *flask* membutuhkan modul-modul yang harus digunakan agar program *flask* dapat berjalan sesuai dengan kebutuhan dengan menggunakan *Python requirements file*. *Python requirements file* merupakan teks sederhana yang akan menyimpan daftar modul dan paket yang dibutuhkan pada *Python* yang berfungsi melacak modul *Python* dan menyederhanakan pemasangan modul yang akan digunakan yaitu hanya cukup dengan cara menginstal atau memperbarui modul yang dibutuhkan. Implementasinya pada VS Code adalah dengan cara membuat *file* baru dengan format *requirements.txt* di dalam folder yang digunakan kemudian impor modul yang akan digunakan dan pastikan simpan *requirements.txt* pada direktori yang sama dengan *app.py*. Implementasi pembuatan *requirements.txt* dapat dilihat pada Gambar 4.32.



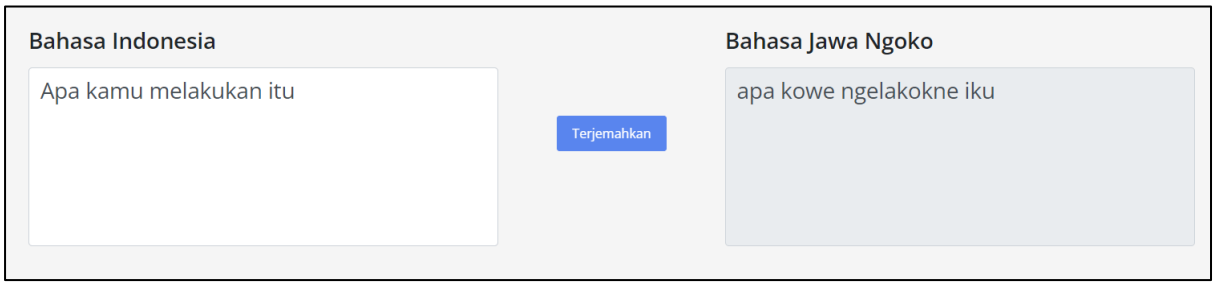
Gambar 4.32 Pembuatan *Python requirements file*

Instal modul yang dibutuhkan dengan cara ketikkan perintah pada Gambar 4.33 dan tunggu hingga paket selesai diinstal.



Gambar 4.33 Instal paket modul *Python requirements*

Langkah terakhir yaitu membuat indeks HTML untuk tampilan *form input* pada laman *website*. Bentuk tampilan dari *form input* untuk mesin translasi bahasa Indonesia ke bahasa Jawa Ngoko dapat dilihat pada Gambar 4.34.



Gambar 4.34 Tampilan *form input* mesin translasi

.

4.9 Pengujian

Pengujian nilai dari hasil translasi akan dilaksanakan dengan tiga metode, pertama dengan melakukan pengujian pada sistem yang dibangun, kedua dengan melakukan

perhitungan nilai hasil evaluasi secara otomatis dengan *Bilingual Evaluation Understudy* (BLEU), dan ketiga pengujian dilakukan oleh responden berjumlah 30 orang.

4.9.1 Pengujian pada Sistem yang Dibangun

Perangkat lunak akan dibuat berbasis *website* menggunakan bahasa pemrograman *Python* dan sebagai menjembatani *softwarenya* dengan menggunakan *Framework flask*. Pelaksanaan pengujian dilakukan dengan menggunakan *software* yang sudah dibangun sebagai perantara pengguna untuk menguji model yang telah diimplementasikan. Adapun hasil dari pelaksanaan pengujiannya dapat dilihat pada Gambar 4.35 dan Gambar 4.36.

Gambar 4.35 Hasil pengujian pada sistem berdasarkan inputan

Gambar 4.36 Hasil pengujian pada sistem berdasarkan dataset

Berdasarkan hasil dari pelaksanaan pengujian yang disajikan pada Gambar 4.35 dan Gambar 4.36 dapat didapati bahwa hasil translasi pada Gambar 4.35 mendekati dari hasil yang sebenarnya dan hasil translasi pada Gambar 4.36 masih jauh dari pengujian berdasarkan inputan yang sebenarnya. Dengan demikian model yang dirancang belum mampu untuk melakukan translasi secara akurat.

