Implementasi Arsitektur *Transformers* Penerjemahan Otomatis Bahasa Indonesia ke Bahasa Papua (Studi kasus Daerah Kokas)

TUGAS AKHIR

Disusun oleh:
Ronggo Haikal
191111014

PROGRAM STUDI INFORMATIKA
SEKOLAH TINGGI INFORMATIKA & KOMPUTER INDONESIA

2024


SEKOLAH TINGGI INFORMATIKA & KOMPUTER INDONESIA
STIKI

Implementasi Arsitektur *Transformers* Penerjemahan Otomatis Bahasa Indonesia ke Bahasa Papua (Studi Kasus Kokas)

TUGAS AKHIR
Sebagai salah satu syarat untuk
memperoleh gelar sarjana
pada Program Studi Informatika

Disusun oleh:
Ronggo Haikal
191111014

PROGRAM STUDI INFORMATIKA

SEKOLAH TINGGI INFORMATIKA & KOMPUTER INDONESIA
2024

PERNYATAAN
ORISINALITAS TUGAS AKHIR

Saya yang bertandatangan dibawah ini:

Nama	: Ronggo Haikal
NRP	: 191111014
Program Studi	: Informatika
Jenjang Studi	: S1

Dengan ini saya menyatakan bahwa:

1. Tugas akhir ini murni ide, rumusan dan penelitian sendiri, tanpa bantuan dari pihak manapun selain Dosen Pembimbing.
2. Tugas akhir ini belum pernah digunakan untuk memperoleh gelar sarjana, baik di Sekolah Tinggi Informatika & Komputer Indonesia atau di perguruan tinggi lain.
3. Tugas akhir ini tidak memuat karya atau pendapat yang ditulis atau diterbitkan oleh pihak ketiga, kecuali secara tertulis dengan mencantumkan sebagai acuan dalam naskah dengan menyebutkan nama pengarang dan mencantumkan dalam daftar pustaka.

Demikian pernyataan ini dibuat dengan sesungguhnya dan jika dikemudian hari terbukti ada unsur-unsur plagiasi, maka saya bersedia menerima sanksi akademik yakni pencabutan gelar yang sudah diberikan melalui karya tulis ini, dan sanksi lainnya sesuai dengan norma yang ada di perguruan tinggi.

Malang, <tanggal bulan tahun>
Yang menyatakan,

Meterai

Ronggo Haikal
191111014

TUGAS AKHIR BERJUDUL

Implementasi Arsitektur *Transformers* Penerjemahan Otomatis Bahasa Indonesia ke Bahasa Papua (Studi kasus Daerah Kokas)

Disusun oleh:
Ronggo Haikal
191111014

Telah dipertahankan dalam Sidang Tugas Akhir
pada tanggal ……..
dan dinyatakan telah memenuhi syarat untuk diterima

KOMISI SIDANG,	KOMISI PENGUJI,
Nira Radita, S.Pd., M.Pd Dosen Pembimbing	<Nama Lengkap> Ketua Penguji
Mukhlis Amien, S.Kom., M.Kom Co. Pembimbing	<Nama Lengkap> Anggota Penguji I
	<Nama Lengkap> Anggota Penguji II

Malang, …….
Sekolah Tinggi Informatika & Komputer Indonesia
Ketua

Dr. Eva Handriyantini, S.Kom., M.MT

PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademika Sekolah Tinggi Informatika & Komputer Indonesia, saya yang bertanda tangan di bawah ini:

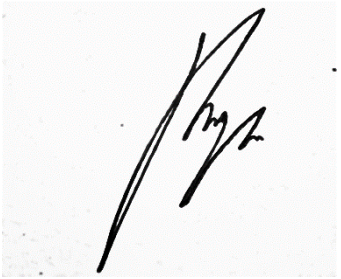
Nama	: Ronggo Haikal
NRP	: 191111014
Program Studi	: Informatika
Jenjang	: S1
Jenis Karya	: Pengembangan

Dengan ini saya menyatakan bahwa,
Demi pengembangan ilmu pengetahuan, setuju untuk memberikan Hak Bebas Royalti Non-Eksklusif kepada Sekolah Tinggi Informatika & Komputer Indonesia atas karya ilmiah saya yang berjudul:

Implementasi Arsitektur *Transformers* Penerjemahan Otomatis Bahasa Indonesia ke Bahasa Papua (Studi kasus Daerah Kokas)

dengan perangkat (jika diperlukan). Dengan hak bebas lisensi ini, Sekolah Tinggi Informatika & Komputer Indonesia berhak untuk menyimpan, mentransfer/memformat, mengelola, memelihara, dan mempublikasikan proyek yang telah selesai dalam format database dan tetap mencantumkan nama saya sebagai penulis dan pemegang hak cipta.
Demikian pernyataan ini dibuat dengan sebenar-benarnya.

Malang,
Yang menyatakan,



Ronggo Haikal
191111014

ABSTRAK

Ronggo Haikal, 2024. **Implementasi Arsitektur Transformers penerjemahan Otomatis Bahasa Indonesia ke Bahasa Papua Kokas**. Tugas Akhir, Program Studi Informatika (S1), Sekolah Tinggi Informatika & Komputer Indonesia, Pembimbing: Nira Radita, Co. Pembimbing: Mukhlis Amien

Kata kunci: Arsitektur Transformers, MarianMT, BLEU, Bahasa Papua Kokas, Penerjemahan otomatis

Penelitian ini bertujuan mengimplementasikan arsitektur Transformers untuk penerjemahan otomatis dari Bahasa Indonesia ke Bahasa Papua Kokas menggunakan framework MarianMT. Tantangan yang dihadapi adalah kurangnya sumber daya bahasa Papua Kokas yang mengancam kelestariannya. Dataset yang digunakan dalam penelitian ini terdiri dari 2908 pasangan kalimat. Model dilatih dengan teknik tokenisasi dan diuji menggunakan metrik BLEU. Evaluasi yang dilakukan menunjukkan bahwa model dapat menerjemahkan teks dengan akurasi yang memadai, di mana peningkatan skor BLEU terlihat pada setiap epoch. Model kemudian diimplementasikan dalam aplikasi berbasis web yang memungkinkan penerjemahan secara real-time. Hasil penelitian menunjukkan bahwa pendekatan arsitektur Transformers layak diterapkan untuk penerjemahan Bahasa Indonesia ke Bahasa Papua Kokas. Saran untuk pengembangan lebih lanjut termasuk memperluas dataset, mengeksplorasi teknik optimasi lain seperti TER dan METEOR, serta mengadaptasi model untuk bahasa daerah lain.

ABSTRACT

Ronggo Haikal, 2024. Implementation of Transformers Architecture for Automatic Translation of Indonesian to Papuan Language Kokas. Final Project, Study Program Informatika S1, Sekolah Tinggi Informatika & Komputer Indonesia, Advisor 1: Nira Radita, Advisor 2: Mukhlis Amien

Keyword: Transformers Architecture, MarianMT, BLEU, Papuan Kokas Language, Automatic translation

This study aims to implement the Transformers architecture for automatic translation from Indonesian to Papuan Kokas language using the MarianMT framework. The challenge faced is the lack of Papuan Kokas language resources that threaten its sustainability. The dataset used in this study consists of 2908 sentence pairs. The model was trained using tokenization techniques and tested using the BLEU metric. The evaluation conducted showed that the model can translate text with adequate accuracy, where an increase in the BLEU score is seen in each epoch. The model was then implemented in a web-based application that allows real-time translation. The results of the study indicate that the Transformers architecture approach is feasible to be applied for translation from Indonesian to Papuan Kokas language. Suggestions for further development include expanding the dataset, exploring other optimization techniques such as TER and METEOR, and adapting the model to other regional languages.

KATA PENGANTAR

Puji syukur kehadirat Allah SWT yang telah memberikan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan laporan tugas akhir yang berjudul **"Implementasi Arsitektur Transformers Penerjemahan Otomatis Bahasa Indonesia ke Bahasa Papua (Studi Kasus Daerah Kokas)"**. Laporan ini merupakan salah satu syarat untuk menyelesaikan program studi Informatika di Sekolah Tinggi Informatika & Komputer Indonesia.

Dalam tugas akhir ini, penulis membahas penerapan arsitektur Transformers untuk penerjemahan otomatis dari Bahasa Indonesia ke Bahasa Papua, khususnya dialek Kokas. Penelitian ini bertujuan untuk menjawab tantangan dalam pengembangan teknologi penerjemahan bagi bahasa-bahasa daerah yang masih minim penelitian. Implementasi arsitektur Transformers diharapkan mampu meningkatkan kualitas terjemahan secara otomatis untuk Bahasa Papua Kokas.

Pada kesempatan ini, penulis ingin menyampaikan rasa terima kasih yang sebesar-besarnya kepada pihak-pihak yang telah memberikan bimbingan dan dukungan selama penyusunan tugas akhir ini, di antaranya:

1. **Ibu Nira Radita, S.Pd., M.Pd.**, selaku dosen pembimbing pertama, yang telah memberikan bimbingan, arahan, serta masukan yang sangat berarti selama proses penyusunan tugas akhir ini.
2. **Bapak Mukhlis Amien, M.Kom.**, selaku dosen pembimbing kedua, yang senantiasa memberikan saran, dukungan, dan ilmu dalam penyempurnaan tugas akhir ini.

3. **Ibu Chaulina Alfianti Oktavia, S.Kom., M.T.**, selaku wali dosen, yang telah memberikan bimbingan akademik serta motivasi selama masa studi penulis di program studi Informatika.
4. Kedua orang tua penulis, yang selalu memberikan dukungan moral, spiritual, serta doa yang tiada henti, sehingga penulis dapat menyelesaikan tugas akhir ini dengan baik.

Penulis menyadari bahwa tugas akhir ini masih jauh dari kesempurnaan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi perbaikan di masa mendatang. Semoga tugas akhir ini bermanfaat bagi semua pihak, terutama dalam pengembangan teknologi penerjemahan bahasa daerah di Indonesia.

Kota Malang, ……..

Ronggo Haikal

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SEGMEN PROGRAM

DAFTAR LAMPIRAN

BAB I
PENDAHULUAN

1.1 Latar Belakang

Bahasa adalah suatu media komunikasi yang dimanfaatkan oleh manusia untuk mengungkapkan ide, pemikiran, konsep, atau emosi kepada orang lain. Menurut (Apriani et al., 2016). Bahasa memiliki fungsi khusus yang dipakai sesuai kebutuhan individu, seperti mengekspresikan diri, berkomunikasi, dan menyesuaikan diri dengan lingkungan sosial serta kondisi tertentu. Indonesia merupakan negara yang memiliki keragaman luar biasa dalam hal suku, budaya, dan bahasa. menurut Badan Bahasa Kementerian Pendidikan dan Kebudayaan RI, tercatat ada 718 bahasa daerah di seluruh wilayah Indonesia, dengan 90 persen di antaranya tersebar di kawasan timur. Di Papua sendiri dilaporkan terdapat 428 bahasa, sementara di Maluku ada 80 bahasa, di Nusa Tenggara Timur 72 bahasa, dan di Sulawesi sebanyak 62 bahasa. perbedaan bahasa antar suku ini dapat menjadi hambatan dalam komunikasi, karena individu dari suku lain mungkin tidak dapat memahami pesan yang disampaikan.

Perbedaan bahasa ini mendorong banyak pihak untuk menciptakan sistem penerjemahan, yang kemudian berkembang menjadi penerjemahan otomatis. terjemahan otomatis adalah proses mentranslasikan kalimat dari satu bahasa ke bahasa lain menggunakan teknologi komputer. Teknologi ini memungkinan komputer atau sistem AI untuk memahami dan menerjemahkan konten dari satu Bahasa ke Bahasa tanpa intervensi manusia.

Mesin penerjemah yang digunakan yaitu pemrosesan bahasa alami *Neural Language Processing* (NLP) dalam bidang *Machine Translation* (MT), MT melakukan sebuah konversi dari bahasa sumber organisasi atau bahasa yang digunakan dalam bahasa target. Menurut (Shah & Bakrola, 2019) dalam penerjemahan mesin, beberapa pendekatan yang umum digunakan yaitu *Statistical Machine Translation* (SMT), *Rule-based Machine Translation* (RMT), *Pharase-based Statistical* (PMT), serta *Neural Machine Translation* (NMT). Menurut (Abidin, 2017) NMT merupakan pendekatan paling sesuai untuk terjemahan otomatis. NMT adalah metode yang relatif baru dalam teknologi penerjemahan mesin, yang menggabungkan penggunaan *encoder* sebuah jaringan saraf tiruan yang mengubah bahasa sumber menjadi vektor dengan dimensi tetap dan *encoder*, yaitu jaringan saraf tiruan yang bertugas menghasilkan, terjemahan akhir dari vektor tersebut . Ada beberapa metode yang paling umum digunakan salah satunya yaitu metode Arsitektur *Transformers,* (Vaswani et al., 2017) Arsitektur *Transformers* adalah arsitektur terjemahan yang menerapkan mekanisme *attention* tanpa adanya tambahan *Recurrent Neural Network* (RNN) ataupun *Convolutional Neural Network* (CNN).

Dengan memperhatikan latar belakang ini, penulis membuat penerapan arsitektur *Transformers* untuk penerjemahan bahasa Indonesia ke bahasa daerah Papua (Studi kasus daerah kokas),masalah ini cukup krusial di daerah Kokas maka dari itu penelitian ini memiliki upaya untuk memfasilitasi komunikasi dan pengaksesan informasi bagi masyarakat yang berbicara dalam bahasa-bahasa daerah serta mempermudah bahasa yang digunakan pada daerah tersebut

dikarenakan tidak sedikit masyarakat menggunakan bahasa Kokas sehingga penulis ingin tetap melestarikan komunikasi Bahasa daerah tersebut.

1.2 Rumusan Masalah

Berdasar pada kajian-kajian yang telah ada, rumusan masalah dapat disusun yaitu bagaimana cara mengimplementasikan Arsitektur Transformers pada penerjemahan Bahasa Indonesia ke Bahasa Papua Kokas.

1.3 Tujuan

Tujuan dari penelitian yaitu mengimplementasikan Arsitektur Transformers pada penerjemahan Bahasa Indonesia ke Bahasa Papua Kokas serta mengetahui kualitas dan tingkat akurasi menggunakan algoritma BLEU.

1.4 Manfaat

Manfaat yang diharapkan pada peneliti

Memperluas wawasan dan keterampilan penulis dalam bidang *machine learning* serta berpotensi menjadi referensi bagi penelitian-penelitian selanjutnya yang memiliki kesaman dengan studi ini.

Manfaat yang diharapkan pada pengguna

Penerjemahan otomatis dari Bahasa Indonesia ke bahasa kokas dapat memperluas jangkauan akses masyarakat setempat terhadap informasi yang tersedia dalam Bahasa Indonesia. Fenomena ini memungkinkan implementasi teknologi sebagai sarana pendukung dalam mendorong proses pembelajaran, fasilitasi komunikasi, serta meningkatkan tingkat partisipasi masyarakat dalam era digital yang terus berkembang.

1.5 Batasan Masalah

Batasan masalah yang ditetapkan lingkup pembahasan ini mencangkup:

1. Metode ini hanya bisa menerjemahkan suatu bahasa ke bahasa lainnya yaitu bahasa Indonesia ke bahasa Papua Kokas;
2. Data yang digunakan yaitu 2908 kalimat bahasa Indonesia dan 2908 kalimat bahasa Papua Kokas dan kalimat tersebut berupa kalimat sehari-hari, yang dimana jumlah data yang dimiliki masih terbatas.
3. Evaluasi yang digunakan untuk mengetahui skor yaitu algoritma BLEU.

