(2026), 4 (1): 305-327 **Scientica** 3021-8209

Jurnal Ilmiah Sain dan Teknologi

ANALISIS NILAI AKURASI PADA *NEURAL MACHINE TRANSLATION* (STUDI KASUS BAHASA INDONESIA KE BAHASA MELAYU KETAPANG)

Ainayyah Pratiwi [a1], Arif Bijaksana Putra Negara [b2], Tursina [b3]

Program Studi Sarjana Informatika Fakultas Teknik Universitas Tanjungpura Jl. Prof. Dr. H. Hadari Nawawi,

Pontianak, Kalimantan Barat 78124

[1] ainayyah2799@student.untan.ac.id [2] arifbpn@informatika.untan.ac.id

[3] tursina@informatika.untan.ac.id

Abstract

Language is a tool that people use to communicate. Indonesia has Indonesian as the national language. Indonesia itself has a lot of regions and regional languages that accompany it, one of which is the Ketapang Malay language commonly used by the people of Ketapang Regency. Ketapang Malay language is almost extinct so it requires technological development as a tool to maintain its preservation. One of the technologies that can help preserve local languages is by making a translation machine. In various studies that have been conducted by researchers in Indonesia, the application of machine translation for local languages is still based on statistics (SMT), while in recent years neural network machine translation (NMT) has become a new method in the practice of machine translation. The Neural Machine Translation used in this research utilizes the attention mechanism and transformer architecture by using a parallel corpus of Indonesian and Ketapang Malay as many as 5000 lines of sentences. This study aims to determine the results of the translation accuracy value on Neural Machine Translation using the same parallel corpus as previous research, namely research conducted by Dinar (2020) using a statistical translation machine. The results of automatic testing for the statistical network translation engine by BLEU with a corpus that has been optimized by Dinar (2020) obtained an increase of 4% compared to the corpus that has not been optimized. As for Neural Machine Translation, an increase of 8.15% was obtained.

Article History

Submitted: 30 Januari 2026

Accepted: 2 Februari 2026

Published: 3 Februari 2026

Key Words

Neural machine translation, attention mechanism, transformer architecture, statistical machine translation, BLEU.

Abstrak

Bahasa merupakan alat yang digunakan masyarakat untuk berkomunikasi. Indonesia memiliki bahasa Indonesia sebagai bahasa nasional. Indonesia sendiri memiliki banyak sekali daerah serta bahasa daerah yang menyertainya, salah satunya adalah bahasa Melayu Ketapang yang biasa digunakan oleh masyarakat Kabupaten Ketapang. Bahasa Melayu Ketapang terbilang sudah hampir punah sehingga membutuhkan perkembangan teknologi sebagai alat untuk menjaga kelestariannya. Salah satu teknologi yang bisa membantu melestarikan bahasa daerah adalah dengan membuat suatu mesin penerjemah. Pada berbagai penelitian yang telah dilakukan oleh peneliti di Indonesia, penerapan mesin penerjemah untuk bahasa daerah yang dilakukan masih berbasis statistik (SMT), sedangkan dalam beberapa tahun terakhir mesin penerjemah jaringan saraf (NMT) telah menjadi metode baru dalam praktik mesin penerjemah. *Neural Machine Translation* yang digunakan pada penelitian ini memggunakan mekanisme *attention* dan arsitektur *transformer* dengan menggunakan korpus paralel bahasa Indonesia dan bahasa Melayu Ketapang sebanyak 5000 baris kalimat. Penelitian ini bertujuan untuk mengetahui hasil nilai akurasi terjemahan pada *Neural Machine Translation* dengan menggunakan korpus paralel yang sama dengan penelitian sebelumnya, yaitu penelitian yang dilakukan oleh Dinar (2020) menggunakan mesin penerjemah statistik. Hasil pengujian otomatis untuk mesin penerjemah jaringan statistik oleh BLEU dengan korpus yang telah dilakukan optimisasi oleh Dinar (2020) memperoleh peningkatan sebesar 4% dibanding dengan

Sejarah Artikel

Submitted: 30 Januari 2026

Accepted: 2 Februari 2026

Published: 3 Februari 2026

Kata Kunci

Neural machine translation, mekanisme attention, arsitektur transformer, statistical meachine translation, BLEU

korpus yanng belum dioptimisasi. Sedangkan untuk *Neural Machine Translation*, memperoleh peningkatan sebesar 8,15%

PENDAHULUAN

Bahasa merupakan alat komunikasi untuk menyampaikan pikiran dan perasaan antar umat manusia. Menurut Kridalaksana dan Djoko Kentjono (2014) Bahasa adalah sistem lambang bunyi yang arbitrer yang digunakan oleh para anggota kelompok sosial untuk bekerja sama, berkomunikasi, dan mengidentifikasi diri. Bahasa digunakan oleh umat manusia untuk saling berkomunikasi di kehidupan sehari-hari. Menurut catatan Grimes (2000) bahasa yang ada didunia sebanyak 6.809 bahasa. Sedangkan di Indonesia memiliki 718 bahasa ibu menurut Kemendikbud (2020). Salah satu Bahasa ibu di Kalimantan Barat ialah Bahasa Melayu Ketapang.

Bahasa Melayu Ketapang merupakan Bahasa yang mayoritas digunakan di Kabupaten Ketapang. Bahasa Melayu Ketapang memiliki perbedaan dalam pengucapan, kosakata dan bentuk katanya. Akan tetapi saat ini Bahasa Melayu Ketapang mulai mengalami kepunahan (Sulissusiawan, 1998) karena masyarakat semakin jarang berkomunikasi menggunakan Bahasa ini. Pada saat ini banyak hal yang bisa dilakukan untuk melestarikan sebuah bahasa, dengan perkembangan teknologi pada saat ini, menciptakan sebuah mesin penerjemah berbasis Bahasa Melayu Ketapang bukanlah hal yang mustahil. Sudah banyak peneliti yang mengembangkan sebuah mesin penerjemah yang menggunakan Bahasa daerah yang dikuasai oleh peneliti itu sendiri. Salah satunya adalah *Neural Machine Translation* yang sedang populer belakangan ini semenjak Google pada tahun 2016 memutuskan untuk mengganti jenis mesin penerjemahannya dari mesin penerjemah statistik ke *Neural Machine Translation* (Turovsky, 2016).