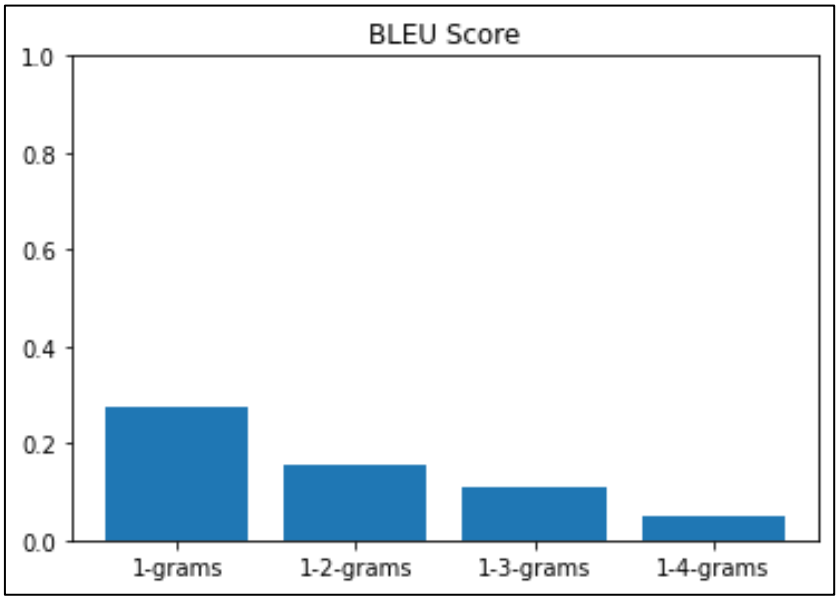
4.9.2 Perhitungan Nilai *Bilingual Evaluation Understudy* (BLEU)

Sistem pengujian *Bilingual Evaluation Understudy* (BLEU) merupakan algoritma sederhana guna untuk mengevaluasi kualitas hasil dari sebuah terjemahan mesin translasi

terhadap terjemahan manusia. Sistem ini bekerja secara otomatis dengan mengukur modifikasi skor presisi *n-gram* antara hasil terjemahan mesin translasi terhadap terjemahan manusia dengan brevity penalty sebagai tetapannya (Papineni, 2002). Perhitungan dilakukan dengan cara mencocokan panjang sebuah kalimat *output* dari mesin penerjemah dengan sumber acuan dari terjemahan. Parameter dari hasil pengujian ini akan berguna untuk menentukan tingkat ketepatan akurasi dalam mentranslasikaan. Pengujian ini berlangsung secara otomatis dengan melakukan langkah translasi terhadap korpusnya dan akan memberikan keluaran atau *output* berupa korpus dalam bahasa Jawa Ngoko. Langkah berikutnya adalah melakukan pengujian otomatis dengan BLEU dan banyaknya korpus yang dipakai pada pelaksanaan pengujian tahap ini berjumlah 5.000. Perhitungan nilai BLEU dilakukan untuk mengevaluasi hasil translasi dari model yang sudah dirancang dan dilatih. Hasil perhitungan nilai BLEU dapat dilihat pada Tabel 4.10 dan grafik nilai BLEU dilihat pada Gambar 4.37.

Tabel 4.10 Hasil perhitungan nilai BLEU

1-*grams*	**2-*grams***	**3-*grams***	**4-*grams***	**BLEU**
0,28	0,16	0,11	0,05	0,15



Gambar 4.37 Grafik nilai BLEU

Berdasarkan hasil perhitungan nilai BLEU, model yang dirancang memiliki nilai 28% pada 1-*grams*, namun selanjutnya mengalami penurunan sesuai dengan pertambahan *n-grams*

hingga 4-*grams* memiliki nilai sebesar 5%. Rata-rata nilai BLEU yang diperoleh sebesar 15%. Hasil dalam pelaksanaan penelitian ini menunjukkan keberhasilan dari model yang dirancang mempunyai nilai BLEU yang kecil dan belum mampu mencapai nilai yang dapat dipahami hasil translasinya.

4.9.3 Pengujian oleh Responden

Pengujian oleh responden dilakukan untuk mengetahui keakuratan dan kecocokan antara hasil translasi dari model yang dibuat dengan hasil translasi langsung oleh responden yang memakai bahasa Jawa Ngoko dalam kesehariannya. Pengujian dilakukan dengan mengambil sampel sebanyak 30 responden yang diambil secara acak. Responden yang dipilih merupakan mahasiswa dari lima universitas dan anggota keluarga di rumah. Kriteria penilaian hasil translasi oleh responden tergantung dari ketepatan hasil translasinya. Hasil dari pengujian oleh responden dapat dilihat pada Tabel 4.11.

Tabel 4.11 Hasil pelaksanaan pengujian oleh responden

Asal Responden	Jumlah Responden	Hasil
Mahasiswa/i Amikom	7	Kurang tepat
Mahasiswa/i STIPARI	4	Kurang tepat
Mahasiswa/i UII	10	Kurang tepat
Mahasiswa/i UGM	2	Kurang tepat
Mahasiswa/i YKPN	1	Kurang tepat
Anggota keluarga	6	Kurang tepat

Berdasarkan hasil pengujian oleh responden, semua responden mengatakan bahwa hasil translasi masih kurang tepat, seperti masih terdapat kata yang tidak sesuai dengan translasi yang sebenarnya dan masih terdapat kata yang tidak memiliki translasi sama sekali. Berikut tabel hasil translasi yang dihasilkan dari pengujian.