1.6 Metodologi Penelitian

1.6.1. Bahan dan Alat Penelitian

Di dalam pembuatan sebuah sistem tentunya diperlukan alat dan bahan yang digunakan untuk menunjang proses pengerjaan sistem tersebut. Dalam hal ini penulis menggunakan software yang berhubungan dengan pemrograman web sebagai berikut:

1. Hardware
   - Laptop
2. Software
   - Sistem Operasi *Microsoft Windows 11*
   - Browser menggunakan *Chrome*
   - Desain *prototype* menggunakan *Figma*
   - Bahasa pemrograman *Python* dan menggunakan *Framework Django*
   - Code Editor menggunakan *Visual Studio Code*

1.6.2. Pengumpulan Data dan Informasi

Dalam melakukan pengembangan sistem ini penulis menggunakan beberapa cara pengumpulan data yaitu:

1. *Interview* (wawancara)

   Melakukan wawancara dengan penduduk daerah kokas untuk mendapatkan serta mengumpulkan kalimat penerjemahan Bahasa Indonesia ke Bahasa kokas

2. Studi kepustakaan

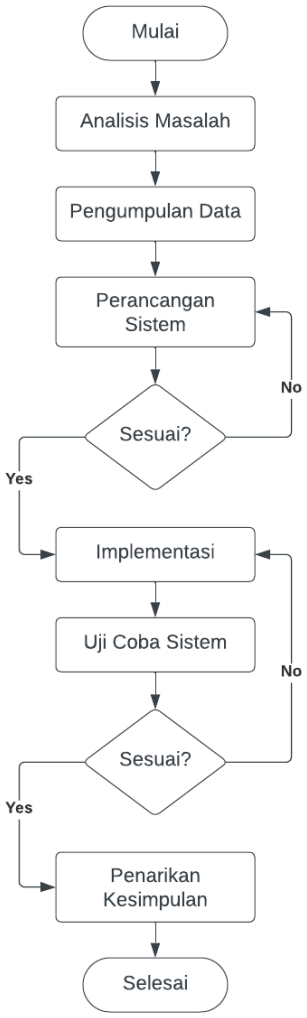
   Dengan berpedoman kepada buku – buku, literatur dan jurnal-jurnal yang berhubungan dengan Analisa dan metode yang akan digunakan sistem informasi.

1.6.3. Analisis Data

Dalam membuat sistem dan analisis ini peneliti menggunakan metode analisis Tekstual, yaitu proses sistematis untuk memahami dan mengekstrak makna dari data teks. Pada akhirnya peneliti menyimpulkan hasil dari analisis data sesuai dengan masalah yang dialami oleh objek penelitian

1.6.4. Prosedur Penelitian

Prosedur penelitian yang dilakukan sebagai berikut:



Gambar 1.1 Diagram Alir Penelitian

Analisis masalah

Pada tahap ini penulis perlu memahami beberapa masalah yang akan dihadapi dalam pembuatan sistem rekomendasi dan membuat pemecahan masalah agar memudahkan penulis dalam merancang solusi yang efektif saat pembuatan sistem.

Pengumpulan Data

Pada tahap ini penulis mengumpulkan data korpus teks dengan metode wawancara bersama masyarakat.

Perancangan Sistem

Pada tahap ini perancangan sistem dibuat untuk memastikan bahwa penelitian dilakukan secara sistematis dan sesuai dengan tujuan yang ditetapkan. Ini juga memberikan kerangka kerja yang memandu penelitian dari awal hingga akhir, sehingga mempermudah penulis untuk mengelola langkah-langkah penelitian dengan efisien dan efektif.

Implementasi

Setelah perancangan sistem maka penulis selanjutnya akan melakukan implementasi terhadap masalah yang dialami.

Uji Coba Sistem

Setelah melakukan implementasi, penulis melakukan uji coba terhadap hasil dari implementasi. Jika sistem belum sesuai dengan hasil perancangan, maka akan dilakukan revisi atau implementasi ulang.

Penarikan Kesimpulan

Setelah melakukan uji coba dan memastikan sistem berjalan dengan baik dan sesuai dengan perancangan, maka penulis dapat menarik kesimpulan.

1.7 Sistematika Penulisan

BAB I: PENDAHULUAN

Berisi tentang Latar Belakang, Rumusan Masalah, Tujuan, Manfaat, Batasan Masalah, dan Sistematika Penulisan.

BAB II: TINJAUAN PUSTAKA

Berisi tentang Tinjauan Empiris, Teori Teoritis, dan Hipotesis.

BAB III: METODOLOGI PENELITIAN

Berisi tentang Jenis Penelitian, Kerangka Konsep Penelitian, Pengumpulan Data, Definisi Operasional Variabel, Rancangan Pengujian, dan Teknik Analisis Data.

BAB IV: PEMBAHASAN

Berisi tentang Gambaran Umum Obyek Penelitian, Hasil Pengujian, Analisis Data, dan Pembahasan.

BAB V: PENUTUP

Berisi tentang Kesimpulan dan Saran.

BAB II
TINJAUAN PUSTAKA

2.1 Penelitian Terdahulu

Pada tahun 2021, Adriansyah Dwi Rendgraha, Moch. Arif Bijaksana, Ade Romadhony melakukan sebuah studi yang berjudul "Pendekatan Metode Transformers untuk Deteksi Bahasa Kasar dalam Komentar Berita Online Indonesia". Tujuan dari penelitian ini adalah untuk mengidentifikasi penggunaan bahasa kasar dalam komentar daring di Indonesia dengan memanfaatkan model transformers, khususnya *Bidirectional Encoder Representational form Transformers* (BERT), serta model BERT *Multilingual* yang telah dilatih sebelumnya sebagai baseline. Sistem ini menerima input berupa teks komentar dan menghasilkan label untuk menyusun komentar tersebut ke dalam kategori *Offensive*, Normal, atau *Non-Offensive* (Rendragraha et al., 2021). Penggunaan BERT memiliki kekurangan yaitu model ini tidak menggunakan *preprocessing* sehingga terjadi error dalam memahami sebuah teks. Atas dasar itu peneliti merancang sebuah metode yang menggabungkan *preprocessing* sehingga kata yang terdapat pada korpus teks dapat dipahami oleh model *transformers*.

Pada tahun 2023, I Nyoman Purnama, Ni Nengah Widya Utami melakukan penelitian yang berjudul "Implementasi Peringkas Dokumen Berbahasa Indonesia Menggunakan Metode *Text to Text Transfer Transformers* (T5)". Penelitian ini bertujuan untuk merangkum dokumen berita berbahasa Indonesia dengan menggunakan metode transformers T5. Dalam proses pengerjaan dibagi menjadi tiga skenario, dimana perbedaan utama antara masing-masing skenario terletak pada

tahap *preprocessing*. Pada skenario pertama, diterapkan proses *stemming* dan penghilangan kata-kata yang tidak penting (*stopwords removal*). Skenario kedua hanya menetapkan *stemming* tanpa menghapus kata-kata yang tidak penting, sedangkan pada skenario ketiga, tidak ada keduanya yang diterapkan (Purnama & Utami, 2023). Penggunan T5 memiliki kekurangan yaitu memerlukan *pre trained* berulang sehingga metode ini memerlukan waktu yang lama. Atas dasar itu sistem tersebut berbeda dengan sistem yang dirancang oleh penulis, dimana penulis merancang sistem yang berfokus pada penerjemahan bahasa Indonesia ke bahasa Papua daerah Kokas menggunakan Metode Arsitektur *Transformers* melakukan *pre-trained* dengan waktu yang tidak lama.

Pada tahun 2017, Zaenal Abidin melakukan penelitian yang berjudul "Penerapan *Neural Machine Translation* untuk Eksperimen Penerjemahan Secara Otomatis pada Bahasa Lampung-Indonesia" Penelitian ini bertujuan untuk menghasilkan 3000 kalimat paralel dalam bahasa Lampung (dialek api) dan bahasa Indonesia, serta menetapkan parameter model NMT untuk proses pelatihan data. Tahapan selanjutnya mencangkup pembangunan model (NMT) dan pengujian performanya. *Neural Machine Translation* (NMT) merupakan pencapaian inovatif pada teknologi penerjemahan mesin dengan menggabungkan penggunaan *encoder*,, yang terdiri dari komponen berupa *recurrent neural network* (RNN) (Abidin, 2017). Penggunaan RNN dalam model NMT memiliki kekurangan yaitu kurangnya penerapan konsep OOV sehingga jika ada kalimat yang mengandung OOV penelitian akan gagal dalam menerjemahkan. Atas dasar itu sistem tersebut peneliti mengumpulkan korpus teks dengan menerapkan konsep OOV.

Pada tahun 2022, Yustiana Fauziyah, Ridwan Ilyas, Fatan Kasyidi. Melakukan penelitian berjudul "Mesin Penerjemah Bahasa Indonesia-Bahasa Sunda Menggunakan *Recurrent Neural Network*". Tujuan dari studi ini adalah untuk mengembangkan sebuah perangkat penerjemah yang mengalihkan bahasa Indonesia ke bahasa Sunda dengan menerapkan pendekatan NMT. Dalam penelitian ini, digunakan arsitektur *Encoder-Decoder* yang memanfaatkan teknik RNN LSTM untuk menerjemahkan teks dari bahasa Indonesia ke bahasa Sunda (Fauziyah et al., 2022). Penggunaan RNN memiliki kekurangan yaitu mempunyai akurasi yang kurang jika metode *attention* tidak diterapkan pada RNN. Atas dasar itu sistem tersebut berbeda dengan sistem yang dirancang oleh penulis, dimana penulis hanya menggunakan metode Arsitektur untuk melakukan suatu penerjemahan otomatis, yang di dalamnya sudah terdapat *attention.*

Pada tahun 2023, Alda Dwi Meilinda, Herry Sujaini, Novi Safriadi melakukan suatu penelitian yang berjudul "*Pivot Language* Bahasa Melayu Pontianak ke Bahasa Bugis Menggunakan *Neural Machine Translation*". Penelitian ini bertujuan membuat penerjemahan bahasa Melayu Pontianak ke bahasa Bugis dengan memanfaatkan *Neural Machine Translation*. Penelitian ini mengaplikasikan metode Arsitektur *Transformers* serta menambahkan pendekatan *Pivot Language* berfungsi sebagai bahasa penghubung untuk menerjemahkan dari bahasa sumber menuju bahasa sasaran. Dengan menggabungkan beberapa metode yang digunakan hasil dari tingkat akurasi penerjemahan meningkat dari hasil sebelumnya yang tidak menggunakan *Pivot Language* (Meilinda et al., 2023). Penggunaan *pivot language* memiliki kekurangan yaitu metode ini digunakan sebagai penengah dari bahasa

daerah ke bahasa daerah lainnya, penggunaan metode ini diharuskan membuat korpus teks tambahan (bahasa terjemahan) agar hasil evaluasi yang dihasilkan meningkat. Perbedaan dengan sistem yang akan dilakukan, penulis merancang suatu terjemahan menggunakan metode *Transformers* tanpa menggunakan *Pivot Language* sehingga penerjemahan yang dilakukan tidak memerlukan suatu perantara Bahasa.

Berdasarkan Review jurnal diatas, research gap pada penelitian ini

1. Dalam penelitian ini, dilakukan pengujian terhadap jumlah data dan jumlah *epoch* yang bertujuan untuk mengetahui skor *bleu* selama proses *training*
2. Pada saat proses *training,* model yang digunakan adalah *Arsitektur Transformers* yang dikembangkan menggunakan *Framework Marian,*

2.2 Teori Terkait

2.2.1 *Machine Learning*

Pembelajaran mesin, yang menjadi bagian dari kecerdasan buatan, adalah ilmu yang berkaitan dengan pengembangan algoritma dan model statistic yang memungkinkan sistem komputer untuk belajar dari pola dan inferensi tanpa memerlukan instruksi secara eksplisit. Dengan menggunakan algoritma *machine learning*, sistem komputer dapat memproses data historis dalam jumlah besar dan mengidentifikasi pola-pola tertentu (Ott et al., 2018). Hal ini memungkinkan sistem untuk memprediksi hasil dengan lebih akurat berdasarkan serangkaian *input* tertentu. Selain itu, *machine learning* banyak digunakan untuk menyelesaikan berbagai kategori utama, yaitu pembelajaran terarah, pembelajaran tidak terarah, dan pembelajaran penguatan (Roihan et al., 2020).

2.2.2 *Teks Mining*

Text mining adalah ekstraksi menarik dari sebuah teks bebas atau teks tidak terstruktur dan mencakup penggalian informasi, pengklasifikasi teks, pengelompokan teks, penggalian entitas, relasi, dan ekstraksi peristiwa lainnya. Sebelum melakukan *text mining*, langkah awal yang harus dilakukan yaitu mempersiapkan data yang akan dianalisis. Tahap *text preprocessing* merupakan tahap pengolahan data yang sebelumnya tidak berstruktur menjadi data yang berstruktur (Adi, Kristian & Sebastian, 2018).

Pada tahap ini teks disusun menjadi teks terstruktur dan bagian – bagian yang tidak diperlukan dihilangkan dari teks hingga siap untuk diproses lebih lanjut. Berikut tahapan preprocessing teks yang digunakan pada penelitian ini:

Case Folding: proses mengubah semua huruf besar menjadi huruf kecil untuk menyamakan kata yang sejenis tetapi memiliki perbedaan kapitalisasi (Wahyuni et al., 2017).

remove punctuation: menghilangkan tanda baca seperti titik, koma, tanda tanya, dan simbol lainnya yang tidak diperlukan.

Remove whitespace: Menghapus spasi yang berlebihan dalam teks

Remove Stopword: Menghilangkan kata – kata yang tidak bermakna atau tidak relevan, seperti ‘dan’, ‘atau’, ‘adalah’, yang tidak memberikan kontribusi berarti pada analisis teks.

Lemmatization: Menghilangkan kata ke bentuk dasarnya dengan mempertimbangkan struktur morfologis dan aturan tata bahasa. Sebagai contoh, kata “melihat”,

“melihatkan”, dan “terlihat” akan dikembalikan ke bentuk dasar “lihat” (Supriyati & Iqbal, 2018).

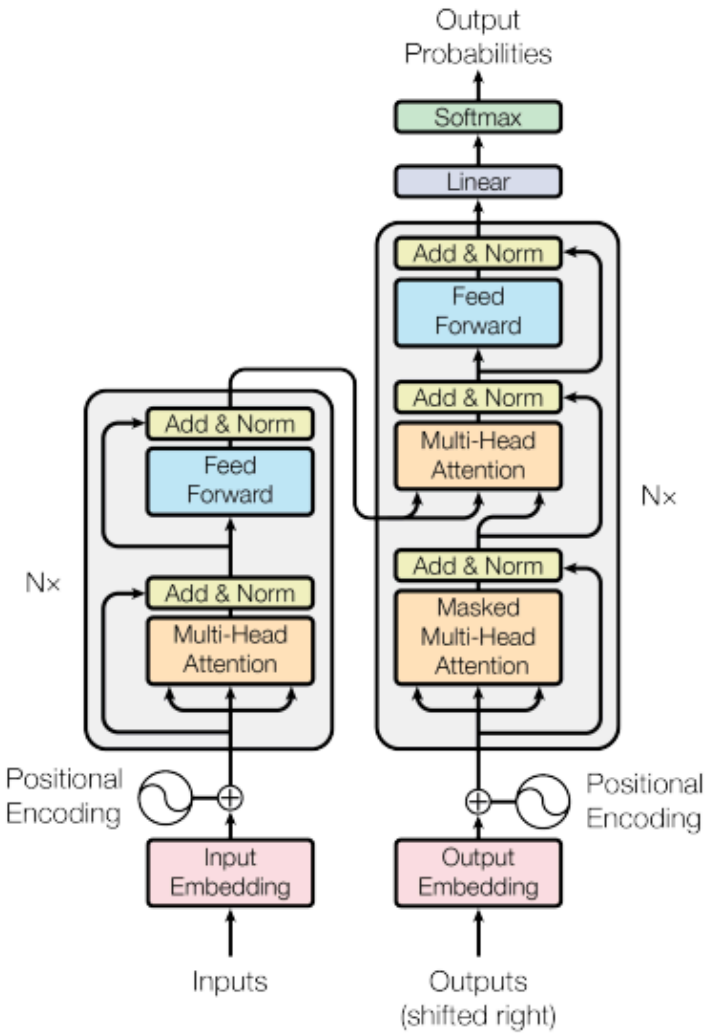
Tokenizing: Proses pemecahan teks menjadi bagian – bagian kecil yang disebut token, berdasarkan spasi atau tanda penghubung. Tokenisasi ini membantu dalam mempermudah pemrosesan teks selanjutnya (Supriyati & Iqbal, 2018).