Mesin penerjemah memiliki banyak jenis, salah satunya adalah *Statistical machine translation* atau dikenal dangan SMT adalah sebuah *machine translation* yang menggunakan pendekatan statistik. Pendekatan statistik yang digunakan adalah konsep probabilitas. Setiap pasangan kalimat (S,T) akan diberikan sebuah P(T|S) yang diinterpretasikan sebagai distribusi probabilitas dimana sebuah penerjemah akan menghasilkan T dalam bahasa sasaran ketika diberikan S dalam bahasa sumber(Tanuwijaya, 2009). Mesin Penerjemah Statistik (*Statistical Machine Translation*) merupakan sebuah pendekatan mesin penerjemah dengan hasil terjemahan yang dihasilkan atas dasar model statistik yang parameter-parameternya diambil dari hasil analisis korpus paralel.

Tidak hanya SMT, mesin penerjemah juga ada yang lainnya yaitu mesin penerjemah jaringan saraf (*Neural Machine Translation)* adalah sebuah pendekatan terjemahan mesin yang menggunakan jaringan saraf tiruan besar untuk memprediksi dan menghasilkan terjemahan bahasa. *Neural* merujuk pada cara kerja komputer yang terinspirasi dari cara kerja otak manusia, menggunakan neuron yang saling terhubung untuk saling mengirimkan informasi dan mempelajari pola data melalui jaringan saraf tiruan yang saling terhubung. Mesin penerjemah jaringan saraf tiruan (NMT) dibangun dari *framework* model *sequence to sequence* yang memiliki hal utama adalah dua jaringan saraf tiruan dengan arsitektur *encoder* dan *decoder* yang membaca kata sumber satu persatu untuk mendapatkan representasi vektor dari dimensi tetap (*encoder*), kemudian meng-kondisikan *input* dan mengekstrak kata target satu per satu (*decoder*)(Gunawan, 2021). Mesin penerjemah jaringan saraf atau *neural machine translation* memiliki beberapa arsitektur, salah satu arsitektur yang memiliki tingkat akurasi lebih baik dan akan digunakan pada penelitian ini adalah arsitektur *transformer*.

Pada mesin penerjemah sendiri terdapat komponen penting, yaitu salah satunya adalah korpus. Korpus adalah kumpulan teks alami, baik bahasa lisan maupun bahasa tulis, yang disusun secara sistematis. Dikatakan alami karena teks yang dikumpulkan merupakan teks

yang diproduksi dan digunakan secara wajar dan tidak dibuat-buat (Hunston, 2002). Pengaruh korpus pada mesin terjemahan sangat bergantung dari kuantitas dan kualitas korpus. Semakin baik kualitas korpus dan semakin banyak kuantitas korpus maka akan meningkatkan kualitas mesin penerjemahan.

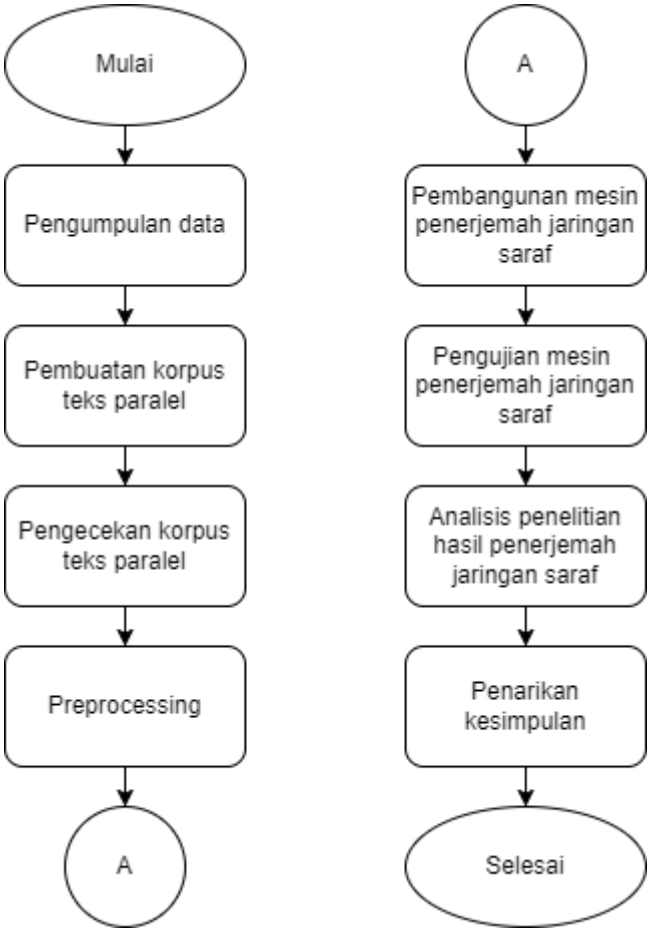
Metode yang akan digunakan pada penelitian ini adalah metode optimasi korpus. Dimana metode ini sudah memiliki berbagai cara yang telah dikembangkan oleh peneliti sebelumnya di antaranya adalah penelitian dengan penggunaan Strategi Memperbaiki Kualitas Korpus Untuk Meningkatkan Kualitas Mesin Penerjemah Statistik dengan strategi yang digunakan pada penelitian ini, kualitas terjemahan sistem MPS dapat ditingkatkan sebesar 7.84% dengan mengeliminasi sebanyak kurang lebih 17% kalimat oleh Bijaksana dan Sujaini (2014), Optimasi korpus memfilter kalimat korpus dapat meningkatkan akurasi mesin penerjemah bahasa daerah yang dilakukan oleh Sujaini (2018), dan memperbaiki kualitas korpus dengan mengeliminasi kalimat-kalimat tidak berkualitas dapat meningkatkan akurasi mesin penerjemah bahasa Indonesia-Jawa Krama yang dilakukan oleh Dio (2018).

Salah satu penelitian yang dilakukan oleh Siti Tirta Dinar (2020) yang membahas penggunaan mesin penerjemah statistik dengan metode optimasi korpus dengan hasil rata-rata nilai BLEU (*Billingual Evaluation Understudy*) sebesar 63,70% yang dimana sebelum di optimasi nilai rata-ratanya hanya mencapai 61,25%. Oleh karena itu pada penelitian ini akan diuji coba dengan meng-implemestansikan data dan metode yang sama yaitu metode optimasi korpus untuk melihat nilai akurasi pada *Neural Machine Translation* dengan meningkatkan kualitas korpus, dimana metode optimasi korpus yang dilakukan dengan cara memfilter kalimat-kalimat yang tidak berkualitas pada korpus dan memperbaiki korpus yang tidak berkualitas dengan studi kasus bahasa Indonesia ke bahasa Melayu Ketapang.