Tabel 4.12 Hasil translasi pengujian oleh responden

No	*Input*	*Output*	Keterangan
1	Apa kamu suka makan nasi	Apa kowe kudu golek suket	Kurang tepat
2	Apa yang sedang kamu lakukan	Apa sing kepriye apa iki	Kurang tepat
3	Aku tidak punya rumah	Kabayan ora bisa mlebu	Kurang tepat

4	Tidak ada yang tahu	Ora ana ing desa iki	Kurang tepat
5	Apa kamu tahu apa yang harus dia lakukan	Apa apa kowe sing wis dituku	Kurang tepat
6	Apakah kamu sudah makan tadi malam	Wis wis suwi maneh nyi	Kurang tepat
7	Aku suka kamu	kabayan	Kurang tepat
8	Aku tidak suka kamu	Kabayan ora bisa apapapa	Kurang tepat
9	Apa kamu melakukan itu	Apa kowe iki	Kurang tepat
10	Saya ingin makan buah	Aku dhewe bakal manik	Kurang tepat

Hasil pengujian data penerjemahan yang dilakukan pada Tabel 4.12 semua hasil translasi masih kurang tepat. Langkah perbaikan yang dilakukan seperti pengubahan model tapi masih menggunakan proses RNN dan LSTM. Perubahan model masih belum mengeluarkan hasil yang baik. Untuk hasil translasi yang lebih lengkap dapat dilihat melalui hasil A terdapat pada lampiran hasil.

4.10 Perbaikan Model

Langkah yang diambil selanjutnya adalah melakukan perubahan struktur dalam perbaikan dan modifikasi dataset. Perubahan dataset ini dikarenakan dataset yang digunakan sebelumnya cenderung lebih sedikit dan tidak terstruktur yang menyebabkan mesin kurang dalam mempelajari dari dataset tersebut. Agar mendapatkan dataset yang maksimal peneliti menggunakan dataset yang berbeda dari sebelumnya yaitu menggunakan dataset modifikasi dan kombinasi dengan cara memadukan dataset yang disediakan dari *ManyThings.org* dan website *Academia* berisikan kamus bahasa Indonesia ke bahasa Jawa, menambahkan juga dari bahasa jawa yang sering peneliti pakai ketika berkomunikasi sehari-hari, dan peneliti juga menambahkan kosakata dan percakapan yang sering digunakan di lingkungan sekitar. Untuk dataset lengkapnya dari dataset B terdapat pada lampiran dataset. Jumlah data translasi bahasa Indonesia ke bahasa Jawa Ngoko yang tersedia terdapat 10.310 data dengan kalimat terpanjang yaitu 25 kata pada dataset dan akan menggunakan seluruh datanya. Hasil dari dataset yang dikumpulkan dari berbagai sumber tersebut akan dijadikan acuan dataset terbaru yang siap digunakan dalam proses mesin translasi.

Langkah berikutnya adalah percobaan penggantian model yang digunakan dalam proses mesin translasi dengan menggunakan model RNN sederhana, RNN dengan *Embedding*, *Bidirectional* RNN, *Optional Encoder-Decoder* RNN. Akan tetapi hasil data *training* dan

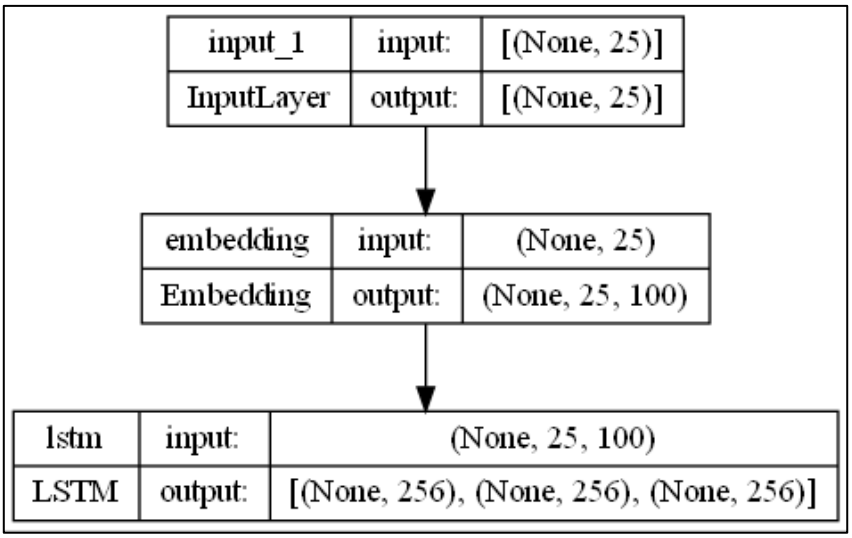
hasil dari translasi bahasa Indonesia ke bahasa Jawa masih meleset jauh dan tingkat akurasi yang didapatkankan lebih rendah dari penggunaan dataset sebelumnya. Maka dari itu proses implementasi mesin translasi yang dilakukan masih sama seperti yang digunakan sebelumnya dengan menggunakan metode dan model yang sama dikarenakan penggunaan beberapa model yang berbeda untuk dicoba dalam rangka memperbaiki hasil output masih tetap kurang tepat bahkan lebih buruk.