Pada penelitian ini, tahapan *text preprocessing* yang digunakan meliputi:

- *Case Folding*
- *Remove Punctuation*
- *Remove Whitespace*
- *Tokenizing*

2.2.3 Arsitektur Transformers

Arsitektur Transformers adalah salah satu inovasi yang sangat penting dalam pengembangan pemrosesan bahasa alami dan tugas-tugas pemrosesan berbasis urutan lainnya. Metode ini memungkinkan mesin untuk mengatasi pemahaman bahasa alami, terjemahan mesin, pengenalan entitas berbasis teks, dan berbagai tugas lainnya dengan sangat baik. Transformers juga merupakan model pertama yang hanya mengandalkan perhatian diri untuk memperoleh representasi kalimat masukan dan keluaran selama proses pelatihan (Vaswani et al., 2017).



Gambar 2. 1 Arsitektur Transformers

2.2.3.1 Encoder dan Decoder

Transformers terdiri dari dua komponen utama, yaitu *encoder* dan *decoder*.

Encoder memiliki dua *sub*-layer, pertama adalah *multi-head attention*, kedua adalah *fully-connected feed-forward network* (Yin, 2020). *Multi-head attention layer* membantu *encoder* memfokuskan perhatian pada kata–kata tertentu dan memahami konteks keseluruhan dari masukan. *Feed-forward network* kemudian menghasilkan vector yang diteruskan ke *decoder*.

Arsitektur **Decoder** mirip dengan *encoder,* namun ditambahkan *sub-layer* ketiga yang disebut *masked multi-head attention*. Lapisan ini berfungsi untuk menyembunyikan beberapa vektor yang dihasilkan oleh *encoder*, sehingga tidak hanya mencegah kebocoran informasi tetapi juga memeriksa output yang dihasilkan. *Input* awal dari *decoder* yaitu *token start*, yang menghindari kekosongan dalam pergeseran *output*. Selain itu, *token end* digunakan untuk menandai akhir dari kalimat, dan *decoder* tidak akan memproses *token end* tersebut

2.2.3.2 Self-Attention

Self-attention merupakan mekanisme penting dalam arsitektur *transformers* yang memungkinkan model untuk memperhatikan bagian tertentu dari input teks selama proses penerjemahan. Mekanisme ini membantu model fokus pada kata-kata tertentu yang relevan, berdasarkan konteksnya.

Proses *self-attention* melibatkan tiga komponen utama yaitu *query*, *key*, dan *value* dimana setiap kata dalam kalimat direpresentasikan dalam bentuk vektor. Mekanisme ini memungkinan model menghitung bobot atau perhatian pada setiap kata dalam kalimat dan menentukan seberapa relevan suatu kata dalam hubungannya dengan kata-kata lainnya di seluruh kalimat.

2.2.3.2.1 Scarled Dot-Product Attention

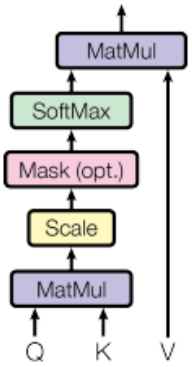
Fungsi utama dari *scarled dot-product attention* adalah memetakan *query* dan *key* ke dalam *output* yang dihitung dari nilai *weighted sum* atau penjumlahan berbobot dari *values*. Proses dimulai dengan mengalikan *query* dengan *key*, kemudian hasil perkaliannya dibagi dengan akar dari dimensi *query* $\sqrt{d_k}$, sebelum melalui proses *sofmax* untuk mendapatkan bobot

atensinya. Setelah itu, bobot-bobot ini dikalikan dengan *value* untuk menghasilkan *output* akhir.

Notasi yang digunakan untuk fungsi ini adalah:

$$\text{Attention}(Q,K,V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Proses ini memastikan bahwa setiap kata dalam *input* dapat menyesuaikan atensinya terhadap kata-kata lain dengan bobot yang sesuai, berdasarkan relevansinya dalam kalimat.

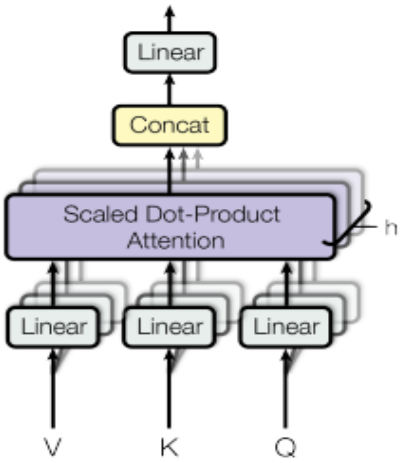


Gambar 2. 2 Scarlet dot-product attention

2.2.3.2.2 Multi-Head Attention

Multi-head attention merupakan pengembangan dari mekanisme *self-attention* yang memungkinkan model untuk memberikan perhatian pada berbagai bagian input secara simultan. Alih-alih hanya menghasilkan satu set *query*, *key*, dan *value*, *multi-head attention* membuat beberapa set secara parallel. Setiap set menghasilkan *attention* yang berbeda, memungkinkan model untuk memproses berbagai aspek dari kalimat *input* secara bersamaan.

Setelah itu, hasil dari setiap *head* digabungkan dan diteruskan ke lapisan *feed-forward* (Vaswani et al., 2017). Vaswani menggunakan 8 *head* dalam eksperimennya, sehingga menghasilkan delapan *matrik attention* yang berbeda, yang kemudian digabungkan sebelum diteruskan ke *feed-forward network*



Gambar 2. 3 Multi-head Attention

2.2.3.3 Position-wise Feed-Forward Network

Pada arsitektur *transformers*, setiap lapisan *encoder* dan *decoder* memiliki jaringan *feed-forward* yang beroperasi dengan kondisi independent pada setiap posisi dalam urutan *input*. *Feed-forward network* ini terdiri dari dua lapisan transformasi linier yang dipisahkan oleh fungsi aktivasi *rectified linear unit* (ReLU). *Input* dari lapisan *self-attention* diproses oleh *feed-forward network* untuk mengekstrak informasi lebih lanjut (Vaswani et al., 2017).

Notasi umum *feed-forward network* yaitu:

$$FFN(x) = \left(0,\ xW_1 + b_1\right) W_2 + b_2$$

Di sini, x adalah *input vektor*, sementara $W1$ dan $W2$ adalah bobot dari dua lapisan linier, dan $b1$ serta $b2$ adalah bias masing-masing lapisan. *Output* dari *feed-forward network* ini kemudian diteruskan ke lapisan berikutnya dalam bentuk model.

2.2.3.4 Embedding dan Softmax

Embedding digunakan sebagai konversi kata-kata dari teks *input* menjadi *vektor* berdimensi tetap, yang dapat diproses oleh model. Setiap kata dalam teks diubah menjadi *vektor* dengan dimensi d_{model}, yang menunjukkan representasi numerik dari kata tersebut. Setelah itu, output dari *decode* pada *Transformers* diubah kembali menjadi probalitasi melalui lapisan *softmax*. Fungsi *softmax* mengonversi *output* menjadi distribusi probabilitas untuk memprediksi token berikutnya dalam urutan terjemahan (Vaswani et al., 2017).

Selama proses pelatihan, *embedding* digunakan untuk menangani *input* dan *output*, dengan bobot *embedding input* dan *output* sering kali dibagi untuk menghemat parameter. *Transformers* linier dan fungsi *softmax* bersama-sama membantu model memprediksi token yang paling mungkin berdasarkan *input* yang telah diproses oleh lapisan sebelumnya.

2.2.3.5 Positional Encoding

Model ini tidak mengandalkan RNN (*Recurrent Neural Network*) atau CNN (*Convolution Neural Network*). Untuk menangani urutan kata dalam kalimat, vektor *positional encoding* memiliki dimensi yang sebanding dengan vektor dari *embedding layer*. Selanjutnya, vektor ini digabungkan dengan vektor yang dihasilkan dari *embedding layer* (Vaswani et al., 2017).

Karena arsitektur *Transformers* tidak memiliki elemen berurutan seperti pada RNN, *positional encoding* ditambahkan untuk memberikan model informasi tentang urutan kata dalam kalimat. *Positional encoding* menggunakan fungsi sinus dan cosinus dengan frekuensi berbeda untuk menghasilkan vektor posisi yang kemudian ditambahkan ke *embedding*, memungkinkan model untuk mempertahankan informasi tentang posisi relatif setiap kata dalam kalimat (Vaswani et al., 2017).

Model ini menggunakan fungsi sin dan cos dari frekuensi yang berbeda

$$PE_{(pos,2i)} = \sin sin\ (pos/10000^{2i/d_{model}})$$

$$PE_{(pos,2i+1)} = \sin sin\ (pos/10000^{2i/d_{model}})$$

dimana pos adalah posisi kata dalam urutan, dan i menunjukkan dimensi vektor. Dengan adanya *positional encoding*, *transformers* dapat mempertimbangkan urutan kata, meskipun tidak menggunakan mekanisme berulang seperti pada RNN.

2.2.4 MarianMT

MarianMT adalah sebuah framework yang dikembangkan untuk melatih model Neural Machine Translation yang efisien dan skalabel (Aulamo & Boggia, 2023). MarianMT dikembangkan oleh tim Microsoft dan menggunakan arsitektur berbasis Transformers, yang telah terbukti sangat efektif dalam berbagai tugas terjemahan mesin (Abidin, 2017). Arsitektur MarianMT memanfaatkan metode self-attention dan multi-head attention yang memungkinkan model untuk fokus pada bagian-bagian tertentu dari input ketika menghasilkan output (Aulamo & Boggia, 2023). MarianMT mendukung pemrosesan batch dan dapat berjalan dengan baik

pada GPU, menjadikannya pilihan yang kuat untuk tugas-tugas penerjemahan yang membutuhkan kinerja tinggi (Purnama & Utami, 2023).

Keunggulan utama MarianMT terletak pada kemampuannya untuk menangani berbagai bahasa dengan sumber daya terbatas dan kemampuannya untuk mengakomodasi model-model besar tanpa memerlukan sumber daya komputasi yang berlebihan (Aulamo & Boggia, 2023). Framework ini juga mendukung berbagai teknik optimasi seperti back-translation dan fine-tuning, yang membantu meningkatkan kualitas terjemahan terutama pada pasangan bahasa dengan data terbatas (Abidin, 2017).

2.2.5 BLEU

BLEU (*bilingual evaluation understudy*) merupakan salah satu metrik evaluasi yang paling sering digunakan untuk mengukur kualitas hasil terjemahan mesin. Algoritma ini menghitung seberapa mirip hasil terjemahan otomatis dengan terjemahan referensi. Skor BLEU mengukur akurasi statistic berbasis *n-gram*, dengan mempertimbangkan panjang kalimat hasil terjemahan dan kalimat referensi. BLEU juga menggunakan *penalti brevity* (BP) untuk mengurangi skor jika terjemahan terlalu pendek dibandingkan dengan referensi, sehingga menjaga keseimbangan antara akurasi dan panjang terjemahan (Abidin, 2017).

Proses perhitungan BLEU dimulai dengan menghitung *precision* dari *n-gram* yang cocok antara hasil terjemahan dan referensi. Formula umum untuk menghitung BLEU adalah sebagai berikut:

$$BP_{BLEU} = \left\{1,\ e\left(1 - \frac{r}{c}\right),\ if\ if\ c > r\ c \leq r\right\}$$

$$p_n = \frac{C \in \sum \{Candidates\} \sum n-gram \in C \ Count_{dip}(n-gram)}{C' \in \sum \{Candidates\} \sum n-gram' \in C' \ Count\,(n-gram')}$$

$$BLEU = BP_{BLEU} \cdot exp\left(\sum_{n=1}^{N} w_n \log log\, p_n \right)$$

Keterangan:

c = jumlah kata dalam hasil terjemahan otomatis.

BP = brevity penalty (penalti singkat).

Pn = jumlah *n-gram* dalam hasil terjemahan yang cocok dengan referensi dibagi dengan jumlah *n-gram* dalam hasil terjemahan.

Pn = *modified precision score* (skor presisi yang di modifikasi).

Wn = 1/N (standar nilai N untuk BLEU yaitu 4).

Semakin tinggi skor BLEU, semakin mendekati hasil terjemahan mesin dengan terjemahan referensi, yang berarti terjemahan tersebut dianggap berkualitas lebih baik. Skor ini berkisar dari 0 hingga 1, dimana 1 adalah hasil terjemahan yang identik dengan referensi (Fauziyah et al., 2022). Skor BLEU pada umumnya dianggap baik jika berada di atas 30, sementara skor di atas 50 menunjukkan bahwa hasil terjemahan sudah sangat mendekati kualitas manusia (Lavie, 2010). Namun, perlu diingat bahwa meskipun BLEU memberikan indikasi kualitas, skor ini tidak selalu mencerminkan aspek linguistik yang lebih kompleks, seperti keakuratan konteks dan kealamian bahasa.

BLEU dirancang untuk mengukur akurasi dalam berbagai tugas penerjemahan, dan penggunaanya yang luas menjadikannya standar dalam menilai performa model terjemahan otomatis (Vaswani et al., 2017). Penggunaan beberapa terjemahan referensi akan meningkatkan skor BLEU karena lebih banyak variasi

terjemahan yang dianggap benar, sehingga memberikan penilaian yang lebih akurat terhadap model (Shaw et al., 2018).

2.2.6 Django

Django, *framework web* tingkat tinggi berbasis *Python*, dirancang untuk mempercepat pengembangan aplikasi web yang aman dan skalabel. Django memungkinkan pengembangan untuk membangun aplikasi web dengan cepat tanpa mengorbankan kinerja dan keamanan, dengan mengadopsi pola arsitektur Model-template-view (MTV) yang mirip dengan model-view-controller (MVC) pada framework lainnya.

Sistem manajemen url yang fleksibel dan fitur Django Admin untuk antarmuka administrative membuat Django sangat efisien dan *user-friendly*. Django unggul dalam kinerja jumlah permintaan dan efisiensi penggunaan dan sumber daya, menjadikannya pilihan yang ideal untuk aplikasi web yang memerlukan skalabilitas tinggi (Amarulloh et al., 2023). Selain itu Django memiliki waktu respon cepat dan penggunaan CPU serta memori yang rendah, membuatnya sangat cocok untuk pengembangan aplikasi web berbasis mobile dan server (Saputra, 2018).

2.2.7 Struktur Bahasa

Struktur bahasa Indonesia mencakup berbagai unsur yang membentuk kalimat. Berikut adalah penjelasan lebih rinci tentang struktur bahasa Indonesia:

1. **Kalimat Dasar:**
    - Setiap kalimat dasar terdiri dari subjek, predikat, dan objek.
    - *Contoh:* "Ani (subjek) makan (predikat) nasi (objek)."
2. **Subjek (Pelaku):**

- Subjek merupakan orang, hewan, atau benda yang melakukan tindakan dalam kalimat.
- *Contoh:* "Mereka (subjek) bermain (predikat) di taman."

3. **Predikat (Pekerjaan atau Tindakan):**
    - Predikat adalah kata kerja yang menyatakan tindakan atau keadaan.
    - *Contoh:* "Dia (subjek) tidur (predikat)."
4. **Objek (Sasaran atau Penerima Tindakan):**
    - Objek adalah orang, hewan, atau benda yang menjadi sasaran atau penerima tindakan.
    - *Contoh:* "Saya (subjek) menyapu (predikat) lantai (objek)."
5. **Pelengkap:**
    - Pelengkap memberikan informasi tambahan tentang subjek atau objek.
    - *Contoh:* "Bunga (subjek) itu wangi (pelengkap)."
6. **Keterangan:**
    - Keterangan memberikan informasi tambahan tentang waktu, tempat, cara, alasan, atau kondisi.
    - *Contoh:* "Mereka (subjek) bermain (predikat) di pantai (keterangan)."
7. **Penghubung:**
    - Penghubung digunakan untuk menghubungkan kata, frasa, atau klausa dalam kalimat.
    - *Contoh:* "Saya suka makan es krim, tetapi adik saya lebih suka cokelat."
8. **Klausa:**
    - Klausa adalah bagian dari kalimat yang memiliki subjek dan predikat sendiri.
    - *Contoh:* "Ketika hujan turun, kami tinggal di dalam rumah."
9. **Kalimat Majemuk:**
    - Kalimat majemuk terdiri dari dua atau lebih klausa yang saling terkait.
    - *Contoh:* "Saya suka membaca buku, dan teman saya suka menulis cerita."

10. **Tanda Baca:**

    - Tanda baca digunakan untuk memisahkan, menghubungkan, atau memberikan penekanan dalam kalimat.
    - *Contoh:* "Maaf, saya tidak bisa datang besok!"