METODOLOGI PENELITIAN

Metode Penelitian

Metode pada penelitian ini memiliki 8 tahapan yang dimulai dari proses pengumpulan data hingga proses penarikan kesimpulan. Berikut ini adalah diagram dan rincian proses yang akan dilakukan selama penelitian.



Gambar 2. 1 Diagram Alir Metodologi Penelitian

Terlihat pada Gambar 2.1 dimana gambar tersebut mengilustrasikan tahapan-tahapan yang akan dilakukan pada penelitian ini.

Pengumpulan data

Proses pengumpulan data merupakan proses mencari dan mengumpulkan data-data yang akan digunakan untuk penelitian. Pencarian diliputi dengan mencari seorang narasumber yaitu ahli bahasa untuk menjadi penilai hasil terjemahan dari mesin penerjemah. Setelah mendapatkan seorang narasumber dalam penelitian ini maka selanjutnya yang dilakukan adalah mencari korpus penelitian yang telah dilakukan dan dikumpulkan sebelumnya oleh Siti Tirta Dinar (2020). Korpus yang telah disusun didapatkan dari buku novel terjemahan Harry Potter *and the Order of Phoenix*, buku novel Laskar Pelangi dan hasil penelitian Rahayu (2016) menggunakan buku Chairul Tanjung Si Anak Singkong. Data yang diperoleh tersebut selanjutnya diolah menjadi korpus teks paralel bahasa Indonesia dan bahasa Melayu Ketapang. Jumlah korpus paralel yaitu 5000 pasang kalimat.

HASIL DAN ANALISIS

Hasil Penelitian

Hasil dari penelitian yang dilakukan pada bab sebelumnya dapat dilihat di sini. Pembuatan korpus teks paralel, pembuatan *Neural Machine Translation* tiruan, penggunaan *Neural Machine Translation* tiruan, dan pengujian atau penilaian hasil penerjemahan mesin translasi adalah beberapa hasil dari penelitian ini.

Hasil Pembuatan Korpus Teks Paralel

Berdasarkan hasil pengumpulan data sebelumnya, analisis data dibuat dengan menyesuaikan data untuk memenuhi kebutuhan penelitian. Hasil analisis terdiri dari dokumen korpus dari penelitian Dinar (2020) dengan buku novel terjemahan *Harry Potter and the Order of Phoenix*, buku Laskar Pelangi, dan penelitian Rahayu (2016) dengan buku Chairul Tanjung Si Anak Singkong yang berbahasa Indonesia dan diterjemahkan ke dalam bahasa Inggris. Data tersebut disusun menjadi korpus paralel yang masing-masing memiliki 5000 kalimat.

Membangun *Neural Machine Translation*

Memanfaatkan layanan produk *Google*, pembuatan *Neural Machine Translation* dapat dilakukan secara online. Perangkat lunak yang diperlukan untuk pekerjaan ini adalah *intertext*, yang berfungsi sebagai editor teks a*lignment* kalimat korpus dan telah disiapkan untuk proses pelatihan di masa mendatang. Berikut ini adalah daftar produk layanan *Google* yang akan digunakan untuk membantu membangun *Neural Machine Translation*:

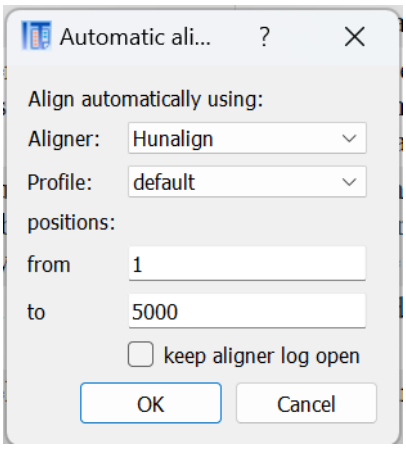
1. *Browser Chrome* untuk mengakses layanan produk *google*
2. *Google Drive* untuk penyimpanan *online* data korpus
3. *Google Collaboratory* untuk menjalankan program
4. *Framework TensorFlow* untuk mengimplementasi model *neural network.*

Implementasi *Neural Machine Translation*

Untuk mengimplementasikan *Neural Machine Translation* tiruan berdasarkan hal yang telah disiapkan sebelumnya yaitu dengan cara *online*. Beberapa tahapan yang akan dilewati yaitu mempersiapkan korpus dengan *intertext*, menyimpan korpus pada *google drive*, mengakses program NMT pada *tensorflow*, menjalankan program NMT dengan *google colaboratory*. Berikut tahapan yang dilakukan.

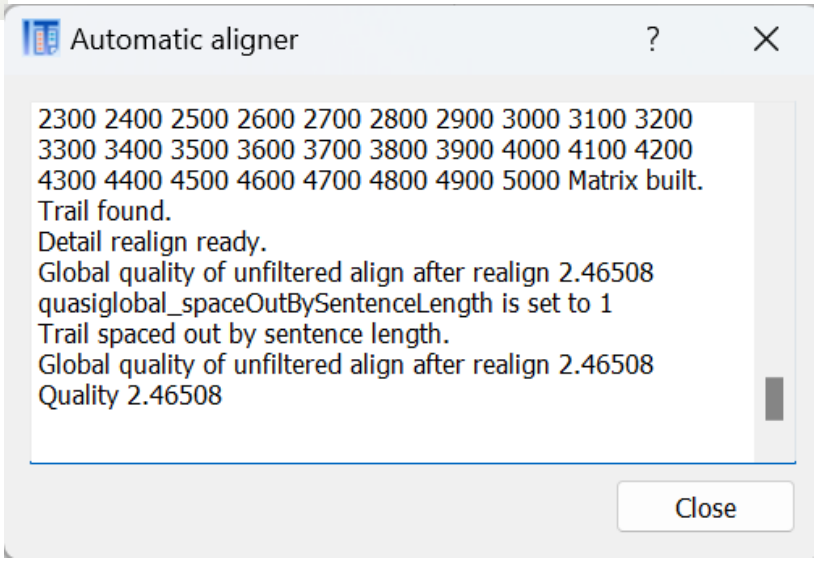
Mempersiapkan Korpus dengan *Intertext*

Pada tahap ini masing-masing korpus paralel berjumlah 5000 kalimat bahasa Indonesia dan bahasa Melayu Ketapang akan di cek keselarasan kalimatnya. Data yang telah dipersiapkan akan disimpan dengan format yang sama yaitu format *.txt* atau teks dokumen, kemudian di cek menggunakan aplikasi *Intertext*. Aplikasi *Intertext* membantu mengecek *alignment* secara otomatis sehingga tidak perlu melakukan penyelarasan secara manual dikarenakan akan meghabiskan lebih banyak waktu.