Pada percobaan *training* dari dataset sebelumnya telah dilakukan dan disimpulkan bahwa semakin besar nilai *epochs*, maka nilai Nilai validasi akurasi dari hasil evaluasi dataset *training* akan meningkat dan konvergen. Nilai validasi akurasi yang digunakan dalam *training* dataset kali ini menggunakan acuan hasil dari nilai validasi terendah, rata-rata, tertinggi, dan uji coba *training* dataset terbaru dengan nilai *epochs* sebesar 256 dan *batch size* sebesar 128. Berdasarkan nilai validasi akurasi terbaru didapatkan hasil evaluasi dataset *training* terendah yang didapat sebesar 89.79% dan nilai tertinggi sebesar 97.40% dengan proses validasi akurasi pada dataset *training* sebesar 0.7462, maka diambil nilai *epochs* sebesar 256 dan *batch size* sebesar 128 untuk menjadi acuan model dari hasil *training* yang akan digunakan. Percobaan yang dilakukan sebanyak 4 kali untuk mendapatkan nilai validasi tertinggi, rincian perbandingan hasil validasi, nilai *epochs* dan *batch size* dari dataset sebelumnya dan yang terbaru yang digunakan dapat dilihat pada Tabel 4.13.

Tabel 4.13 Perbandingan validasi akurasi hasil *training* dataset sebelumnya dan terbaru

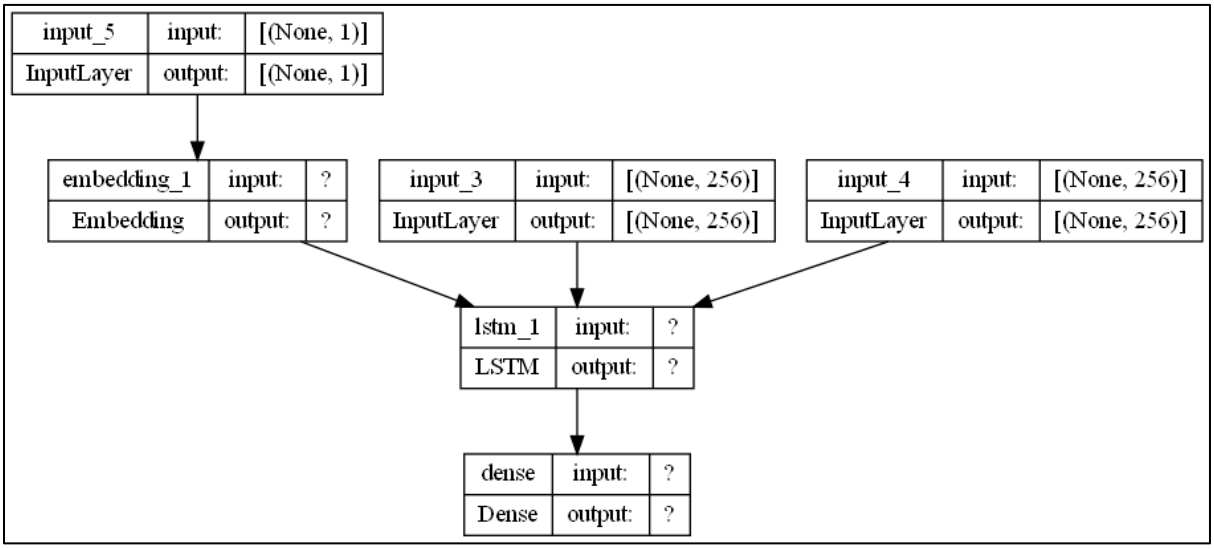
Dataset Sebelumnya			**Dataset Terbaru**		
Batch Size	***Epochs***	**Validasi Akurasi**	***Batch Size***	***Epochs***	**Validasi Akurasi**
64	8	75,26%	64	8	89,79%
128	128	97,84%	128	128	97,39%
98	54	97,85%	98	54	97,34%
-	-	-	128	256	97,40%

Inference encoder model yang dihasilkan memiliki tiga lapisan yaitu *input layer* sebagai lapisan input, *embedding layer* sebagai lapisan embedding, dan LSTM *layer* sebagai lapisan LSTM. Berikut hasil alur model dari *encoder* dihasilkan oleh inference model. Berikut hasil alur model dari *encoder* dihasilkan oleh *inference model* dari hasil *training* dataset terbaru dapat dilihat pada Gambar 4.38.