Penting untuk memahami dan mengaplikasikan struktur bahasa Indonesia agar komunikasi lebih jelas dan efektif (Sulistiyo, 2016). Dengan memahami struktur ini, Anda dapat merangkai kalimat dengan jelas sehingga mudah dipahami oleh orang lain.

Berikut adalah contoh struktur bahasa indonesia dan bahasa kokas

Saya akan pergi ke kantor sekarang

Saya (subjek) akan pergi (predikat) ke kantor (objek) sekarang (keterangan)

Yai e mau eti kantor madige

Yai (subjek) e mau (predikat) kantor (objek) madige (keterangan)

BAB III
ANALISIS DAN PERANCANGAN

3.1Analisis

3.1.1 Identifikasi Masalah

Jumlah bahasa daerah di Indonesia adalah sebanyak 718 bahasa dan sebanyak 90 persen terbesar di wilayah Indonesia timur dengan 428 jenis bahasa terdapat di Papua, salah satunya adalah daerah Kokas, yang kini menghadapi ancaman kepunahan. Hal ini disebabkan oleh kurangnya dokumentasi, pemanfaatan, dan pelestarian bahasa tersebut di tengah arus modernisasi dan dominasi bahasa nasional serta global (Sumatera & Rahima, 2024).

Pelestarian bahasa daerah menjadi semakin penting dalam era globalisasi untuk memastikan keberlanjutan budaya dan identitas lokal (Apriani et al., 2016). Namun, di daerah seperti Kokas, tidak banyak orang yang masih fasih atau bahkan memahami bahasa daerah tersebut kurangnya akses terhadap informasi dan materi pembelajaran dalam bahasa Kokas mempercepat proses kepunahan.

Dalam konteks diatas, penelitian ini mengidentifikasi kebutuhan untuk mengembangkan alat penerjemah otomatis yang dapat menerjemahkan bahasa Indonesia ke bahasa Papua Kokas. Alat ini diharapkan dapat memudahkan komunikasi dan akses informasi bagi masyarakat yang berbicara dalam bahasa Kokas, sekaligus menjadi upaya untuk melestarikan bahasa yang semakin terpinggirkan ini.

3.1.2 Pemecahan Masalah

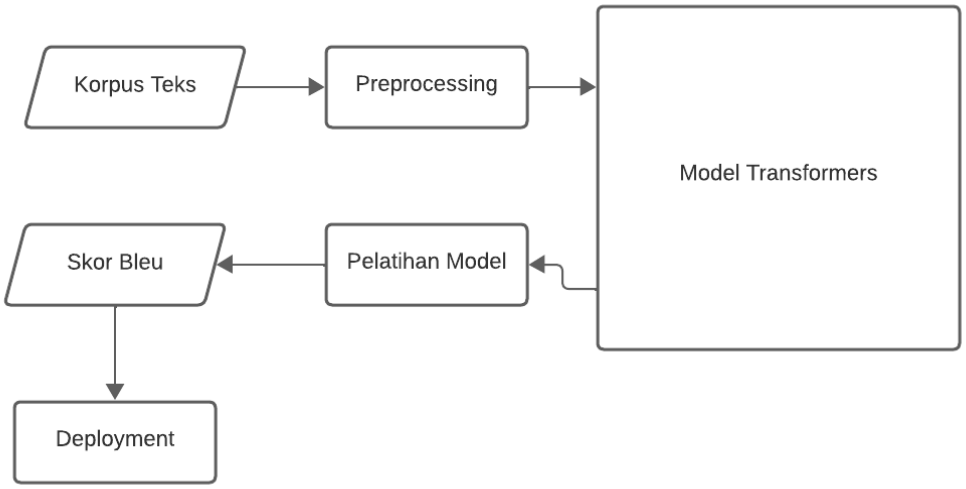
Berkaitan dengan permasalahan yang ditemukan, peneliti kemudian mengusulkan pengembangan sistem penerjemahan otomatis berbasis *Transformers*.

Metode *Transformers* dipilih karena telah terbukti efektif dalam tugas penerjemahan mesin dan dapat menangani konteks kalimat secara keseluruhan tanpa memerlukan urutan waktu seperti pada model RNN. Sistem ini dirancang untuk menerjemahkan teks dari bahasa Indonesia ke bahasa Papua Kokas dengan akurasi yang tinggi. Teknologi penerjemahan mesin dapat menjadi alat yang efektif untuk mendukung pelestarian bahasa daerah melalui penerjemahan dan pembelajaran yang lebih mudah diakses (Fauziyah et al., 2022).

3.2 Perancangan

3.2.1 Perancangan Sistem Transformers

Pada penelitian metode Transformers untuk melakukan terjemahan bahasa terdapat beberapa tahapan yaitu:



Gambar 3. 1 Perancangan Sistem Transformers

3.2.1.1 Korpus Teks

Korpus teks merujuk pada kumpulan data teks yang digunakan sebagai bahan dasar dari proses pelatihan model penerjemahan. Dalam penelitian ini, korpus teks yang digunakan terdiri dari pasangan kalimat dalam bahasa Indonesia dan bahasa Papua Kokas. Korpus ini diperoleh melalui wawancara dengan masyarakat asli di daerah Kokas, Papua. Data yang telah terkumpul kemudian disusun dalam format CSV dengan dua kolom: bahasa Indonesia dan bahasa Papua Kokas

Proses pengumpulan data ini sangat penting karena korpus teks menjadi fondasi dari model penerjemah yang akan dibangun. Kualitas dan representasi dari data yang digunakan sangat mempengaruhi hasil akhir dari model penerjemah. Kuantitas dan kualitas korpus berperan penting dalam meningkatkan akurasi model penerjemahan statistik (Apriani et al., 2016).

Tabel 3. 1 Data korpus

Korpus Teks bahasa Indonesia dan bahasa Papua Kokas	Jumlah data
data_2908.csv	2908 data parallel

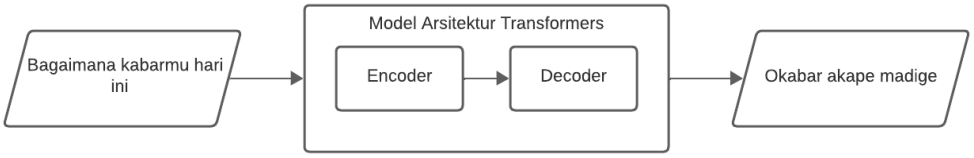
3.2.1.2 Preprocessing

Tahapan *preprocessing* pada penelitian ini dilaksanakan untuk memastikan bahwa data yang digunakan dalam pelatihan model bersih dan siap untuk diproses. *Preprocessing* mencangkup langkah-langkah berikut :

1. Memuat Dataset : pada bagian ini dataset yang dikumpulkan lalu di proses untuk membagi dua data yaitu data pelatihan dan data validasi. Pembagian dataset ini dilakukan untuk memastikan bahwa model dapat dilatih dengan jumlah data yang cukup dan diuji dengan data yang tidak pernah dilihat sebelumnya untuk mengukur kinerjanya secara objektif.
2. *Tokenizing* : Proses ini dilakukan untuk memecah teks dalam bahasa Indonesia dan Papua Kokas menjadi token-token yang dapat diproses oleh model. Tokenisasi dilakukan dengan menggunakan *MarianTokenizer* dari model *Helsinki-NLP/opus-mt-id-en*. Tokenisasi ini secara otomatis mencangkup proses seperti *case folding* dan *removing punctuation*. Tokenisasi merupakan tahap awal yang penting dalam text mining karena membantu mengubah teks yang tidak terstruktur menjadi lebih mudah di proses (Wahyuni et al., 2017).
3. *Encoding* : Setelah tokenisasi, teks yang telah diubah menjadi token kemudian di-encode menjadi tensor menggunakan *Pytorch*. Tensor ini diperlukan agar data dapat diproses oleh model selama pelatihan. *Encoding* juga mencangkup *Padding*, yaitu penambahan token khusus agar semua input memiliki panjang yang sama, dan *Truncation*, yaitu pemotongan teks yang terlalu panjang. Pentingnya proses dalam mengubah teks menjadi format numerik yang dapat dipahami oleh model pembelajaran mesin (Sulistiyo, 2016).

3.2.1.3 Model Transformers

Model *Transformers,* Metode ini memungkinkan mesin untuk mengatasi pemahaman bahasa alami, terjemahan mesin, pengenalan entitas berbasis teks, dan berbagai tugas lainnya dengan sangat baik. Dalam penelitian ini, model menyertakan lapisan *encoder-decoder*. Lapisan encoder mengekstrak fitur dari kalimat *input* dan lapisan *decoder* mengembalikan nilai yang diekstraksi ke kalimat yang diterjemahkan.



Gambar 3. 2 (contoh model transformers)

Model *Transformers* yang digunakan dalam penelitian ini adalah Model *MarianMT* dari *Helsinki-NLP/opus-mt-id-en*.Model ini sudah terlatih sebelumnya (*pre-trained*) dan digunakan sebagai dasar untuk melatih model penerjemah dari bahasa Indonesia ke bahasa Papua Kokas. Arsitektur *Transformers* dipilih karena kemampuannya yang unggul dalam menangani masalah penerjemahan mesin tanpa memerlukan struktur urutan seperti RNN. Model ini terbukti efektif dalam tugas penerjemahan bahasa karena menggunakan *self-attention* yang mampu menangkap konteks kalimat secara keseluruhan (Vaswani et al., 2017).

Pada tahap ini, model dimuat bersama dengan *tokenizer* yang sesuai. Model ini kemudian dilatih menggunakan dataset yang telah diproses sebelumnya. Selama pelatihan, model diberi pasangan kalimat dalam bahasa Indonesia dan Papua Kokas,

dan model diharapkan dapat mempelajari pola terjemahan yang tepat di antara kedua bahasa ini.

3.2.1.4 Pelatihan Model

Proses pelatihan model dilakukan dengan menggunakan dataset yang telah diproses. Model dilatih menggunakan *Trainer* dari pustaka *Transformers* dengan konfigurasi yang sudah ditentukan, seperti jumlah *epoch*, ukuran *batch,* dan strategi evaluasi. Selama pelatihan, model terus dievaluasi menggunakan data validasi untuk memantau kinerja dan memastikan bahwa model tidak mengalami *overfitting*. Dalam penelitian (Roihan et al., 2020) menggunakan pendekatan serupa untuk melatih model pembelajaran mesin, dimana proses evaluasi berkala sangat penting untuk mendapatkan model yang optimal.

Pelatihan dilakukan dalam beberapa epoch untuk memastikan model dapat mengkonvergensi ke solusi yang optimal. Selama pelatihan, model terus diperbarui untuk meningkatkan akurasi terjemahan, dengan log pelatihan yang disimpan untuk analisis lebih lanjut.

3.2.1.5 Skor Bleu

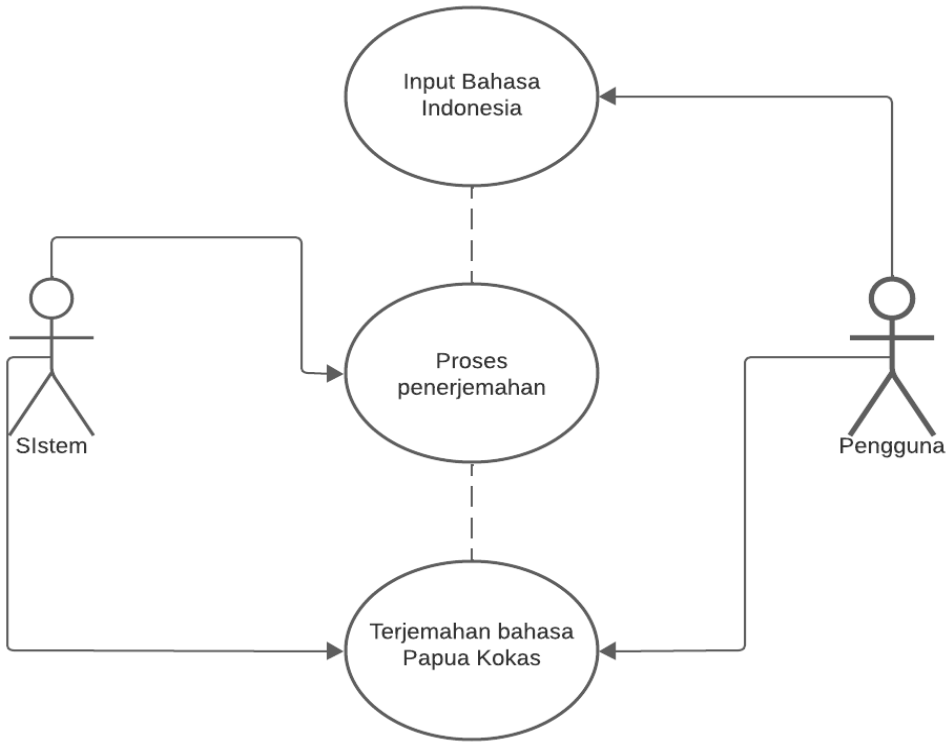
Metrik *BLEU (Bilingual Evaluation Understudy)* digunakan menilai kualitas hasil terjemahan yang dihasilkan pada model. Skor *BLEU* dihitung dengan membandingkan hasil terjemahan model dengan referensi terjemahan yang benar. Dalam penelitian ini, *BLEU* digunakan untuk mengevaluasi sejauh mana model menerjemahkan teks dari bahasa Indonesia ke bahasa Papua Kokas. *BLEU* adalah metrik yang umum digunakan dalam evaluasi penerjemahan mesin, karena mampu

memberikan gambaran tentang akurasi terjemahan berdasarkan n-gram yang cocok dengan referensi. Nilai *BLEU* yang tinggi menunjukkan bahwa hasil terjemahan sangat mendekati terjemahan yang diharapkan, yang sangat penting dalam konteks penerjemahan otomatis (Abidin, 2017).

3.2.1.6 Deployment

Pada tahap *deployment*, model yang telah dilatih diimplementasikan dalam sebuah sistem yang dapat diakses oleh pengguna. Dalam penelitian ini deployment dilakukan dengan mengintegrasikan model penerjemahan otomatis kedalam produk aplikasi website. Aplikasi ini dirancang untuk memungkinkan teks dalam bahasa Indonesia dan mendapatkan terjemahan dalam bahasa Papua Kokas secara real-time.

3.2.2 Perancangan *Use Case*



Gambar 3. 3 Use Case

Input bahasa Indonesia

Pengguna cukup memasukkan teks bahasa Indonesia

1. Proses penerjemahan

   Pada proses ini sistem akan langsung memproses bahasa Indonesia yang sudah di input oleh pengguna

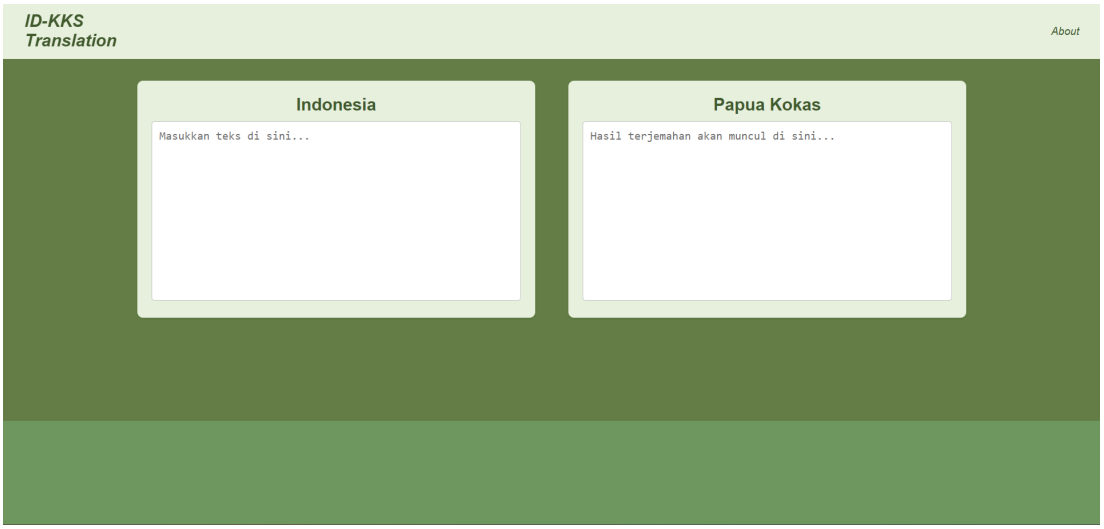
2. Terjemahan bahasa Papua Kokas

   Pada tahap ini output yang di keluarkan sistem akan terbentuk dalam terjemahan yang di input oleh pengguna pada sebelumnya

3.2.3 Perancangan *User Interface* / *Mock-up* aplikasi

Dibawah ini merupakan rancangan website yang digunakan sebagai pengujian hasil penelitian:

- Halaman Home



Gambar 3. 4 Halaman Home

- Halaman About

Gambar 3. 5 Halaman About

3.3 Rancangan Pengujian

Pada rancangan pengujian menjelaskan tentang bagaimana rencana pengujian yang akan dilakukan. Metode yang digunakan contohnya *white box, black box, grey box* dan lain-lain.