Gambar 3. 1 Tampilan awal aplikasi *Intertext*

Pada **Gambar 3.1** merupakan tampilan awal sebelum pengecekan otomatis. Dapat dilihat pada gambar, dapat ditentukan jumlah baris yang akan dicek. Berdasarkan data yang telah disediakan maka baris yang akan di cek berjumlah 5000 baris kalimat. Setelah itu tekan tombol OK data tersebut akan diproses.



Gambar 3. 2 Proses penyelarasan oleh aplikasi *Intertext*

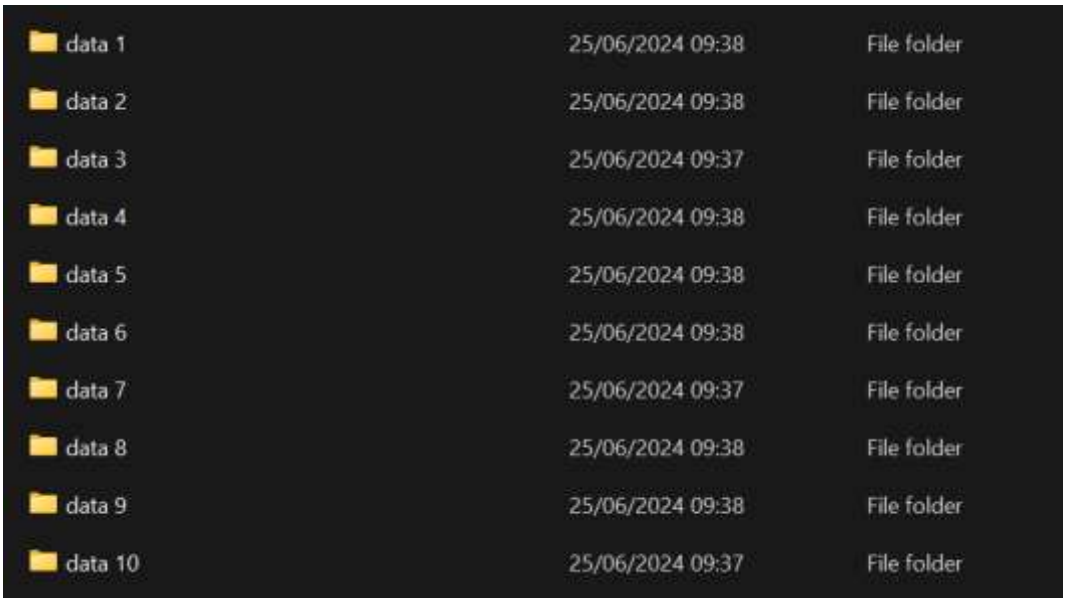
Dapat dilihat pada **Gambar 3.2** proses penyelarasan hanya butuh beberapa detik saja sehingga akan lebih efektif dan efisien dibandingkan dengan pengecekan manual satu-persatu.

	M	id sblm	ml sblm	S
1	☆	Saudara-saudaraku, pimpinan dan seluruh karyawan perusahaan di bawah naungan ct corp yang saya cintai dan banggakan.	Saudare-saudareku, pimpenan dan seluroh karyawan perusahaan tang bawah naungan ct corp yang saye cintai dan banggeek.	
2	☆	Buku yang anda pegang ini adalah sebuah catatan sejarah yang berisi perjalanan hidup saya.	Buku yang kau pegang nin adalah sebutik catatan sejarah yang berisik perjalanan idop saye.	
3	☆	Melalui buku ini, saya ingin berbagi kisah dan cerita dengan anda semua tentang liku-liku, pengalaman, serta makna dan nilai-nilai kehidupan yang saya pegang mulai dari kecil sampai hari ini menjadi pemimpin anda semua.	Melaluek buku nin, saye ingin bebagi kisah dan cerite dengan kau semue tentang liku-liku, pengalaman, serte makne dan nilai-nilai keidopan yang saye pegang mulai dari kecik sampe hari nin menjadi pemimpen kau semue.	
4	☆	Dengan memahami sejarah hidup saya ini, saya percaya anda akan menjadi bagian yang tidak terpisahkan dari mimpi seorang chairul tanjung.	Dengan memahami sejarah idop saye nin saye percaye kau akan menjadi bagian yang endak terpisahek dari mimpi seorang chairul tanjung.	
	☆	Mari kita melangkah dan bekerja bersama membesarkan seluruh	Mari kite melangkah dan bekerje same membesakek seluroh	

Gambar 3. 3 Hasil penyelarasan otomatis oleh aplikasi *Intertext*

Berikut **Gambar 3.3** Yang menunjukkan hasil dari proses penyelarasan secara otomatis menggunakan *Intertext*. Setelah itu data korpus yang telah diselaraskan akan disimpan ulang dengan menggunakan nama yang berbeda dari sebelumnya.

Korpus yang telah didapatkan akan dibagi secara manual untuk pengujian otomatis oleh BLEU. Korpus akan dibagi berdasarkan *fold* untuk pengujian *K-fold cross validation*, oleh karena itu korpus akan dibagi menjadi 10 *fold* untuk penerjemahan bahasa Indonesia ke bahasa Melayu Ketapang. 10 *fold* tersebut terdiri atas 1 fold sebagai data uji dan 9 fold lainnya akan menjadi data pelatihan.