Gambar 4.38 Alur *Inference Encoder Model* dengan menggunakan dataset terbaru

Inference decoder model yang dihasilkan memiliki empat lapisan yaitu *input layer* sebagai lapisan input, *embedding layer* sebagai lapisan embedding, LSTM *layer* sebagai lapisan LSTM, dan *dense layer* sebagai lapisan padat. Berikut hasil alur model dari *encoder* dihasilkan oleh inference model. Berikut hasil alur model dari *decoder* dihasilkan oleh inference model dari hasil *training* dataset terbaru dapat dilihat pada Gambar 4.39.



Gambar 4.39 Alur *Inference Decoder Model* dengan menggunakan dataset terbaru

Walaupun hasil dari validasi akurasi dan tingkat akurasi *training* lebih rendah dari hasil sebelumnya, penggunaan dataset jauh lebih terstruktur dan memiliki lebih banyak data. Perubahan struktur dalam perbaikan dan modifikasi dataset akan mempengaruhi proses

translasi pada mesin penerjemah yang membuat output semakin membaik seiring perbaikan dan modifikasi dataset yang telah dilakukan. Berikut Tabel 4.14 hasil translasi yang dihasilkan dari pengujian dengan dataset terbaru dengan sampel pengujian dataset sebelumnya. Untuk hasil translasi yang lebih lengkap dapat dilihat melalui hasil B terdapat pada lampiran hasil.

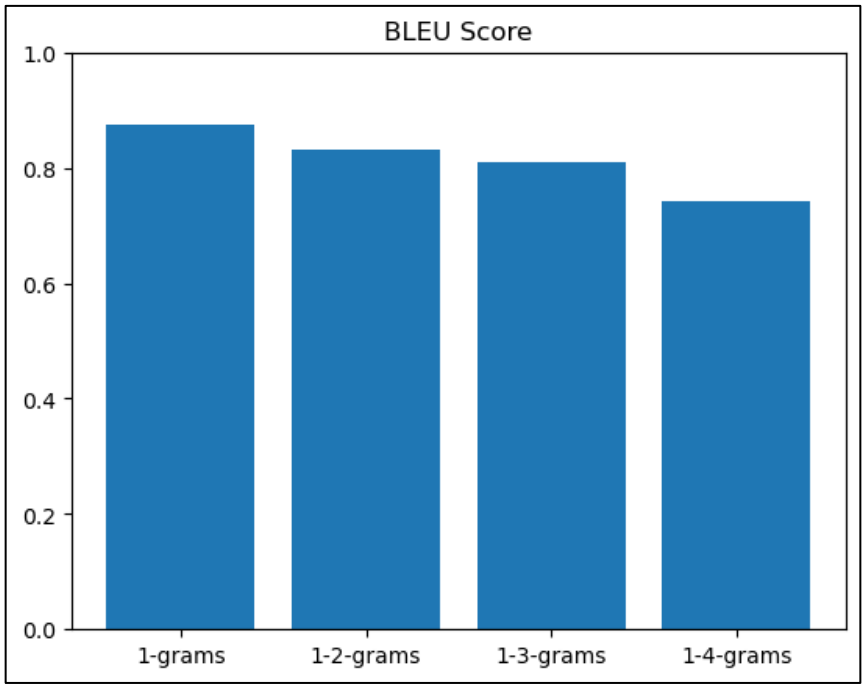
Tabel 4.14 Hasil translasi pengujian oleh responden dengan menggunakan dataset terbaru

No	***Input***	***Output***	**Keterangan**
1	Apa kamu suka makan nasi	Apa kowe seneng mangan	Kurang tepat
2	Apa yang sedang kamu lakukan	Apa sing kowe lakoni	Tepat
3	Aku tidak punya rumah	Aku ora duwe	Kurang tepat
4	Tidak ada yang tahu	Ora ana sing ngerti	Tepat
5	Apa kamu tahu apa yang harus dia lakukan	Apa kowe ngerti apa sing kudu dhewekke lakoni	Tepat
6	Apakah kamu sudah makan tadi malam	Apa kowe wes mangan nedha mau	Kurang tepat
7	Aku suka kamu	Aku seneng kowe	Tepat
8	Aku tidak suka kamu	Aku ora seneng kowe	Tepat
9	Apa kamu melakukan itu	Apa kowe ngelakokne iku	Tepat
10	Saya ingin makan buah	Aku arep mangan	Kurang tepat
11	Aku sudah sangat lelah sekali	Aku wis kesel banget	Tepat

Langkah berikutnya adalah melakukan pengujian otomatis dengan BLEU dan banyaknya korpus yang dipakai pada pelaksanaan pengujian tahap ini berjumlah 10.310. dataset baru akan mempengaruhi hasil dikarenakan konsep dari perhitungan BLEU adalah semakin banyak terjemahan sumber per kalimatnya maka akan semakin tinggi pula dalam memperoleh nilainya. Hasil perhitungan nilai BLEU dengan dataset terbaru dapat dilihat pada Tabel 4.15 dan grafik peningkatan nilai BLEU dapat dilihat pada Gambar 4.40.