Dalam Penelitian ini menggunakan dua pengujian yaitu:

3.3.1 Pengujian *Black box*

Pengujian black box pada model penerjemah bertujuan untuk memastikan bahwa model dapat secara konsisten menghasilkan terjemahan yang akurat dan berkualitas tinggi dari bahasa Indonesia ke bahasa Papua Kokas tanpa memerlukan pemahaman mendalam tentang algoritma atau struktur internal model. Pengujian ini penting untuk memastikan bahwa model siap digunakan dalam aplikasi dunia nyata, seperti di website yang kembangkan.

3.3.2 Pengujian *Matrik BLEU*

Pengujian BLEU yang dilakukan secara konsisten di berbagai tahap pelatihan dan dengan dataset uji yang berbeda akan memberikan pemahaman yang lebih mendalam mengenai kemampuan model dalam penerjemahan otomatis. proses pengujian dilakukan beberapa langkah :

1. Pengumpulan data uji yang digunakan dalam pengujian *BLEU* terdiri dari kalimat parallel bahasa Indonesia dan bahasa Papua Kokas. Data ini terbagi menjadi dua bagian utama, yaitu data pelatihan dan data validasi, yang masing-masing berfungsi untuk melatih dan menguji model (Shaw et al., 2018).
2. Pengukuran Skor *BLEU*, setelah model dilatih menggunakan data pelatihan, model diuji dengan data validasi. Skor *BLEU* dihitung untuk menilai seberapa baik model dapat menghasilkan terjemahan yang mendekati terjemahan referensi. Metrik ini menggunakan pendekatan n-gram untuk mengevaluasi

keakuratan terjemahan, dimana semakin tinggi skor *BLEU*, semakin baik kualitas terjemahan yang dihasilkan (Shaw et al., 2018).

3. Evaluasi berkala, Evaluasi dilakukan pada setiap epoch pelatihan untuk memantau perkembangan skor *BLEU* dan memastikan model tidak mengalami overfitting. Evaluasi ini penting untuk menentukan jumlah *epoch* yang optimal dalam pelatihan model, mengacu pada metode yang telah diusulkan (Abidin, 2017).
4. Analisis hasil, hasil dari pengukuran skor *BLEU* dianalisis untuk melihat tren peningkatan atau penurunan performa model. Peningkatan skor *BLEU* menunjukkan bahwa model semakin akurat dalam menghasilkan terjemahan. Stagnasi atau penurunan skor dapat mengindikasikan perlunya penyesuaian dalam parameter pelatihan atau data yang digunakan (van der Wees et al., 2017).

BAB IV
PEMBAHASAN

4.1 Gambaran Umum Proyek Penelitian

Penelitian ini berfokus pada pengembangan dan implementasi model penerjemah otomatis dari bahasa Indonesia ke bahasa Papua Kokas, sebuah bahasa yang digunakan oleh komunitas di daerah Papua Kokas. Metode yang digunakan pada penelitian ini yaitu Arsitektur Transformers melalui penerapan framework MarianMT.

Sistem ini terdiri dari dua komponen utama. Pertama, fokus penelitian utama adalah pelatihan model Arsitektur Transformers, di mana model dilatih menggunakan korpus teks dalam bahasa Indonesia dan bahasa Papua Kokas. Kedua, implementasi model yang telah dilatih ke dalam sebuah website sederhana, yang memungkinkan pengguna untuk secara langsung menerjemahkan teks dari bahasa Indonesia ke bahasa Papua Kokas.

Proses pelatihan model terdiri dari beberapa tahapan. Tahapan pertama adalah memproses dataset dengan membaginya menjadi dua bagian: data pelatihan dan data validasi. Tahapan kedua adalah tokenisasi dataset yang telah dibagi menggunakan teknik Subword (sub-kata) yang ada pada Arsitektur Transformers, menggunakan MarianTokenizer. Model Arsitektur Transformers kemudian diimplementasikan dan dilatih menggunakan dataset yang sudah diproses melalui tahapan *encoding, padding,* dan *truncation*. Setelah proses pelatihan, model diuji menggunakan data validasi untuk mengevaluasi performa model dengan metrik seperti skor BLEU.

Setelah model dilatih dan dievaluasi, langkah berikutnya adalah mengimplementasikan model tersebut ke dalam sebuah website. Website ini dirancang untuk memberikan kemampuan penerjemahan otomatis dari bahasa Indonesia ke bahasa Papua Kokas secara real-time. Proses inferensi dilakukan di backend website, di mana teks input dalam bahasa Indonesia dikirim ke server, diterjemahkan oleh model yang telah dilatih, dan hasil terjemahan ditampilkan kembali kepada pengguna. Website ini mengintegrasikan teknik optimisasi seperti *debounce* untuk memastikan bahwa proses terjemahan berjalan efisien dan responsif.

4.2 Implementasi

Bagian ini menjelaskan tentang penerapan rancangan yang dideskripsikan pada bab sebelumnya. Implementasi ini terfokus pada implementasi program untuk sistem yang telah dirancang.

4.2.1 Spesifikasi Produk

Spesifikasi produk yang dihasilkan dalam penelitian ini adalah sebuah aplikasi atau sistem yang menggunakan Arsitektur Transformers untuk melakukan implementasi penerjemahan otomatis bahasa Indonesia dan bahasa Papua Kokas. Aplikasi ini menerima input bahasa Indonesia dan mengeluarkan output terjemahan bahasa Papua Kokas

A. Perangkat keras (Hardware) Perangkat keras yang digunakan pada penelitian ini adalah sebuah komputer dengan spesifikasi sebagai berikut:

1. Processor : AMD Ryzen 5 4500U @ 2.4GHz

2. Graphics Card : Radeon Graphics

3. Memory : 20480MB RAM

4. Storage : SSD 512 dan SSD 512

5. Operating System : Windows 11 Education Insader Preview 64-bit

b. Perangkat lunak (Software) Perangkat lunak yang digunakan pada penelitian ini yaitu:

1. Teks Editor : Microsoft Visual Code Studio

2. Aplikasi Lain – lain : Figma, Draw.io, Library Transformers

4.2.2 Implementasi Program

Bagian ini menjelaskan tahap-tahap implementasi program yang digunakan penerjemahan bahasa Indonesia ke bahasa Papua Kokas menggunakan Arsitektur Transformers. Implementasi program ini meliputi proses persiapan dataset dan tokenisasi, pemuatan model dan pengaturan pelatihan, pelatihan model dan evaluasi model, penyimpanan model dan inferensi model, dan website.

4.2.2.1 Persiapan dataset dan tokenisasi

Langkah pertama, persiapan dataset dimuat dari file CSV dan dibagi menjadi data pelatihan dan validasi, diikuti dengan proses tokenisasi menggunakan MarianTokenizer. Proses ini memastikan model memiliki data yang cukup untuk dilatih dan di uji secara objektif, dimana data teks bahasa Indonesia dan Papua Kokas diubah menjadi token ID. Berikut segmen program untuk memuat dataset dan tokenisasi.

```
# 1. Memuat Dataset
file_path = 'data_2908.csv'
df = pd.read_csv(file_path)
# 2. Membagi Dataset menjadi Training dan Validation
train_df, val_df = train_test_split(df, test_size=0.2)
print(f"Training data size: {len(train_df)}")
print(f"Validation data size: {len(val_df)}")
# 3. Memuat Model dan Tokenizer
model_name = "Helsinki-NLP/opus-mt-id-en"
tokenizer = MarianTokenizer.from_pretrained(model_name)
# 4. Memproses Data untuk Pelatihan
train_encodings = tokenizer(
    list(train_df['indonesian']),
    text_target=list(train_df['papua_kokas']),
    return_tensors="pt",
    padding=True,
    truncation=True
)
val_encodings = tokenizer(
    list(val_df['indonesian']),
    text_target=list(val_df['papua_kokas']),
    return_tensors="pt",
    padding=True,
    truncation=True
)
# Menyimpan label (karena ini bagian dari output yang perlu
digunakan untuk pembuatan dataset)
train_labels = train_encodings["labels"]
val_labels = val_encodings["labels"]
# 5. Membuat Dataset untuk PyTorch
class TranslationDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels
    def __getitem__(self, idx):
        item = {key: val[idx].clone().detach() for key, val in
self.encodings.items()}
        item['labels'] = self.labels[idx].clone().detach()
        return item
    def __len__(self):
        return len(self.labels)
train_dataset = TranslationDataset(train_encodings, train_labels)
val_dataset = TranslationDataset(val_encodings, val_labels)
```

Segmen Program 4. 1 pemuatan dataset dan tokenisasi

Pada segmen program 4.1 pemuatan dataset dan tokenisasi akan dijelaskan beberapa fungsi dari kode diatas pada tabel 4.1

Tabel 4. 1 Penjelasan pemuatan dataset dan tokenisasi

Kelas	Objek	Keterangan
‘pd.read_csv’	‘file_path’	Memuat dataset dari file CSV yang digunakan sebagai data pelatihan dan validasi
‘train_test_split’	‘train_df’, ‘val_df’	Membagi dataset menjadi dua ‘train_df’ untuk pelatihan dan ‘val_df’ untuk validasi dengan rasio 90:10
‘MarianTokenizer’	‘model_name’	Memuat tokenizer pre-trained dari model ‘Helsinki-NLP/opus-mt-id-en’.
‘TranslationDataset’	‘train_encodings’, ‘train_labels’	Kelas custom untuk menyimpan dan mengelola data pelatihan ‘train_encodings’ dan label ‘train_labels’ dalam format PyTorch.
	‘val_encodings’, ‘val_labels’	Kelas custom untuk menyimpan dan mengelola data pelatihan ‘val_encodings’ dan label ‘val_labels’ dalam format PyTorch.

Pada saat memuat dataset lalu dataset akan dibagi menjadi dua bagian 2617 untuk data yang akan dipakai melatih model dan 291 untuk data yang dipakai untuk memvalidasi performa model.

Setelah melakukan pembagian data maka data akan diproses menjadi token ID yang kemudian akan digunakan oleh model. Proses token ID memungkinkan model untuk memahami dan memproses teks menjadi numerik, yang diperlukan untuk pelatihan model dan inferensi dalam model pembelajaran mesin.

Setelah melakukan tokenisasi pada proses token ID, lalu proses akan berlanjut pada kelas *TranslationDataset* untuk menyiapkan data pelatihan dan validasi dalam format yang bisa diproses oleh *Pytroch* selama pelatihan model.

4.2.2.2 Pemuatan model dan pengaturan model

Langkah kedua dalam mengimplementasi program yaitu mempersiapkan model dan mengatur model agar model tersebut dapat digunakan untuk melatih data. Berikut segmen program pemuatan model dan pengaturan model.

```
model_name = "Helsinki-NLP/opus-mt-id-en"
model = MarianMTModel.from_pretrained(model_name)
# 6. Mengatur Argumen Pelatihan
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=100,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=10,
    eval_strategy="epoch",
    save_strategy="epoch"
)
```

Segmen Program 4. 2 Pemuatan model dan pengaturan pelatihan

Pada segmen program 4.2 pemuatan model dan pengaturan pelatihan akan dijelaskan fungsi dari kode diatas pada tabel 4.2

Tabel 4. 2 penjelasan pemuatan model dan pengaturan pelatihan

kelas	objek	keterangan
'TrainingArguments'	'training_args'	Menyimpan pengaturan pelatihan, seperti direktori output, jumlah epoch, ukuran batch, dan strategi logging.
'MarinModel'	'model_name'	Memuat model pre-trained 'Helsinki-NLP/opus-mt-id-en ' untuk melakukan terjemahan

Model MarianMT dimuat bersama dengan tokenizer, dan parameter pelatihan diatur menggunakan TrainingArguments. Pengaturan ini mencangkup aspek penting seperti jumlah epoch, ukuran batch, dan strategi logging.

4.2.2.3 Pelatihan model dan evaluasi

Langkah ketiga yaitu melatih model yang sudah disiapkan sebelumnya lalu melihat hasil pelatihan melalui evaluasi. Berikut segmen program pelatihan model dan evaluasi.

```
# 7. Menghitung Skor BLEU dan waktu pelatihan
bleu = load_evaluate("sacrebleu")
training_times = []
eval_losses = []
eval_bleus = []
eval_epochs = []
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    # Jika logits adalah tuple, ambil elemen pertama
    if isinstance(logits, tuple):
        logits = logits[0]
    # Jika logits adalah numpy array, konversi menjadi tensor
    if isinstance(logits, np.ndarray):
        logits = torch.tensor(logits)
    # Memastikan logits diubah ke bentuk argmax untuk mendapatkan
prediksi token ID
    pred_ids = torch.argmax(logits, dim=-1)
    # Decode prediksi menjadi teks
    pred_str = tokenizer.batch_decode(pred_ids,
skip_special_tokens=True)
    # Decode label menjadi teks
    labels = labels[:, 1:]  # Remove the starting token
    label_str = tokenizer.batch_decode(labels,
skip_special_tokens=True)
    # Menghitung BLEU score
    bleu_score = bleu.compute(predictions=pred_str,
references=[[label] for label in label_str])
    return {"bleu": bleu_score["score"]}
# 8. Melatih Model dengan waktu
start_time = time.time()
trainer = Trainer(
    model=model,
```

```
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    compute_metrics=compute_metrics
)
trainer.train()
end_time = time.time()
training_times.append(end_time - start_time)
# 9. Plot Hasil Evaluasi dan Simpan Stat BLEU
training_log = trainer.state.log_history
for log in training_log:
    if 'eval_loss' in log:
        eval_losses.append(log['eval_loss'])
        eval_bleus.append(log['eval_bleu'])
        eval_epochs.append(log['epoch'])
plt.figure(figsize=(10, 6))
plt.plot(eval_epochs, eval_losses, label='Evaluation Loss')
plt.plot(eval_epochs, eval_bleus, label='BLEU Score')
plt.xlabel('Epoch')
plt.ylabel('Metric')
plt.title('Evaluation Loss and BLEU Score over Epochs')
plt.legend()
plt.grid(True)
plt.savefig('./evaluation_metrics.png')
plt.show()
```

Segmen Program 4. 3 Pelatihan model dan evaluasi

Pada segmen program 4.3 pelatihan model dan evaluasi akan dijelaskan beberapa fungsi dari kode diatas pada tabel 4.3

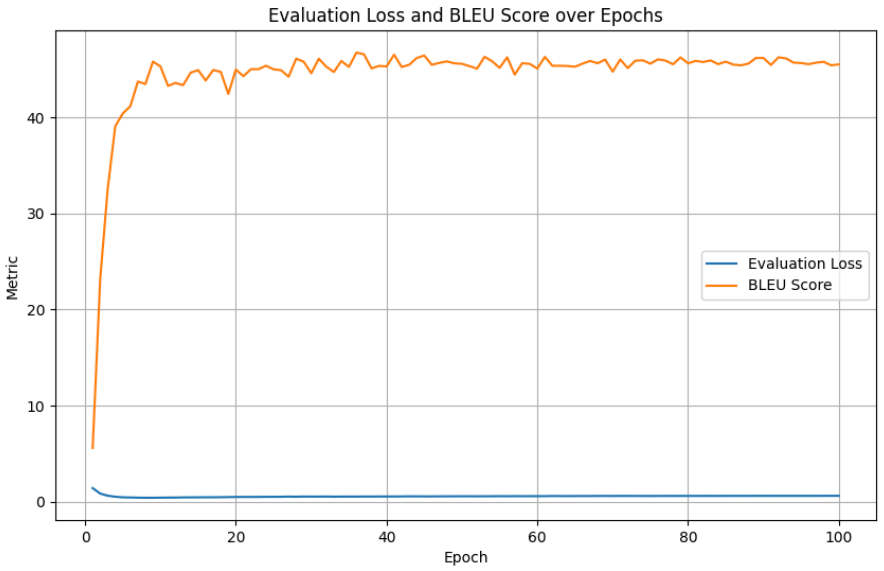
Tabel 4. 3 penjelasan pelatihan model dan evaluasi

Kelas	**Objek**	**Keterangan**
‘compute_metrics’	‘bleu_score’	Menghitung skor BLEU dari hasil terjemahan model selama evaluasi
‘Trainer’	‘trainer’	Melatih model dengan menggunakan ‘train_dataset’ untuk pelatihan dan ‘val_dataset’ untuk evaluasi, serta menghitung metrik dengan ‘compute_metrics’
‘plt.plot’	‘eval_epochs’, ‘eval_losses’, ‘eval_bleus’	Membuat plot dari epoch ‘eval_epoch’, loss ‘eval_loss’, dan skor BLEU ‘eval_bleus’ untuk

		memvisualisasikan kinerja model

Model dilatih menggunakan Trainer, dengan parameter yang telah ditentukan sebelumnya. Proses pelatihan dilakukan selama 100 *epoch*, dengan data pelatihan dan validasi yang telah disiapkan sebelumnya. Selama pelatihan, dilakukan evaluasi berkala menggunakan metrik *BLEU* untuk mengukur kualitas terjemahan yang dihasilkan oleh model.