Gambar 3. 4 Hasil data korpus yang telah dipilah sesuai dengan *k-fold cross validation*

Setiap fold yang telah dibagi akan dinamakan data 1 yang berisi 3 text file yaitu korpus Bahasa Indonesia, Bahasa Melayu Ketapang dan korpus paralel Bahasa Indonesia-Bahasa Melayu Ketapang. Pada folder data 1 berisi kalimat 1-500 untuk Bahasa Indonesia, Bahasa Melayu Ketapang sebagai data uji dan 501-5000 untuk korpus paralel Bahasa Indonesia-Bahasa Melayu Ketapang sebagai data training. Begitupula folder data 2 berisi 3 text file yaitu kalimat 501-1000 untuk Bahasa Indonesia, Bahasa Melayu Ketapang sebagai data uji dan 1-500 dan 1001-5000 untuk korpus paralel Bahasa Indonesia-Bahasa Melayu Ketapang sebagai data training dan begitu seterusnya untuk folder data selanjutnya.

Menyimpan Korpus pada *Google Drive*

Korpus paralel yang telah disiapkan sebelumnya akan diunggah ke *Google Drive*. Pada penelitian ini akan menggunakan akun *Google Drive* dengan penyimpanan tanpa batas. Namun, jika menggunakan akun *Google Drive* biasa dengan penyimpanan 15 GB nantinya akan kehabisan ruang penyimpanan karena penelitian ini akan melakukan banyak pembuatan model NMT yang disimpan di *Google Drive*

Mengakses program NMT pada *Tersorflow*

Untuk menerapkan NMT dalam penelitian ini, menggunakan *framework deep learning tensorflow* yang sudah tersedia secara *open-source* pada link berikut: *https://www.tensorflow.org/text/tutorials/nmt_with_attention*. *TensorFlow* juga menyediakan fitur untuk pengembangan bahasa natural alami (NLP), seperti penghasil dan klasifikasi teks. Penelitian ini menggunakan layanan penghasil teks, yaitu penerjemahan teks dengan model *sequence-to-sequence*, untuk membangun NMT secara *open-source*. Siapa pun yang tertarik dapat bereksperimen dengannya.

Menjalankan Program NMT di *Google Colab*

Salah satu hal yang perlu diperhatikan saat mengimplementasikan NMT adalah proses pembangunan model NMT. Menurut penelitian sebelumnya, ada banyak alat yang dapat digunakan untuk membangun model NMT, seperti *OpenNMT* dan *MarianNMT*. Namun, dengan menggunakan alat ini akan dilakukan secara *offline*, yang memungkinkan untuk menggunakan sumber daya perangkat komputer lokal. Secara umum, untuk membangun NMT dengan alat ini harus memiliki sumber daya perangkat komputer dengan spesifikasi khusus.

Untuk melakukan eksperimen mengenai NMT, layanan *Google Colaboratory* dapat diakses secara gratis melalui *browser web* dan tidak membutuhkan sumber daya perangkat komputer yang kuat.

Pelatihan dan pengujian model NMT yang telah dilakukan pada penelitian ini diakses terakhir di bulan Maret 2021 adalah contoh dari pembaharuan rutin yang dilakukan oleh *framework deep learning tensorflow* yang menyediakan layanan implementasi NMT. Karena programnya open-source, penulis dapat mengubah versi awalnya untuk memenuhi kebutuhan mereka.

Persiapan Data Latih

Pada tahap ini, data latih yang telah diproses akan diunggah pada *google drive* untuk dilakukan proses lebih lanjut yaitu memuat data latih, membaca data latih, *preprocessing* dan *tokenizing*, perubahan data latih menjadi data *tensor* dan pengaturan *parameter* dan *hyperparameter*

Memuat Data Latih

Langkah pertama adalah memuat data latih dengan memasang *google drive* pada *google colaboratory* agar dapat mengakses data latih yang ada pada *google drive*. Setelah terpasang, kemudian pindah ke *folder* yang telah menyimpan data latih yang telah diunggah. Selanjutnya melakukan pengecekan apakah *notebook* sudah aktif pada *folder* tersebut. Kode program dan *output*nya dapat dilihat pada **Gambar 3.5**

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Gambar 3. 5 Menyambungkan *google drive* ke *google colaboratory*

```
%cd /content/drive/MyDrive/Colab Notebooks/setelah optimasi/data 6/
```

```
/content/drive/MyDrive/Colab Notebooks/setelah optimasi/data 6
```

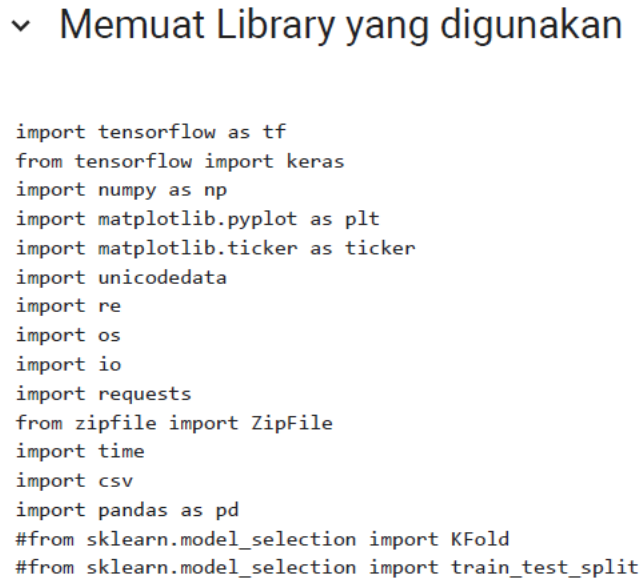
Cek notebook yang digunakan

```
!pwd
```

```
/content/drive/MyDrive/Colab Notebooks/setelah optimasi/data 6
```

Gambar 3. 6 Mengecek *folder* yang akan digunakan

Sebelum data dimuat yang harus dilakukan adalah mengimport *library* yang dibutuhkan untuk membangun mesin terjemahan jaringan saraf pada *framework tersorflow*. kode program dapat diliat pada **Gambar 3.7** berikut :



Gambar 3. 7 Kode program untuk *mengimport library* yang akan digunakan pada mesin penerjemah

Setelah mengimport library yang akan digunakan, maka selanjutnya adalah memuat data pelatihan yang sebelumnya sudah diunggah pada google drive yang telah tersambung dengan google colaboratory. Memuat data latih dilakukan berdasarkan variabel nama file. Kode program dapat dilihat pada **Gambar 3.8** berikut:

```
#Read the data
text_file = '/content/drive/MyDrive/Colab Notebooks/setelah optimasi/data 6/training 6.txt'
lines_raw= pd.read_table(text_file,names=['input','target','label'])
lines_raw.sample(15)
```