Tabel 4.15 Hasil perhitungan nilai BLEU dengan menggunakan dataset terbaru

1-*grams*	**2-*grams***	**3-*grams***	**4-*grams***	**BLEU**
0,87	0,83	0,81	0,74	0,81



Gambar 4.40 Grafik nilai BLEU dengan menggunakan dataset terbaru

Berdasarkan hasil perhitungan nilai BLEU, model yang dirancang dengan menggunakan dataset yang terbaru memiliki peningkatan hingga 66% dari hasil yang telah dilakukan menggunakan dataset sebelumnya. Peningkatan n-*gram*s yang didapatkan yaitu 87% pada 1-*grams*, 83% pada 2-*grams*, 81% pada 3-*grams*, dan 74% pada 4-*grams*. Rata-rata nilai BLEU yang diperoleh sebesar 81%. Hasil dalam pelaksanaan penelitian ini menunjukkan keberhasilan adanya peningkatan dari model dengan memakai dataset yang terbaru yang telah dirancang menghasilkan nilai BLEU yang mampu merubah dari nilai yang dapat dipahami hasil translasinya.

4.11 *Confusion Matrix*

Implementasi *confusion matrix* menggunakan data hasil dari translasi mesin penerjemah yang telah dibuat oleh peneliti saat ini dengan asumsi data sebanyak 30 hasil translasi. Untuk data hasil pengklasifikasi *confusion matrix* dapat dilihat melalui hasil C terdapat pada lampiran hasil. Dari model yang diklasifikasikan 18 hasil translasi di antaranya melakukan hasil translasi yang tepat dan 12 mengalami hasil yang kurang tepat dalam menerjemahkan bahasa Indonesia ke bahasa Jawa Ngoko. Tabel 4.16 menampilkan matriks dari *confusion matrix* dari hasil translasi.

Tabel 4.16 *Confusion matrix* dari data hasil translasi

n = 30	Aktual Positif (1)	Aktual Negatif (0)
Prediksi Positif (1)	*True Positive*: 18	*False Positive*: 3
Prediksi Negatif (0)	*False Negative*: 5	*True Negative*: 4
	23	7

Peneliti melakukan perhitungan berbagai jenis matriks evaluasi klasifikasi seperti akurasi, presisi, *recall*, dan F1-*score* dengan data *confusion matrix* dari hasil translasi di atas. Hasil yang diperoleh dalam perhitungan akurasi sebesar 73,4% untuk mengukur seberapa akurat model dalam memprediksi kelas yang benar, 85,7% pada perhitungan presisi sebagai pengukur seberapa banyak data positif yang diprediksi benar, 78,2% pada *Recall* atau *sensitivity* untuk mengukur seberapa banyak data positif yang diidentifikasi secara benar dari total data positif yang seharusnya teridentifikasi, dan 81,7% dari hasil perhitungan *F1score* sebagai nilai rata-rata harmonik antara presisi dan *recall.*

Hasil dari perhitungan *confusion matrix* pada kinerja model mesin translasi dalam mengklasifikasikan informasi tentang berapa banyak sampel yang diklasifikasikan dengan benar dan yang salah dalam masing-masing kelas. Dengan demikian, penggunaan *confusion matrix* sangat penting dalam proses evaluasi dan pengembangan model klasifikasi, sehingga dapat memastikan model yang dihasilkan dapat memberikan prediksi yang akurat dan dapat diandalkan.

BAB V
KESIMPULAN DAN SARAN

5.1 Kesimpulan

Kesimpulan yang diperoleh berdasarkan proses pelaksanaan penelitian yang telah dilakukan adalah sebagai berikut:

a. Hasil translasi bahasa Indonesia ke bahasa Jawa Ngoko menggunakan model jaringan saraf berulang atau *Recurrent Neural Network* (RNN) dan memori jangka panjang-pendek atau *Long Short-Term Memory* (LSTM) belum mampu untuk mentranslasi secara akurat dikarenakan kekurangan dari model ini sendiri bahwa ia tidak efektif untuk data yang memiliki dimensi yang sangat besar, data yang kurang terstruktur dan data kurangnya data yang dimiliki.
b. Penerapan LSTM pada RNN untuk translasi diterapkan pada waktu pembuatan model dan inference model.
c. Hasil perhitungan nilai *Bilingual Evaluation Understudy* (BLEU) yang dirancang dengan menggunakan dataset yang terbaru memiliki peningkatan hingga 66% dari hasil yang telah dilakukan menggunakan dataset sebelumnya. Peningkatan n-*gram*s yang didapatkan yaitu 87% pada 1-*grams*, 83% pada 2-*grams*, 81% pada 3-*grams*, dan 74% pada 4-*grams*. Rata-rata nilai BLEU yang diperoleh sebesar 81%. Hasil penelitian ini menunjukkan bahwa model dengan menggunakan dataset yang terbaru yang dirancang memiliki peningkatan nilai BLEU yang mampu mengubah dari nilai yang dapat dipahami hasil translasinya.
d. Dengan menggunakan *confusion matrix* masing-masing kelas memperoleh 73,4% sebagai nilai akurasi, 85,7% sebagai nilai presisi, 78,2% sebagai nilai *recall* atau *sensitivity*, dan 81,7% sebagai nilai F1-*score*, peneliti dapat mengevaluasi dan memahami kinerja model klasifikasi, mengidentifikasi kesalahan prediksi, dan memperbaiki model untuk meningkatkan kinerjanya. *confusion matrix* juga dapat membantu peneliti dalam memilih model yang terbaik dan mengoptimalkan *hyperparameter* model.