Hasil pelatihan dan evaluasi ditampilkan dalam bentuk grafik yang menggambarkan perkembangan *Evaluation Loss* dan *BLEU Score* selama *epoch* pelatihan. Grafik pada gambar 4.1 dibawah menunjukkan bahwa *BLEU Score* meningkat dengan cepat pada awal pelatihan dan stabil setelah beberapa *epoch*, sementara *Evaluation Loss* tetap rendah, yang menandakan bahwa model tidak mengalami *overfitting*.



Gambar 4. 1 Hasil Evaluasi

4.2.2.4 Penyimpanan model dan Inferensi

Langkah keempat menyimpan model dan melakukan inferensi dari model yang sudah di simpan. Berikut segmen program penyimpanan model dan segmen program inferensi.

```
# 10. Mengatur dan Menyimpan Configurasi Generasi
generation_config = GenerationConfig(
    max_length=512,
    num_beams=6,
    bad_words_ids=[[54795]],
    bos_token_id=0,
    decoder_start_token_id=54795,
    eos_token_id=0,
    forced_eos_token_id=0,
    pad_token_id=54795,
    renormalize_logits=True,
)

# Simpan konfigurasi generasi
generation_config.save_pretrained('./TransformersModel_2908')
```

```
# Inferensi dengan Model yang Telah Dilatih
example_text = "aktivitas fisik teratur mendukung kesehatan jantung
dan sistem kekebalan tubuh"
inputs = tokenizer(example_text, return_tensors="pt", padding=True,
truncation=True)
translated_tokens = model.generate(**inputs,
generation_config=generation_config)
translation = tokenizer.decode(translated_tokens[0],
skip_special_tokens=True)
print(f"Original: {example_text}")
print(f"Translation: {translation}")
# 11. Menyimpan Model
model.save_pretrained('./TransformersModel_2908')
tokenizer.save_pretrained('./TransformersModel_2908')
generation_config.save_pretrained('./TransformersModel_2908')
```

Segmen Program 4. 4 Penyimpanan model

Pada segmen program 4.4 penyimpanan model akan dijelaskan beberapa fungsi dari kode diatas pada tabel 4.1

Tabel 4. 4 penjelasan penyimpanan model

Kelas	Objek	Keterangan
‘GenerationConfig’	‘generation_config’	Mengatur konfigurasi generasi teks seperti panjang maksimum ‘max_lenght’ , dan token lainnya untuk model MarianMT
‘model.save_pretrained’	‘model_path’	Menyimpan model yang telah dilatih, tokenizer, dan konfigurasi generasi ke direktori yang ditentukan ‘model_path’

```
# 1. Memuat model dan tokenizer dari direktori tempat model disimpan
model_path = 'model_NonModif/TransformersModel_2908'
model = MarianMTModel.from_pretrained(model_path)
tokenizer = MarianTokenizer.from_pretrained(model_path)

# 2. Memuat konfigurasi generasi (opsional)
generation_config = GenerationConfig.from_pretrained(model_path)

# 3. Contoh teks yang akan diterjemahkan
example_text = "aku pergi ke pasar"
```

```
# 4. Tokenisasi teks input
inputs = tokenizer(example_text, return_tensors="pt", padding=True,
truncation=True)

# 5. Melakukan inferensi dengan model menggunakan konfigurasi
generasi yang disimpan
translated_tokens = model.generate(**inputs,
generation_config=generation_config)

# 6. Mendekode hasil terjemahan menjadi teks
translation = tokenizer.decode(translated_tokens[0],
skip_special_tokens=True)

# 7. Menampilkan hasil terjemahan
print(f"Original: {example_text}")
print(f"Translation: {translation}")
```

Segmen Program 4. 5 Inferensi

Setelah pelatihan selesai, model dan konfigurasi generasi disimpan untuk penggunaan di masa depan. Model yang dilatih digunakan untuk inferensi teks dan menghasilkan terjemahan dari bahasa Indonesia ke bahasa Papua Kokas.

4.2.2.5 Website

Setelah model dibuat dan diuji, maka model tersebut diimplementasi pada *website* sederhana. Proses implementasi model pada website menggunakan *library Django*. *Django* adalah *framework* web yang ringan dan fleksibel dengan menggunakan bahasa pemrograman *python*. *Django* digunakan untuk menjembatani sistem website dengan pemanggilan model. Berikut segmen dari pemanggilan model.

```
from django.shortcuts import render
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from transformers import MarianMTModel, MarianTokenizer
import torch

# Memuat model dan tokenizer yang sudah disimpan
```

```
model_path = './transformers_model'  # Sesuaikan jalur jika model
berada di tempat lain
model = MarianMTModel.from_pretrained(model_path)
tokenizer = MarianTokenizer.from_pretrained(model_path)

def home(request):
    return render(request, 'translation_app/home.html')

def about(request):
    return render(request, 'translation_app/about.html')

@csrf_exempt
@csrf_exempt
def translate_text(request):
    if request.method == 'POST':
        input_text = request.POST.get('text')
        print("Received text: ", input_text)  # Debugging line for
input text

        # Tokenisasi teks input
        inputs = tokenizer(input_text, return_tensors="pt",
padding=True, truncation=True)

        # Melakukan inferensi dengan model
        translated_tokens = model.generate(**inputs)

        # Mendekode hasil terjemahan menjadi teks
        translation = tokenizer.decode(translated_tokens[0],
skip_special_tokens=True)

        print("Translation: ", translation)  # Debugging line for
output translation

        # Mengembalikan hasil terjemahan dalam format JSON
        return JsonResponse({'translation': translation})

    return JsonResponse({'error': 'Invalid request'}, status=400)
```

Segmen Program 4. 6 view.py

Setelah model dimuat pada kode diatas, lalu kelas untuk memuat model akan diarahkan menggunakan url. Berikut segmentasi program url.

```
from django.urls import path
from . import views

urlpatterns = [
```

```
    path('', views.home, name='home'),
    path('about/', views.about, name='about'),
    path('translate/', views.translate_text, name='translate_text'),
 # URL untuk terjemahan
]
```

Segmen Program 4. 7 url.py

Model yang telah diarahkan akan digunakan pada kelas home agar terjemahan langsung digunakan pada saat website di jalankan. Berikut segmentasi program home.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    {% load static %}
    <link rel="stylesheet" href="{% static
'translation_app/home.css' %}">
    <link rel="icon" href="{% static 'translation_app/favicon.ico'
%}" type="image/x-icon">
    <title>Home</title>
</head>
<body>
    <header class="header">
        <div class="title">
            <h2>ID-KKS</h2>
            <h2>Translation</h2>
        </div>
        <nav class="navbar">
            <a href="{% url 'about' %}">About</a>
        </nav>
    </header>
    <div class="container">
        <div class="translation-container">
            <div class="translation-box">
                <h3>Indonesia</h3>
                <textarea id="indonesia" placeholder="Masukkan teks
di sini..."></textarea>
            </div>
            <div class="translation-box">
                <h3>Papua Kokas</h3>
                <textarea id="papua-kokas" placeholder="Hasil
terjemahan akan muncul di sini..." readonly></textarea>
```

```
            </div>
        </div>
    </div>
    <!-- Pastikan script berada di bawah semua elemen -->
    <script>
        console.log("Before defining performTranslation function");

        // Fungsi debounce untuk menghindari pemanggilan fungsi
terjemahan terlalu sering
        function debounce(func, wait) {
            let timeout;
            return function(...args) {
                const context = this;
                clearTimeout(timeout);
                timeout = setTimeout(() => func.apply(context,
args), wait);
            };
        }
        // Fungsi untuk melakukan terjemahan
        function performTranslation() {
            var indonesiaText =
document.getElementById("indonesia").value;
            // Jika teks input kosong, kosongkan hasil terjemahan
juga
            if (indonesiaText === "") {
                document.getElementById("papua-kokas").value = "";
                return; // Keluar dari fungsi jika input kosong
            }
            // Lakukan terjemahan
            fetch("{% url 'translate_text' %}", {
                method: 'POST',
                headers: {
                    'Content-Type':
'application/x-www-form-urlencoded',
                    'X-CSRFToken': '{{ csrf_token }}'
                },
                body: new URLSearchParams({
                    'text': indonesiaText
                })
            })
            .then(response => {
                console.log("Response status: ", response.status);
                return response.json();
            })
            .then(data => {
                console.log("Response data: ", data);
                if (data.translation) {
```

```
                    document.getElementById("papua-kokas").value =
data.translation;
                } else {
                    alert("Terjadi kesalahan pada proses
terjemahan.");
                }
            })
            .catch(error => {
                console.error("Error during translation:", error);
            });
        }
        // Menggunakan debounce dengan waktu tunda 300ms (0.3 detik)
        const debouncedTranslation = debounce(performTranslation,
300);

document.getElementById("indonesia").addEventListener("input",
debouncedTranslation);
        console.log("After defining performTranslation function");
    </script>
</body>
</html>
```

Segmen Program 4. 8 Home.html

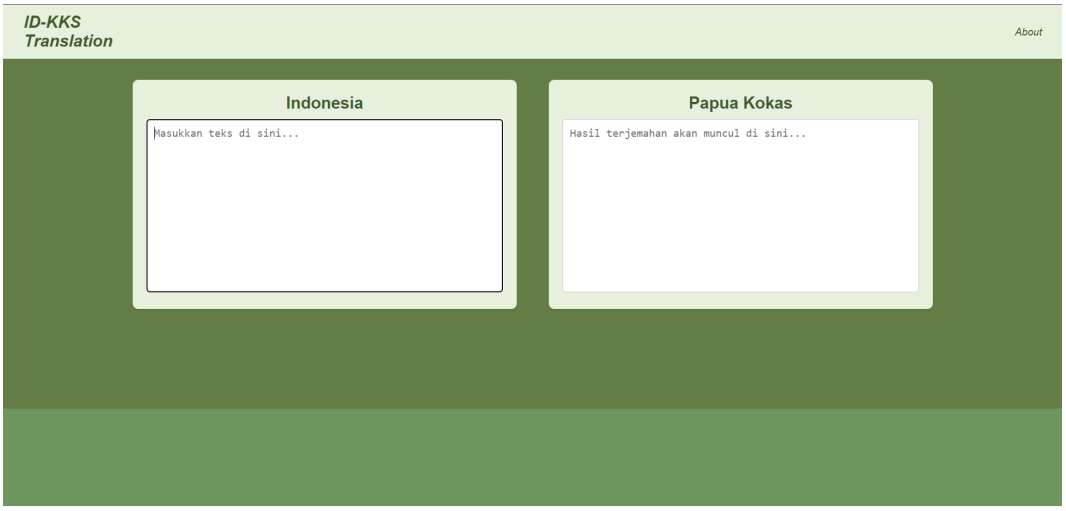
Kemudian kode akan dijalankan akan muncul seperti gambar dibawah. Kode *python*

```
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 20, 2024 - 19:09:31
Django version 5.0.7, using settings 'translation_project.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Gambar 4. 2 Output RunServer

Jika sudah berjalan seperti gambar diatas maka program akan memberikan alamat url untuk membuka website secara lokal, selanjutnya masuk ke website untuk memasukkan teks bahasa Indonesia dan diterjemahkan ke bahasa Papua Kokas.



Gambar 4. 3 tampilan website

Gambar diatas merupakan tampilan terjemahan website yang berisi kotak teks bahasa Indonesia dan kotak teks bahasa Papua Kokas.

Gambar 4. 4 Hasil terjemahan

Setelah kotak teks bahasa Indonesia diisi dengan kalimat seperti bagaimana kabarmu maka keluaran dari terjemahan yang dihasilkan berisi okabar akape.

4.3 Uji coba

4.3.1 Pengujian *Black Box*

Pengujian *Black Box* dilakukan untuk menguji proses *input* yang diberikan ke dalam *website* dan *output* dikeluarkan oleh *website*. Pengujian ini menguji respon dan perilaku *website* terhadap berbagai kondisi saat diuji

Tabel 4. 5 Pengujian BlackBox

Skenario Pengujian	Hasil yang Diharapkan	Hasil Pengujian	Keterangan
Menginput kalimat bahasa indonesia	Kalimat yang di input akan muncul pada kotak teks bahasa Indonesia	Kalimat berhasil diinputkan	Sesuai
Memproses kalimat yang di input	Kalimat terjemahan muncul pada kotak teks Papua Kokas	Kalimat berhasil diproses dan menghasilkan terjemahan	Sesuai

4.3.3 Pengujian *Matrik BLEU*

Pengujian Matrik Bleu dilakukan untuk menguji performa model yang telah dilatih sebelumnya. Berikut segmen program dari matrik Bleu.

```
import pandas as pd
import torch
import numpy as np
import matplotlib.pyplot as plt
import time
from sklearn.model_selection import train_test_split
from transformers import MarianTokenizer, MarianMTModel, Trainer,
TrainingArguments, GenerationConfig
from evaluate import load as load_evaluate
import openpyxl

# 1. Memuat Dataset
file_path = 'data_2908.csv'
df = pd.read_csv(file_path)

# 2. Membagi Dataset menjadi Training dan Validation
train_df, val_df = train_test_split(df, test_size=0.1)
print(f"Training data size: {len(train_df)}")
print(f"Validation data size: {len(val_df)}")

# 3. Memuat Model dan Tokenizer
```

```
model_name = "Helsinki-NLP/opus-mt-id-en"
tokenizer = MarianTokenizer.from_pretrained(model_name)
model = MarianMTModel.from_pretrained(model_name)

# 4. Memproses Data untuk Pelatihan
train_encodings = tokenizer(
    list(train_df['indonesian']),
    text_target=list(train_df['papua_kokas']),
    return_tensors="pt",
    padding=True,
    truncation=True
)

val_encodings = tokenizer(
    list(val_df['indonesian']),
    text_target=list(val_df['papua_kokas']),
    return_tensors="pt",
    padding=True,
    truncation=True
)

# Menyimpan label (karena ini bagian dari output yang perlu
digunakan untuk pembuatan dataset)
train_labels = train_encodings["labels"]
val_labels = val_encodings["labels"]

# 5. Membuat Dataset untuk PyTorch
class TranslationDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels

    def __getitem__(self, idx):
        item = {key: val[idx].clone().detach() for key, val in
self.encodings.items()}
        item['labels'] = self.labels[idx].clone().detach()
        return item

    def __len__(self):
        return len(self.labels)

train_dataset = TranslationDataset(train_encodings, train_labels)
val_dataset = TranslationDataset(val_encodings, val_labels)

# 6. Mengatur Argumen Pelatihan
training_args = TrainingArguments(
    output_dir='./results',
    num_train_epochs=100,
```

```
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
    warmup_steps=500,
    weight_decay=0.01,
    logging_dir='./logs',
    logging_steps=10,
    eval_strategy="epoch",
    save_strategy="epoch"
)

# 7. Menghitung Skor BLEU dan waktu pelatihan
bleu = load_evaluate("sacrebleu")
training_times = []
eval_losses = []
eval_bleus = []
eval_epochs = []

def compute_metrics(eval_pred):
    logits, labels = eval_pred

    # Jika logits adalah tuple, ambil elemen pertama
    if isinstance(logits, tuple):
        logits = logits[0]

    # Jika logits adalah numpy array, konversi menjadi tensor
    if isinstance(logits, np.ndarray):
        logits = torch.tensor(logits)

    # Memastikan logits diubah ke bentuk argmax untuk mendapatkan
prediksi token ID
    pred_ids = torch.argmax(logits, dim=-1)

    # Decode prediksi menjadi teks
    pred_str = tokenizer.batch_decode(pred_ids,
skip_special_tokens=True)

    # Decode label menjadi teks
    labels = labels[:, 1:]  # Remove the starting token
    label_str = tokenizer.batch_decode(labels,
skip_special_tokens=True)

    # Menghitung BLEU score
    bleu_score = bleu.compute(predictions=pred_str,
references=[[label] for label in label_str])

    return {"bleu": bleu_score["score"]}

# 8. Melatih Model dengan waktu
```

```
start_time = time.time()

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=val_dataset,
    compute_metrics=compute_metrics
)

trainer.train()

end_time = time.time()
training_times.append(end_time - start_time)
```

Segmen Program 4. 9 Matrik Bleu

Pada pelatihan ini dilakukan penghitungan jumlah skor bleu pada setiap kali di *epoch*, *epoch* yang dilakukan selama 100 kali yang berarti data yang dilihat sebanyak 100 kali, berikut hasil dari skor bleu yang diambil secara acak.