Gambar 3. 8 Membaca data latih yang telah di import dari *google drive*

Output yang dihasilkan menujukkan 15 baris kalimat sebagai *sample* dari 4500 baris kalimat dari hasil menggunakan data *frame* dari *library* pandas yang terdiri dari *input*, *target* dan label kalimat. *Output*-nya dapat dilihat pada **Gambar 3.9.**

	input	target	label
2051	kemudian saya berpikir lagi	kemudian saye berpiker agik	id-ml
4419	walaupun sudah berusaha sekuat tenaga dan meng...	walaupon udah berusahe sekuat tenage dan menge...	id-ml
811	hp kamu mirip mas-mas rul, sindir mereka	hp kau rul, mirip mas-mas sindir sidak	id-ml
68	setengah porsi nasi, ditambah sayur, tempe ata...	setengah porsi nasik, ditambah sayuk, tempe at...	id-ml
1033	saat ini saya sudah sangat jarang berkomunikas...	sekarang nin saye udah sangat jarang berkomuni...	id-ml
1380	hasil diskusi internal grup fisika, kami menet...	hasil diskusi internal grup fisika, kami menet...	id-ml
242	rapat di tingkat dewan mahasiswa segera dilaku...	rapat tang tingkat dewan mahasiswa segere dila...	id-ml
239	sebagai penghargaan, saya dan para juara lainn...	sebagai penghargaan, aku dan pare juara laenny...	id-ml
3843	setelah ungkut ungkut berlalu hinggaplah kawan...	setelah ungkut ungkut berlalu hinggeplah kawan...	id-ml
2797	orang-orang cebol digiring bonenga ke muara su...	orang orang cebol digiring bonenga ke muare su...	id-ml
2293	taufik merupakan binaan pbsi pengda jawa barat...	taufik merupekan binaan pbsi pengda jawe barat...	id-ml
137	bayangkan, teman-teman mahasiswa menghemat rp ...	bayangkan, kawan-kawan mahasiswa menghemat rp ...	id-ml
1558	di meja makan menjelang tidur dan saat sarapan...	tang meje makan menjelang tidok dan saat sarap...	id-ml
4099	ada keindahan yang unik dalam interaksi masing...	ade kejangakan yang unik dalam interaksi masin...	id-ml
456	misalnya 7 persen dari total penduduk 161,5 ju...	misalnye 7 persen dari total pendudok 161,5 ju...	id-ml

Gambar 3. 9 output dari data latih yang telah di import dari *google drive*

Membaca Data Latih

Setelah data dimuat dan ditampilkan *sample*-nya, selanjutnya adalah membaca data latih dari baris pertama hingga baris terakhir dalam daftar *string* menggunakan *split* karakter pada *file* untuk mendapatkan baris pertama hingga akhir yang berisi karakter dari tabulator, lalu disimpan menjadi *array*. Kode program dapat di lihat pada **Gambar 3.10** berikut :

```
#lines = lines.decode('utf-8')
text_file = '/content/drive/MyDrive/Colab Notebooks/setelah optimasi/data 6/training 6.txt'
with open(text_file) as f:
    lines = f.read()[:-1]

raw_data = []

for line in lines.split('\n'):
    raw_data.append(line.split('\t'))

print(raw_data[-5:])
print(np.shape(raw_data))

# The last element is empty, so omit it
raw_data = raw_data[:-1]
```

Gambar 3. 10 Kode Program untuk membaca data latih yang telah di-*import* dari *google drive*

Pre-Processing Data Latih

Setelah data latih dibaca, selanjutnya data latih akan diubah menjadi format yang dapat dengan mudah dibaca oleh komputer agar efektif dan efisien. Proses ini dikenal dengan fungsi *unicode_to_ascii* yaitu mengubah *string ascii* menjadi karakter *unicode*, dan fungsi *preprocessing* mengubah semua kalimat menjadi huruf kecil dengan spasi di antara huruf dan tanda baca, kecuali huruf "a-z", "A-Z", dan "tanda baca". Kemudian akan disimpan *string* tersebut menggunakan *list zip* dengan menambahkan *token* “<start>” untuk awal kalimat dan

token "<end>" untuk akhir kalimat. Sehingga model dapat memahami kapan memulai dan mengakhiri prediksi. Kode program dapat ditunjukkan pada **Gambar 3.11** dan **Gambar 3.12** berikut:

```
# Mode can be either 'train' or 'infer'
# Set to 'infer' will skip the training
# URL = 'http://www.manythings.org/anki/fra-eng.zip'

MODE = 'train'
text_file = '/content/drive/MyDrive/Colab Notebooks/setelah optimasi/data 6/training 6.txt'
with open(text_file) as f:
    lines = f.read()[:-1]

def unicode_to_ascii(s):
    return ''.join(
        c for c in unicodedata.normalize('NFD', s)
        if unicodedata.category(c) != 'Mn')

def normalize_string(s):
    s = unicode_to_ascii(s.lower().strip())
    s = re.sub(r'([!.?])', r' \1', s)
    s = re.sub(r'[^a-zA-Z.!?]+', r' ', s)
    s = re.sub(r'\s+', r' ', s)
    return s
```

Gambar 3. 11 *Kode program untuk pre-processing data*

```
raw_data_id, raw_data_ml,raw_data_attr  = list(zip(*raw_data))
raw_data_id = [normalize_string(data) for data in raw_data_id]
raw_data_ml_in = ['<start> ' + normalize_string(data) for data in raw_data_ml]
raw_data_ml_out = [normalize_string(data) + ' <end>' for data in raw_data_ml]
```