5.2 Saran

Beberapa saran dari penelitian ini berdasarkan hasil proses pelaksanaan penelitian dan kesimpulan di atas adalah dengan menerapkan pada tahap *tokenization* selain menggunakan *library* dari *tensorflow keras*, seperti menggunakan *IndoBERT* dan *IndoGPT* dan memperbaiki struktur dari dataset yang akan digunakan pada mesin penerjemah dari bahasa Indonesia ke bahasa Jawa Ngoko karena sangat berpengaruh pada prediksi ditahap *one hot vector decoder*.

DAFTAR PUSTAKA

Abidin, Z., Sucipto, A., & Budiman, A. (2018). Penerjemahan Kalimat Bahasa Lampung - Indonesia dengan Pendekatan Neural Machine Translation Berbasis Attention. *Jurnal Kelitbangan Vol. 06 No. 02*, 191-206.

Budaya, I. G., Kesiman, M. W., & Sunarya, I. M. (2022). Perancangan Mesin Translasi berbasis Neural dari Bahasa Kawi ke dalam Bahasa Indonesia menggunakan Microframework Flask. *JURNAL SISTEM DAN INFORMATIKA (JSI)*, 94-103.

Datta, D., David, P. E., Mittal, D., & Jain, A. (2020). Neural Machine Translation using Recurrent. *International Journal of Engineering and Advanced Technology (IJEAT)*, 1395-1400.

Devianty, R. (2017). Bahasa Sebagai Cermin Kebudayaan. *Jurnal Tarbiyah Vol. 24 No. 2*, 226-245.

Fauziyah, Y., Ilyas, R., & Kasyidi, F. (2022). Mesin Penterjemah Bahasa Indonesia - Bahasa Sunda. *Jurnal Teknoinfo*, 313-322.

He, W., He, Z., Wu, H., & Wang, H. (2016). Improved Neural Machine Translation with SMT Features. *Thirtieth AAAI Conference on Artificial Intelligence*, 151-157.

Hermanto, A., Adji, T. B., & Setiawan, N. A. (2015). Recurrent Neural Network Language Model for English-Indonesian Machine Translation: Experimental Study. *International Conference on Science in Information Technology (ICSITech)*, 132-136.

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 1735–1780.

Indrayanto, B., & Yuliastuti, K. (2015). FENOMENA TINGKAT TUTUR DALAM BAHASA JAWA AKIBAT TINGKAT SOSIAL MASYARAKAT. *Magistra No. 91 Th. XXVII*, 37-44.

Khan, A., & Sarfaraz, A. (2019). RNN-LSTM-GRU based language transformation. *Soft Computing*.

Koehn, P., & Knowles, R. (2017). Six Challenges for Neural Machine Translation.

Laskar, S. R., Dutta, A., Pakray, P., & Bandyopadhyay, S. (2019). Neural Machine Translation: English to Hindi. *IEEE Conference on Information and Communication Technology (CICT)*.

Meng, F., Lu, Z., Li, H., & Liu, Q. (2016). Interactive Attention for Neural Machine Translation.

Muhid, A. (2011). Tingkat Tutur Bahasa Jawa Masyarakat Samin Desa Klopoduwur Kabupaten Blora. *Majalah Ilmiah INFORMATIKA Vol. 2 No. 1 (2011)*, 82-103.

Na'im, A., & Syaputra, H. (2010). *Kewarganegaraan, Suku Bangsa, Agama dan Bahasa Sehari-hari Penduduk Indonesia Hasil Sensus Penduduk 2010.* Jakarta: Badan Pusat Statistik.

Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: a Method for Automatic Evaluation of Machine Translation. *ACL '02: Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, 318-318.

Poedjasoedarma, S., Kundjana, T., Soepomo, G., & Suharso, A. (1979). *Tingkat Tutur Bahasa Jawa.* Jakarta: Pusat Pembinaan dan Pengembangan Bahasa Departemen Pendidikan dan Kebudayaan.

Salehinejad, H., Sankar, S., Barfett, J., Colak, E., & Valaee, S. (2018). Recent Advances in Recurrent Neural Networks. *Neural and Evolutionary Computing*.