Tabel 4. 6 Skor Bleu

Epoch	Eval Loss	BLEU Score	Training Time (s)
1	1.437492	5.615219	44432.38
2	0.866988	23.18221	44432.38
3	0.644426	32.60257	44432.38
4	0.537822	39.08235	44432.38
5	0.477115	40.40219	44432.38
45	0.574157	46.42361	44432.38
46	0.574115	45.46316	44432.38
47	0.577398	45.6626	44432.38
48	0.582214	45.81687	44432.38
49	0.584458	45.61366	44432.38
50	0.586556	45.56784	44432.38
51	0.587614	45.3197	44432.38
52	0.583926	45.06312	44432.38
53	0.584521	46.29453	44432.38
54	0.588642	45.82968	44432.38
55	0.597317	45.14942	44432.38
95	0.631597	45.64443	44432.38
96	0.631917	45.52492	44432.38
97	0.632621	45.68719	44432.38

98	0.633608	45.77077	44432.38
99	0.633763	45.40761	44432.38
100	0.633533	45.50379	44432.38

Pengujian dilakukan untuk menilai kualitas terjemahan yang dihasilkan oleh model penerjemahan otomatis dari bahasa Indonesia ke bahasa Papua Kokas menggunakan arsitektur *Transformers* dengan *framework MariaMT*. Salah satu metrik evaluasi yang digunakan adalah *BLEU* (*Bilingual Evaluation Understudy*), yang mengukur seberapa dekat terjemahan yang dihasilkan oleh model dengan terjemahan.

Hasil Pengujian :

Dari hasil pengujian, *BLEU score* mengalami peningkatan seiring dengan jumlah epoch yang digunakan dalam pelatihan model. Hal ini menunjukkan bahwa semakin baik dalam menerjemahkan kalimat setelah mengalami lebih banyak pelatihan.

Pada *epoch* ke -5 mengalami *BLEU score* 40.40219, lalu pada epoch ke – 50 mengalami peningkatan *BLEU score* 45.56784 dan pada *epoch* ke – 53 mengalami peningkatan 46.29453, tapi pada *epoch* ke – 100 mengalami penurunan *BLEU score* 45.5079, yang mengindikasikan bahwa model telah mencapai konvergensi dan peningkatan skor *BLEU* setelahnya tidak signifikan.

Keseluruhan proses pelatihan menunjukkan tren yang positif, dimana penurunan nilai *Eval Loss* yang konsisten juga mendukung peningkatan skor

BLEU. Namun. Tidak terjadi overfitting yang berarti skor *BLEU* tetap stabil meski *Eval Loss* menurun.

Berdasarkan hasil pengujian diatas, matrik *BLEU* menunjukkan bahwa model penerjemah otomatis berbasis arsitektur *Transformers* ini memiliki performa yang cukup baik dalam menerjemahkan dari bahasa Indonesia ke bahasa Papua Kokas. Hasil terjemahan mendekati kualitas yang diharapkan sesuai dengan referensi, terutama setelah model dilatih dalam beberapa *epoch*. Peningkatan nilai *BLUE* yang signifikan pada awal pelatihan menunjukkan bahwa model dengan cepat mempelajari pola terjemahan, namun stabilitas pada *epoch* akhir menunjukkan bahwa model telah mencapai titik optimal untuk dataset yang digunakan.

BAB V
KESIMPULAN

5.1 Kesimpulan

Penelitian ini berfokus dalam penerapan arsitektur Transformers untuk penerjemahan otomatis dari bahasa Indonesia ke bahasa Papua Kokas. Berdasarkan analisis objektif yang dilakukan, beberapa kesimpulan dapat diambil sebagai berikut :

1. Implementasi Arsitektur Transformers pada penerjemahan Bahasa Indonesia ke Bahasa Papua Kokas dilakukan dengan menggunakan **MarianMT framework**. Model yang dihasilkan telah diuji dengan dataset yang terbatas, yang berisi **2908 pasangan kalimat**, dan sistem ini mampu melakukan proses penerjemahan secara otomatis. Dalam konteks penelitian ini, model diterapkan pada **website sederhana** untuk memberikan kemampuan penerjemahan secara real-time.
2. Berdasarkan evaluasi yang dilakukan dengan **metrik BLEU**, hasil menunjukkan bahwa model berhasil menghasilkan terjemahan yang akurat. Evaluasi dilakukan secara berkala setiap epoch untuk memastikan model tidak mengalami overfitting, Hal ini memberikan gambaran bahwa performa model dapat diukur secara objektif dan hasil evaluasi menunjukkan **peningkatan skor BLEU** yang stabil selama proses pelatihan.
3. Dari pengujian yang dilakukan, model berhasil mencapai **skor BLEU yang cukup memadai**, menunjukkan bahwa terjemahan yang dihasilkan mendekati referensi yang diharapkan. Pengujian secara **Black Box**

menunjukkan bahwa sistem mampu merespon input dengan baik dan menghasilkan output yang konsisten dengan teks referensi. Selain itu, **pengujian inferensi** juga dilakukan untuk memastikan bahwa model dapat bekerja dengan lancar dalam aplikasi yang dirancang.

5.2 Saran

Berdasarkan temuan dari penelitian yang telah dilakukan oleh penulis, terdapat beberapa rekomendasi untuk pengembangan selanjutnya, yaitu :

1. **Perluasan dataset** : Mengingat dataset yang digunakan terbatas pada 2908 kalimat, disarankan untuk memperluas jumlah data dengan menambahkan lebih banyak korpus teks dalam bahasa Indonesia dan kokas. Hal ini dapat meningkatkan akurasi terjemahan dan memastikan model dapat menangani lebih banyak variasi bahasa.
2. **Penggunaan Teknik Optimasi Lain** : Selain menggunakan BLEU, disarankan untuk menggunakan evaluasi lain seperti TER (Translation Edit Rate) atau METEOR, yang dapat memberikan gambaran lebih untuk membandingkan kualitas performa model.
3. **Implementasi Bahasa daerah lain** : Mengingat keberhasilan model ini dalam menerjemahkan Bahasa Indonesia ke Bahasa Papua Kokas, model ini bisa diadaptasi untuk bahasa-bahasa daerah lainnya di Indonesia. Pengembangan lebih lanjut dapat mengarah pada pelestarian bahasa daerah lainnya yang menghadapi ancaman kepunahan.

DAFTAR PUSTAKA

Abidin, Z. (2017). Penerapan Neural Machine Translation untuk Eksperimen Penerjemahan secara Otomatis pada Bahasa Lampung – Indonesia. *Prosiding Seminar Nasional Metode Kuantitatif*, *978*, 53–68.

Adi, Kristian, N., & Sebastian, D. (2018). Pembentukan Dataset Topik Kata Bahasa Indonesia pada Twitter Menggunakan TF-IDF & Cosine Similarity. *Jurnal Teknik Informatika Dan Sistem Informasi*, *4*, 2443–2229. http://dx.doi.org/10.28932/jutisi.v4i3.862

Amarulloh, A., Kurniasih, K., & Muchlis, M. (2023). ANALISIS PERBANDINGAN PERFORMA WEB SERVICE REST MENGGUNAKAN FRAMEWORK LARAVEL, DJANGO, DAN Node JS PADA APLIKASI BERBASIS WEBSITE. *Jurnal Teknik Informatika*, *9*(1), 14–19.

Apriani, T., Sujaini, H., & Safriadi, N. (2016). Pengaruh kuantitas korpus terhadap akurasi mesin penerjemah statistik bahasa Bugis Wajo ke bahasa Indonesia. *JUSTIN (Jurnal Sistem Dan Teknologi Informasi)*, *1*(1), 1–6.

Aulamo, M., & Boggia, M. (2023). *arXiv : 2212 . 01936v3 [cs . CL] 4 Jul 2023 Democratizing Neural Machine Translation with OPUS-MT*.

Fauziyah, Y., Ilyas, R., & Kasyidi, F. (2022). Mesin Penterjemah Bahasa Indonesia-Bahasa Sunda Menggunakan Recurrent Neural Networks. *Jurnal Teknoinfo*, *16*(2), 313. https://doi.org/10.33365/jti.v16i2.1930

Lavie, A. (2010). Evaluating the output of machine translation systems. *AMTA 2010 - 9th Conference of the Association for Machine Translation in the Americas*.

Meilinda, A. D., Sujaini, H., & Safriadi, N. (2023). *Pivot Language Bahasa Melayu Pontianak ke Bahasa Bugis Menggunakan Neural Machine Translation*. *9*(2), 234–241.

Ott, M., Auli, M., Grangier, D., & Aurelio, M. (2018). *Analyzing Uncertainty in Neural Machine Translation*.

Purnama, I. N., & Utami, N. N. W. (2023). Implementasi Peringkas Dokumen Berbahasa Indonesia Menggunakan Metode Text To Text Transfer Transformer (T5). *Jurnal Teknologi Informasi Dan Komputer*, *9*(4), 381–391.

Rendragraha, A. D., Bijaksana, M. A., & Romadhony, A. (2021). Pendekatan Metode Transformers untuk Deteksi Bahasa Kasar dalam Komentar Berita Online Indonesia. *E-Proceeding of Engineering*, *8*(2), 3385–3395.

Roihan, A., Sunarya, P. A., & Rafika, A. S. (2020). Pemanfaatan Machine Learning dalam Berbagai Bidang: Review paper. *IJCIT (Indonesian Journal on Computer and Information Technology)*, *5*(1), 75–82. https://doi.org/10.31294/ijcit.v5i1.7951

Saputra, D. (2018). Analisis Perbandingan Performa Web Service Rest Menggunakan Framework Laravel, Django Dan Ruby On Rails Untuk Akses Data Dengan. *Jurnal Bangkit Indonesia*, *7*(2), 17. https://doi.org/10.52771/bangkitindonesia.v7i2.90

Shah, P., & Bakrola, V. (2019). Neural machine translation system of indic languages-an attention based approach. *2019 2nd International Conference on Advanced Computational and Communication Paradigms, ICACCP 2019*. https://doi.org/10.1109/ICACCP.2019.8882969

Shaw, P., Uszkoreit, J., & Vaswani, A. (2018). Self-attention with relative position representations. *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*, *2*, 464–468. https://doi.org/10.18653/v1/n18-2074

Sulistiyo, N. A. (2016). *PEMBUATAN APLIKASI M-LEARNING SPOK BAHASA INDONESIA UNTUK BLACKBERRY DENGAN MENGGUNAKAN SUN JAVA WIRELESS TOOLKIT DAN JAVA DEVELOPMENT ENVIRONMENT*. 1–23.

Sumatera, P. D., & Rahima, A. (2024). *Revitalisasi bahasa dokumentasi bahasa*. *3*(1), 56–61.

Supriyati, E., & Iqbal, M. (2018). Pengukuran Similarity Tema Pada Juz 30 Al Qur'an Menggunakan Teks Klasifikasi. *Simetris: Jurnal Teknik Mesin, Elektro Dan Ilmu Komputer*, *9*(1), 361–370. https://doi.org/10.24176/simet.v9i1.1955

van der Wees, M., Bisazza, A., & Monz, C. (2017). Dynamic data selection for neural machine translation. *EMNLP 2017 - Conference on Empirical Methods in Natural Language Processing, Proceedings*, 1400–1410. https://doi.org/10.18653/v1/d17-1147

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

Wahyuni, R. T., Prastiyanto, D., & Supraptono, E. (2017). Penerapan Algoritma Cosine Similarity dan Pembobotan TF-IDF pada Sistem Klasifikasi Dokumen Skripsi. *Jurnal Teknik Elektro Universitas Negeri Semarang*, *9*(1), 18–23. https://journal.unnes.ac.id/nju/index.php/jte/article/download/10955/6659

Yin, K. (2020). *SIGN LANGUAGE TRANSLATION WITH TRANSFORMERS*.

LAMPIRAN

1. Surat Keputusan Tugas Akhir (SK TA)
2. Biodata Penulis
3. Hasil cek plagiarism
4. Dokumen pendukung penelitian (bukti wawancara, form isian, format laporan, standar operasional prosedur, aturan – aturan yang berlaku, dll)
5. Listing Program/Coding

Lampiran 1 Surat Keputusan Tugas Akhir (SK TA)

SEKOLAH TINGGI INFORMATIKA & KOMPUTER INDONESIA
STIKI MALANG
Jl. Raya Tidar 100, Malang 65146, Telp. (0341) 560823, Fax. (0341) 562525
Website: www.stiki.ac.id Email: stiki@stiki.ac.id

SURAT KEPUTUSAN

No.163.9 /AKD.BAA.08/P1/STIKI/VIII/2024

TENTANG
PELAKSANAAN TUGAS AKHIR

WAKIL KETUA I SEKOLAH TINGGI INFORMATIKA & KOMPUTER INDONESIA MALANG :

MENIMBANG :
1. Bahwa dalam rangka menyelesaikan persyaratan kelulusan jenjang studi, mahasiswa perlu menyusun Tugas Akhir sebagai hasil implementasi dan pengembangan kemampuan akademik.
2. Bahwa untuk menyelesaikan Tugas Akhir, mahasiswa perlu memperhatikan dan mematuhi peraturan, pedoman yang berlaku.

MEMPERHATIKAN :
1. Peraturan Akademik
2. Pedoman Tugas Akhir
3. Hasil Seminar Proposal Tugas Akhir

MEMUTUSKAN

MENETAPKAN : 1. Tugas Akhir Mahasiswa

Nama	:	**Ronggo Haikal**
NRP	:	**191111014**
Program Studi	:	**S1-IF**
Judul	:	**Implementasi Arsitektur Transformers Penerjemahan Otomatis Bahasa Indonesia ke Bahasa Papua (Studi kasus Daerah Kokas)**
Dosen Pembimbing	:	**Nira Radita, S.Pd., M.Pd**
Co. Pembimbing	:	**Mukhlis Amien, S.Kom., M.Kom**

2. Ketentuan Pelaksanaan dan Penyelesaian Tugas Akhir sebagai berikut :
   a. Mahasiswa wajib melakukan konsultasi dengan Dosen Pembimbing / Dosen Co. Pembimbing minimal 12 kali dan mengisi **Log Book Bimbingan Tugas Akhir** secara luring minimal 2 minggu sekali .
   b. Mahasiswa wajib melaksanakan tahapan TA , mengikuti jadwal sbb :

Tahapan TA	Batas Akhir Pelaksanaan
Pra Seminar II	**22 Oktober 2024**
Seminar TA/Sidang TA	**22 Januari 2025**

   c. Jika sampai dengan **batas akhir waktu pelaksanaan Tugas Akhir**, mahasiswa belum melaksanakan Seminar Tugas Akhir maka mahasiswa **wajib mengajukan permohonan perpanjangan paling lambat 2 minggu setelah batas berakhirnya SK**.
   d. Jika ketentuan pada point 2.c tidak dilaksanakan, maka **Tugas Akhir mahasiswa dinyatakan gugur**, dan mahasiswa wajib melakukan proses permohonan ulang.
   e. Mahasiswa wajib menyelesaikan seluruh tanggungan akademik maupun keuangan untuk dapat melaksanakan seminar Tugas Akhir.