Gambar 3. 12 Sambungan kode program untuk *pre-processing* data

Tokenizing

Selanjutnya adalah tahap *tokenizing*. Menyesuaikan urutan angka dari kata-kata ke dalam urutan numerik dan memberikan angka nol hingga ukuran kalimat terbesar dalam kosakata adalah fungsi *tokenizing*. Pastikan setiap rangkaian sama panjang agar dapat dilatih pada model setelah semua kalimat dalam teks diubah menjadi rangkaian berikutnya. *Padding* adalah proses mengubah setiap rangkaian menjadi panjang yang sama. Selain itu, panjang maksimum urutan dapat dipotong dengan *padding* ke panjang yang tepat. Selain itu, dapat digunakan sebagai *input/output* kosa kata dan tipe *padding ('pre'/'post'-default: post). Tokenizer.fit* pada *text* adalah jenis baris data yang dimaksudkan untuk disesuaikan dengan kelas teks *tokenizer* yang menggantikan kosakata internal yang didasarkan pada frekuensi kata. Hasil *tokenizing* lalu akan disimpan pada *pipeline dataset tensor slice* dalam tipe data *tensor*, Dengan cara ini, elemen sumber data dapat digunakan selama perulangan. di mana ukuran *batch_size* untuk membuat *pipeline dataset tensor* ini adalah 64. Kode program dapat dilihat pada **Gambar 3.13** berikut:

```
## Tokenization

id_tokenizer = tf.keras.preprocessing.text.Tokenizer(filters='')
id_tokenizer.fit_on_texts(raw_data_id)
data_id = id_tokenizer.texts_to_sequences(raw_data_id)
data_id = tf.keras.preprocessing.sequence.pad_sequences(data_id,
                                                        padding='post')

ml_tokenizer = tf.keras.preprocessing.text.Tokenizer(filters='')
ml_tokenizer.fit_on_texts(raw_data_ml_in)
ml_tokenizer.fit_on_texts(raw_data_ml_out)
data_ml_in = ml_tokenizer.texts_to_sequences(raw_data_ml_in)
data_ml_in = tf.keras.preprocessing.sequence.pad_sequences(data_ml_in,
                                                           padding='post')

data_ml_out = ml_tokenizer.texts_to_sequences(raw_data_ml_out)
data_ml_out = tf.keras.preprocessing.sequence.pad_sequences(data_ml_out,
                                                            padding='post')

## Create tf.data.Dataset object

BATCH_SIZE = 64
dataset = tf.data.Dataset.from_tensor_slices(
    (data_id, data_ml_in, data_ml_out))
dataset = dataset.shuffle(len(data_id)).batch(BATCH_SIZE)
```

Gambar 3. 13 Kode program untuk proses *tokenizing*

Positional Encoding

Positional encoding menggambarkan input sebagai kumpulan *vektor* yang diatur. Pengkodean posisi memberikan informasi awal tentang lokasi *token* dalam kalimat ke lapisan model pada proses selanjutnya. Dengan demikian, *token* akan lebih dekat satu sama lain berdasarkan kesamaan makna dan posisi setelah pengkodean posisi ditambahkan. Pada *output* dapat dilihat hasil *pes* atau *positional encoding sequence shape* pada posisi *token*-nya. Kode program untuk *positional encoding* dapar dilihat pada **Gambar 3.14** berikut:

```
## Create the Positional Embedding
""" Compute positional encoding for a particular position
    Args:
        pos: position of a token in the sequence
        model_size: depth size of the model
    Returns:
        The positional encoding for the given token
"""

def positional_encoding(pos, model_size):
    PE = np.zeros((1, model_size))
    for i in range(model_size):
        if i % 2 == 0:
            PE[:, i] = np.sin(pos / 10000 ** (i / model_size))
        else:
            PE[:, i] = np.cos(pos / 10000 ** ((i - 1) / model_size))
    return PE

max_length = max(len(data_id[0]), len(data_ml_in[0]))
MODEL_SIZE = 512

pes = []
for i in range(max_length):
    pes.append(positional_encoding(i, MODEL_SIZE))

pes = np.concatenate(pes, axis=0)
pes = tf.constant(pes, dtype=tf.float32)


print(pes.shape)
print(data_id.shape)
print(data_ml_in.shape)
```

Gambar 3. 14 Kode program pada tahap *positional encoding*

```
(68, 512)
(4499, 56)
(4499, 68)
```

Gambar 3. 15 Output dari tahap *positional encoding*

***Masking* atau Penyamaran**

Proses *masking* atau penyamaran, berfungsi untuk menutup semua *pad token* yang ada dalam kumpulan urutan. Ini dilakukan untuk mencegah model menganggap *pad* sebagai *input*, dan menunjukkan nilai 1 jika *token* hadir pada posisinya, dan nilai 0 jika sebaliknya. Pada proses *masking* juga memiliki fungsi *create look ahead mask* untuk memorediksi *token ketiga*,

dimana yang nantinya akan digunakan hanya *token* pertama dan kedua Kode program dan *output* pada proses *masking* dapat dilihat pada **Gambar 3.16** berikut:

```
def create_padding_mask(seq):
  seq = tf.cast(tf.math.equal(seq, 0), tf.float32)

  return seq[:, tf.newaxis, tf.newaxis, :]  #(batch_size, 1, 1, seq_len)

x = tf.constant([[7, 6, 0, 0, 1], [1, 2, 3, 0, 0], [0, 0, 0, 4, 5]])
create_padding_mask(x)
```

Gambar 3. 16 Kode program pada tahap *masking*

```
<tf.Tensor: shape=(3, 1, 1, 5), dtype=float32, numpy=
array([[[[0., 0., 1., 1., 0.]]],


       [[[0., 0., 0., 1., 1.]]],


       [[[1., 1., 1., 0., 0.]]]], dtype=float32)>
```

Gambar 3. 17 Output pada tahap *masking*

Encoder

Encoder adalah lapisan pembuat kode, pada arsitektur *transformer* setiap lapisan terdiri dari sub-lapisan yang akan dijabarkan sebagai berikut.

Proses Data Latih pada Lapisan *Scale Dot Product Attention*

Transformer memerlukan tiga input: Q (*query*), K (*key*), dan V (*value*). Dengan menggunakan faktor akar kuadrat dari kedalaman, perhatian produk titik diskalakan. Ini dilakukan karena titik bertambah besar produk akan meningkatkan fungsi *softmax* dengan gradien kecil yang menghasilkan *softmax* yang keras. *Mask* dikalikan dengan *c-1e9*, yang mendekati *infinity* negatif. Ini dilakukan karena *mask* ditambahkan dengan perkalian matriks skala Q dan K dan diterapkan sebelum *softmax*. Tujuannya adalah untuk menghilangkan sel-sel ini, dan keluaran memiliki masukan negatif besar ke *softmax* yang mendekati nol. Kode program untuk proses data latih pada lapisan *scale dot product attention* dapat dilihat pada **Gambar 3.18** berikut.

```
def scaled_dot_product_attention(q, k, v, mask):

  matmul_qk = tf.matmul(q, k, transpose_b=True)  #(..., seq_len_q, seq_len_k)

  # scale matmul_qk
  dk = tf.cast(tf.shape(k)[-1], tf.float32)
  scaled_attention_logits = matmul_qk / tf.math.sqrt(dk)

  # add the mask to the scaled tensor.
  if mask is not None:
    scaled_attention_logits += (mask * -1e9)

  # softmax is normalized on the last axis (seq_len_k) so that the scores
  # add up to 1.
  attention_weights = tf.nn.softmax(scaled_attention_logits, axis=-1)  # (..., seq_len_q, seq_len_k)

  output = tf.matmul(attention_weights, v)  # (..., seq_len_q, depth_v)

  return output, attention_weights
```