Sasangka, & Wisnu, S. S. (2009). *Unggah Ungguh Bahasa Jawa (Editor: Yeyen Maryani).* Jakarta: Yayasan Paramalingua.

Sennrich, R., Haddow, B., & Birch, A. (2016). Improving Neural Machine Translation Models with Monolingual Data.

Sennrich, R., Haddow, B., & Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units.

Septarina, A. A., Rahutomo, F., & Sarosa, M. (2019). Machine Translation of Indonesian: A Review. *Communications in Science and Technology*, 12-19.

Sherstinsky, A. (2020). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) network. *Physica D: Nonlinear Phenomena*.

Tiwari, G., Sharma, A., Sahotra, A., & Kapoo, R. (2020). *International Conference on Communication and Signal Processing*, 871-875.

Wang, X., Tu, Z., Xiong, D., & Zhang, M. (2017). Translating Phrases in Neural Machine Translation.

Wilian, S. (2006). Tingkat tutur dalam bahasa Sasak dan bahasa Jawa. *Wacana Vol. 8 No. 1*, 32-53.

S. Raschka and V. Mirjalili, “Python machine learning : machine learning and deep learning with python, scikit-learn, and tensorflow 2,” *Int. J. Knowledge-Based Organ.*, vol. 11, no. 1, p. 741, 2021.

D. J. & J. H. Martin., “Speech and Language Processing: An introduction to natural language processing,” *SPEECH Lang. Process. An Introd. to Nat. Lang. Process. Comput. Linguist. Speech Recognit.*, pp. 1–18, 2001, [Online]. Available: http://www.cs.colorado.edu/~martin/slp.html.

Yoav Goldberg, “A Primer on Neural Network Models for Natural Language Processing,” *J. Artif. Intell. Res.*, vol. 57, pp. 345–420, 2016, [Online]. Available: http://www.jair.org/papers/paper4992.html.

I. G. Bintang, A. Budaya, M. Windu, A. Kesiman, and I. M. G. Sunarya, “Perancangan Mesin Translasi berbasis Neural dari Bahasa Kawi ke dalam Bahasa Indonesia menggunakan Microframework Flask,” pp. 94–103, 2022.

M. Y. Aristyanto and R. Kurniawan, “Pengembangan Metode Neural Machine Translation Berdasarkan Hyperparameter Neural Network,” *Semin. Nas. Off. Stat.*, vol. 2021, no. 1, pp. 935–946, 2021, doi: 10.34123/semnasoffstat.v2021i1.789.

W. Gunawan, H. Sujaini, and T. Tursina, “Analisis Perbandingan Nilai Akurasi Mekanisme Attention Bahdanau dan Luong pada Neural Machine Translation Bahasa Indonesia ke Bahasa Melayu Ketapang dengan Arsitektur Recurrent Neural Network,” *J. Edukasi dan Penelit. Inform.*, vol. 7, no. 3, p. 488, 2021, doi: 10.26418/jp.v7i3.50287.

LAMPIRAN

A. Lampiran dataset

dataset A https://drive.google.com/file/d

dataset B https://docs.google.com/spreadsheets/d

B. Lampiran hasil

hasil A https://docs.google.com/spreadsheets/d

hasil B https://docs.google.com/spreadsheets/d

hasil C https://docs.google.com/spreadsheets/d

C. Lampiran *Website* yang digunakan

https://deepai.org/machine-learning-glossary-and-terms/transformer-neural-network

https://towardsdatascience.com/animated-rnn-lstm-and-gru-ef124d06cf45

https://towardsdatascience.com/foundations-of-nlp-explained-bleu-score-and-wer-metrics-1a5ba06d812b

https://www.kaggle.com/code/databeru/machine-translation-fr-en-with-bleu-score

https://machinelearningmastery.com/calculate-bleu-score-for-text-python/

https://github.com/herrysujaini/korpusnusantara/blob/main/korpus%20nusantara.xlsx

https://github.com/IndoNLP/nusax/tree/main/datasets/mt

https://github.com/dindainastra/indowikiparalelcorpora/tree/main/manualsets

http://www.manythings.org/bilingual/

https://www.academia.edu/32630591/KAMUS_BASA_JAWA_INDONESIA

https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

https://socs.binus.ac.id/2020/11/01/confusion-matrix/

https://towardsdatascience.com/accuracy-recall-precision-f-score-specificity-which-to-optimize-on-867d3f11124

https://machinelearningmastery.com/confusion-matrix-machine-learning/

https://books.google.co.id/books?hl=en&lr=&id=sKXIDwAAQBAJ&oi=fnd&pg=PP1&dq=Raschka,+S.,+%26+Mirjalili,+V.+(2021).+Python+Machine+Learning,+Third+Edition.+Packt+Publishing+Ltd.&ots=V9KknOUJJm&sig=BAZBSI2ZjB6JqKfCRCrGPZYqscU&redir_esc=y#v=onepage&q&f=false