Surat keputusan ini berlaku sejak tanggal ditetapkan dan apabila dikemudian hari terdapat kekeliruan dalam keputusannya akan dilakukan pembetulan.

Ditetapkan : **di Malang**
Pada Tanggal : **20 Agustus 2024**
Wakil Ketua I STIKI,

Daniel Rudiaman Sijabat., S.T., M.Kom.

Tembusan :
1. Yth. WAKET II & Ka.Prodi
2. Mahasiswa/i yang bersangkutan

Lampiran 2 Biodata Penulis

BIODATA PENULIS

Nama	:	Ronggo Haikal
Alamat	:	Jl. Melati Raya KM 9,5, RT 004/RW 003, Kel KLASABI, Kec SORONG MANOI, Kota Sorong, Papua Barat Daya
Tempat/Tanggal Lahir	:	Sorong, 01 Mei 2001
Telp. / Email	:	082199223418/ronggohaikal05@gmail.com

PENDIDIKAN

No	Pendidikan	Tempat	Tahun Lulus		Gelar	Bidang Spesialisasi
			Mulai	Lulus		
	SD	MI AL-MAARIF	2007	2013		
	SMP	SMP MUHAMMADIYAH AL-AMIN	2013	2016		
	SMK	SMK NEGERI 3 KOTA SORONG	2016	2019		ELEKTRONIKA INDUSTRI

PEKERJAAN

No	Pekerjaan	Bidang	Perusahaan	Tahun

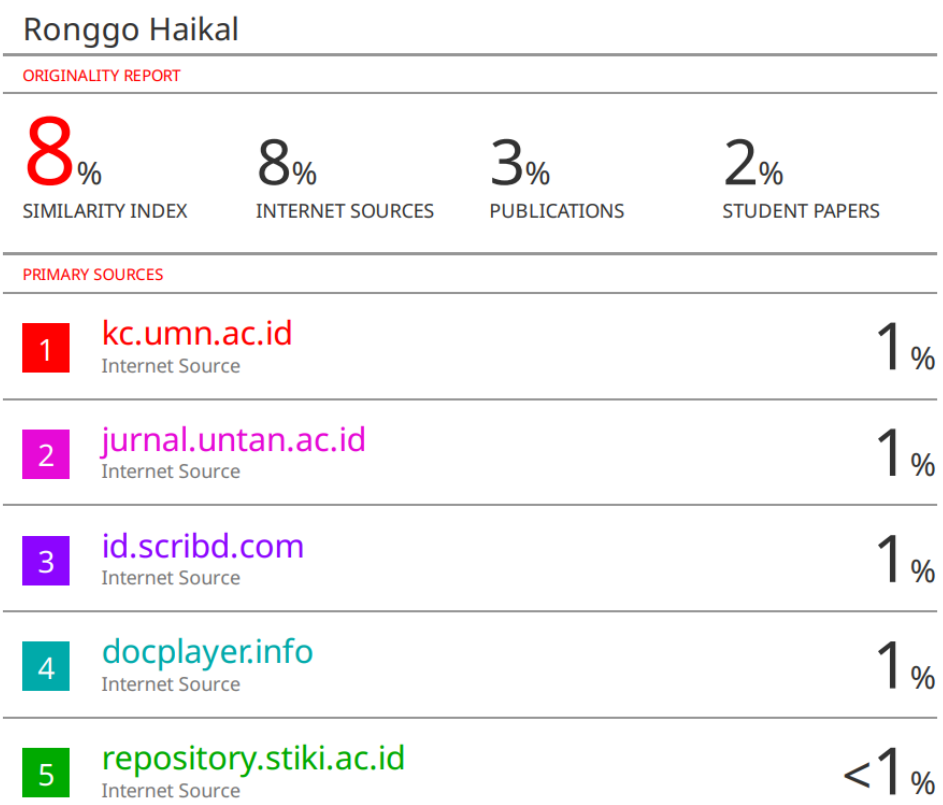
PRESTASI

No	Prestasi	Bidang	Penyelenggara	Tahun

SERTIFIKASI KOMPETENSI

No	Sertifikasi	Penyelenggara	Tahun
1	Pemograman Basis Data	Lembaga Sertifikasi Profesi STIKI Malang	2022
2	Web Developer	Lembaga Sertifikasi Profesi STIKI Malang	2023
3	TOEFL	PPTIK	2024

Lampiran 3 Hasil Cek Plagiarisme



Ronggo Haikal

ORIGINALITY REPORT

8%	8%	3%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	kc.umn.ac.id Internet Source	1%
2	jurnal.untan.ac.id Internet Source	1%
3	id.scribd.com Internet Source	1%
4	docplayer.info Internet Source	1%
5	repository.stiki.ac.id Internet Source	<1%

Lampiran 4 Dokumen pendukung penelitian

Lampiran 5 Listing Program/Coding

Memuat Dataset

```
import pandas as pd
import torch
import numpy as np
import matplotlib.pyplot as plt
import time
from sklearn.model_selection import train_test_split
from transformers import MarianTokenizer, MarianMTModel, Trainer,
TrainingArguments, GenerationConfig
from evaluate import load as load_evaluate
import openpyxl
# 1. Memuat Dataset
file_path = 'data_2908.csv'
df = pd.read_csv(file_path)
# 2. Membagi Dataset menjadi Training dan Validation
train_df, val_df = train_test_split(df, test_size=0.1)
print(f"Training data size: {len(train_df)}")
print(f"Validation data size: {len(val_df)}")
```

Listing Program 1 Memuat data

Tokenisasi

```
# Bagian tokenisasi
# Memuat tokenizer dari model pre-trained
model_name = "Helsinki-NLP/opus-mt-id-en"
tokenizer = MarianTokenizer.from_pretrained(model_name)
# Tokenisasi dan encoding data pelatihan
train_encodings = tokenizer(
    list(train_df['indonesian']),          # Kolom teks bahasa
Indonesia
    text_target=list(train_df['papua_kokas']),  # Kolom teks bahasa
Papua Kokas
    return_tensors="pt",                   # Mengembalikan sebagai
tensor PyTorch
    padding=True,                          # Padding otomatis agar
panjang sama
    truncation=True                        # Truncation otomatis
untuk teks yang terlalu panjang
)
# Tokenisasi dan encoding data validasi
val_encodings = tokenizer(
    list(val_df['indonesian']),
    text_target=list(val_df['papua_kokas']),
    return_tensors="pt",
    padding=True,
    truncation=True
```

```
)
```

Listing Program 2 Tokenisasi

Pembuatan dataset

```
# Membuat kelas custom untuk dataset
class TranslationDataset(torch.utils.data.Dataset):
    def __init__(self, encodings, labels):
        self.encodings = encodings
        self.labels = labels
    def __getitem__(self, idx):
        # Mengambil item berdasarkan index
        item = {key: val[idx].clone().detach() for key, val in
self.encodings.items()}
        item['labels'] = self.labels[idx].clone().detach()
        return item
    def __len__(self):
        # Mengembalikan panjang dataset
        return len(self.labels)
# Membuat dataset PyTorch untuk data pelatihan dan validasi
train_dataset = TranslationDataset(train_encodings,
train_encodings["labels"])
val_dataset = TranslationDataset(val_encodings,
val_encodings["labels"])
# Mengelola dataset ke dalam DataLoader PyTorch
train_loader = DataLoader(train_dataset, batch_size=32,
shuffle=True)
val_loader = DataLoader(val_dataset, batch_size=32, shuffle=False)
```

Listing Program 3 Pembuatan Dataset

Pemuatan model

```
# Memuat model pre-trained dari Helsinki-NLP/opus-mt-id-en
model_name = "Helsinki-NLP/opus-mt-id-en"
tokenizer = MarianTokenizer.from_pretrained(model_name)
model = MarianMTModel.from_pretrained(model_name)
```

Listing Program 4 Pembuatan Model

Pengaturan parameter

```
# Pengaturan parameter pelatihan menggunakan TrainingArguments
training_args = TrainingArguments(
    output_dir='./results',              # Direktori output
    num_train_epochs=100,                # Jumlah epoch
    per_device_train_batch_size=16,      # Ukuran batch untuk
pelatihan
    per_device_eval_batch_size=16,       # Ukuran batch untuk
evaluasi
    warmup_steps=500,                    # Jumlah langkah warmup
```

```
    weight_decay=0.01,                  # Tingkat weight decay
untuk regularisasi
    logging_dir='./logs',               # Direktori untuk menyimpan
log
    logging_steps=10,                   # Frekuensi logging selama
pelatihan
    eval_strategy="epoch",              # Strategi evaluasi
(misalnya setiap epoch)
    save_strategy="epoch",              # Strategi penyimpanan
model (misalnya setiap epoch)
)
```

Listing Program 5 Pengaturan parameter

Pelatihan model

```
# 1. Membuat objek Trainer dengan model, data, dan parameter
pelatihan
trainer = Trainer(
    model=model,                    # Model yang dilatih
    args=training_args,             # Parameter pelatihan yang
sudah diatur sebelumnya
    train_dataset=train_dataset,    # Dataset untuk pelatihan
    eval_dataset=val_dataset,       # Dataset untuk evaluasi
    compute_metrics=compute_metrics  # Fungsi untuk menghitung
metrik, seperti BLEU
)
# 2. Melatih model
trainer.train()
```

Listing Program 6 Pelatihan model

Perhitungan metrik *BLEU*

```
# Memuat evaluator BLEU
bleu = load_evaluate("sacrebleu")
def compute_metrics(eval_pred):
    logits, labels = eval_pred
    # Jika logits adalah tuple, ambil elemen pertama
    if isinstance(logits, tuple):
        logits = logits[0]
    # Jika logits adalah numpy array, konversi menjadi tensor
    if isinstance(logits, np.ndarray):
        logits = torch.tensor(logits)
    # Memastikan logits diubah ke bentuk argmax untuk mendapatkan
prediksi token ID
    pred_ids = torch.argmax(logits, dim=-1)
    # Decode prediksi menjadi teks
    pred_str = tokenizer.batch_decode(pred_ids,
skip_special_tokens=True)
    # Decode label menjadi teks
```

```
    labels = labels[:, 1:]  # Menghapus token awal
    label_str = tokenizer.batch_decode(labels,
skip_special_tokens=True)
    # Menghitung BLEU score
    bleu_score = bleu.compute(predictions=pred_str,
references=[[label] for label in label_str])
    return {"bleu": bleu_score["score"]}
```

Listing Program 7 Pengaturan Matrik BLEU

Pengaturan konfigurasi

```
generation_config = GenerationConfig(
    max_length=512,                    # Panjang maksimum teks yang
dihasilkan
    num_beams=6,                       # Jumlah beams yang digunakan
dalam beam search
    bad_words_ids=[[54795]],           # Daftar ID token yang tidak
boleh muncul dalam hasil
    bos_token_id=0,                    # ID token awal (beginning of
sequence)
    decoder_start_token_id=54795,      # ID token awal untuk decoder
    eos_token_id=0,                    # ID token akhir (end of
sequence)
    forced_eos_token_id=0,             # Memaksa penggunaan ID token
akhir di akhir teks yang dihasilkan
    pad_token_id=54795,                # ID token padding untuk
mengisi ruang kosong dalam sequence
    renormalize_logits=True,           # Menormalkan kembali logits
selama inferensi
)
```

Listing Program 8 Pengaturan konfigurasi

Evaluasi akhir

```
# Menyimpan statistik selama pelatihan
training_times = []
eval_losses = []
eval_bleus = []
eval_epochs = []
# Melakukan evaluasi setelah setiap epoch
training_log = trainer.state.log_history
for log in training_log:
    if 'eval_loss' in log:
        eval_losses.append(log['eval_loss'])
        eval_bleus.append(log['eval_bleu'])
        eval_epochs.append(log['epoch'])
# Plot hasil evaluasi
plt.figure(figsize=(10, 6))
plt.plot(eval_epochs, eval_losses, label='Evaluation Loss')
```

```
plt.plot(eval_epochs, eval_bleus, label='BLEU Score')
plt.xlabel('Epoch')
plt.ylabel('Metric')
plt.title('Evaluation Loss and BLEU Score over Epochs')
plt.legend()
plt.grid(True)
plt.show() # Menunjukkan hasil evaluasi dalam bentuk gambar
# Menyimpan statistik dalam bentuk file Excel
import pandas as pd
df_stats = pd.DataFrame({
    'Epoch': eval_epochs,
    'Eval Loss': eval_losses,
    'BLEU Score': eval_bleus
})
df_stats.to_excel('training_stats.xlsx', index=False)
```

Listing Program 9 Evaluasi akhir

Penyimpanan model

```
# Menyimpan model yang telah dilatih
model.save_pretrained('./trained_transformers_model')
# Menyimpan tokenizer yang digunakan untuk melatih model
tokenizer.save_pretrained('./trained_transformers_model')
# Menyimpan konfigurasi generasi
generation_config.save_pretrained('./trained_transformers_model')
```

Listing Program 10 Penyimpanan model

Inferensi model

```
from transformers import MarianMTModel, MarianTokenizer,
GenerationConfig

# 1. Memuat model dan tokenizer dari direktori tempat model disimpan
model_path = 'model_NonModif/TransformersModel_2908'
model = MarianMTModel.from_pretrained(model_path)
tokenizer = MarianTokenizer.from_pretrained(model_path)

# 2. Memuat konfigurasi generasi (opsional)
generation_config = GenerationConfig.from_pretrained(model_path)

# 3. Contoh teks yang akan diterjemahkan
example_text = "aku pergi ke pasar"

# 4. Tokenisasi teks input
inputs = tokenizer(example_text, return_tensors="pt", padding=True,
truncation=True)

# 5. Melakukan inferensi dengan model menggunakan konfigurasi
generasi yang disimpan
```

```
translated_tokens = model.generate(**inputs,
generation_config=generation_config)

# 6. Mendekode hasil terjemahan menjadi teks
translation = tokenizer.decode(translated_tokens[0],
skip_special_tokens=True)

# 7. Menampilkan hasil terjemahan
print(f"Original: {example_text}")
print(f"Translation: {translation}")
```

Listing Program 11 Inferensi

Memuat Model pada website

```
from django.shortcuts import render
from django.http import JsonResponse
from django.views.decorators.csrf import csrf_exempt
from transformers import MarianMTModel, MarianTokenizer
import torch

# Memuat model dan tokenizer yang sudah disimpan
model_path = './TransformersModel_2908'  # Sesuaikan jalur jika
model berada di tempat lain
model = MarianMTModel.from_pretrained(model_path)
tokenizer = MarianTokenizer.from_pretrained(model_path)

def home(request):
    return render(request, 'translation_app/home.html')

def about(request):
    return render(request, 'translation_app/about.html')

@csrf_exempt
@csrf_exempt
def translate_text(request):
    if request.method == 'POST':
        input_text = request.POST.get('text')
        print("Received text: ", input_text)  # Debugging line for
input text

        # Tokenisasi teks input
        inputs = tokenizer(input_text, return_tensors="pt",
padding=True, truncation=True)

        # Melakukan inferensi dengan model
        translated_tokens = model.generate(**inputs)
```

```
        # Mendekode hasil terjemahan menjadi teks
        translation = tokenizer.decode(translated_tokens[0],
skip_special_tokens=True)

        print("Translation: ", translation)  # Debugging line for
output translation

        # Mengembalikan hasil terjemahan dalam format JSON
        return JsonResponse({'translation': translation})

    return JsonResponse({'error': 'Invalid request'}, status=400)
```

Listing Program 12 Memuat model pada website

Pengalihan Model ke Halaman utama melalui Url

```
from django.urls import path
from . import views

urlpatterns = [
    path('', views.home, name='home'),
    path('about/', views.about, name='about'),
    path('translate/', views.translate_text, name='translate_text'),
 # URL untuk terjemahan
]
```

Listing Program 13 URL

Halaman utama website

```
#!/usr/bin/env python
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    """Run administrative tasks."""
    os.environ.setdefault('DJANGO_SETTINGS_MODULE',
'translation_project.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and
"
            "available on your PYTHONPATH environment variable? Did
you "
            "forget to activate a virtual environment?"
```

```
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

Listing Program 14 Home page