Gambar 3. 18 Kode program pada tahap *scale-dot attention*

Nilai K setelah normalisasi *softmax* menunjukkan jumlah kepentingan yang diberikan kepada Q. Hasilnya mewakili perkalian bobot perhatian dan *vektor* V memastikan bahwa *token* yang ingin difokuskan disimpan apa adanya dan *token* yang tidak relevan dihilangkan.

```
def print_out(q, k, v):
  temp_out, temp_attn = scaled_dot_product_attention(
      q, k, v, None)
  print('Attention weights are:')
  print(temp_attn)
  print('Output is:')
  print(temp_out)

np.set_printoptions(suppress=True)

temp_k = tf.constant([[10, 0, 0],
                      [0, 10, 0],
                      [0, 0, 10],
                      [0, 0, 10]], dtype=tf.float32)  # (4, 3)

temp_v = tf.constant([[1, 0],
                      [10, 0],
                      [100, 5],
                      [1000, 6]], dtype=tf.float32)  # (4, 2)

# This `query` aligns with the second `key`,
# so the second `value` is returned.
temp_q = tf.constant([[0, 10, 0]], dtype=tf.float32)  # (1, 3)
print_out(temp_q, temp_k, temp_v)

# This query aligns with a repeated key (third and fourth),
# so all associated values get averaged.
temp_q = tf.constant([[0, 0, 10]], dtype=tf.float32)  # (1, 3)
print_out(temp_q, temp_k, temp_v)

# This query aligns equally with the first and second key,
# so their values get averaged.
temp_q = tf.constant([[10, 10, 0]], dtype=tf.float32)  # (1, 3)
print_out(temp_q, temp_k, temp_v)

temp_q = tf.constant([[0, 0, 10],
                      [0, 10, 0],
                      [10, 10, 0]], dtype=tf.float32)  # (3, 3)
print_out(temp_q, temp_k, temp_v)
```

Gambar 3. 19 Residual lapisan normalisasi pada *scale-dot attention*

Berikut *output* dari kode program dari **Gambar 3.19** diatas.

```
Attention weights are:
tf.Tensor([[0. 1. 0. 0.]], shape=(1, 4), dtype=float32)
Output is:
tf.Tensor([[10.  0.]], shape=(1, 2), dtype=float32)
Attention weights are:
tf.Tensor([[0.  0.  0.5 0.5]], shape=(1, 4), dtype=float32)
Output is:
tf.Tensor([[550.    5.5]], shape=(1, 2), dtype=float32)
Attention weights are:
tf.Tensor([[0.5 0.5 0.  0. ]], shape=(1, 4), dtype=float32)
Output is:
tf.Tensor([[5.5 0. ]], shape=(1, 2), dtype=float32)
Attention weights are:
tf.Tensor(
[[0.  0.  0.5 0.5]
 [0.  1.  0.  0. ]
 [0.5 0.5 0.  0. ]], shape=(3, 4), dtype=float32)
Output is:
tf.Tensor(
[[550.    5.5]
 [ 10.    0. ]
 [  5.5   0. ]], shape=(3, 2), dtype=float32)
```

Gambar 3. 20 Output pada tahap *scale-dot attention*

Proses Data Latih pada *Multi-head Attention*

Setiap blok *multi-head attention* menerima tiga masukan: Q(*query*), K(*key*), dan V(*value*). Sebelum fungsi *multi-head attention*, ini diletakkan melalui lapisan dekat atau padat. Kode berikut diimplementasikan menggunakan lapisan *linear* padat dengan jumlah *heads*. Kemudian, hasilnya diubah ke bentuk *batch*, jumlah kepala, sebelum diterapkan dalam lapisan *multi* kepala. Untuk efisiensi, fungsi perhatian produk yang diskalakan diterapkan dalam satu panggilan. Dalam langkah perhatian, lapisan penyamaran yang sesuai harus digunakan. Keluaran perhatian untuk setiap kepala harus diubah dengan *tf.transpose* dan *tf.reshape*, dan kemudian dimasukkan melalui lapisan *dense* akhir. *Multi head attention* menjalankan semua delapan kepala perhatian di setiap lokasi dalam urutan. Ini mengembalikan vektor baru dengan panjang yang sama di setiap lokasi dalam urutan. Kode program dapat dilihat pada **Gambar 3.21** berikut,

```
class MultiHeadAttention(tf.keras.Model):
    def __init__(self, model_size, h):
        super(MultiHeadAttention, self).__init__()
        self.key_size = model_size // h
        self.h = h
        self.wq = tf.keras.layers.Dense(model_size) #[tf.keras.layers.Dense(key_size) for _ in range(h)]
        self.wk = tf.keras.layers.Dense(model_size) #[tf.keras.layers.Dense(key_size) for _ in range(h)]
        self.wv = tf.keras.layers.Dense(model_size) #[tf.keras.layers.Dense(value_size) for _ in range(h)]
        self.wo = tf.keras.layers.Dense(model_size)

    def call(self, decoder_output, encoder_output, mask=None):


        query = self.wq(decoder_output)
        key = self.wk(encoder_output)
        value = self.wv(encoder_output)

        # Split for multihead attention
        batch_size = query.shape[0]
        query = tf.reshape(query, [batch_size, -1, self.h, self.key_size])
        query = tf.transpose(query, [0, 2, 1, 3])
        key = tf.reshape(key, [batch_size, -1, self.h, self.key_size])
        key = tf.transpose(key, [0, 2, 1, 3])
        value = tf.reshape(value, [batch_size, -1, self.h, self.key_size])
        value = tf.transpose(value, [0, 2, 1, 3])

        score = tf.matmul(query, key, transpose_b=True) / tf.math.sqrt(tf.dtypes.cast(self.key_size, dtype=tf.float32))

        if mask is not None:
            score *= mask
            score = tf.where(tf.equal(score, 0), tf.ones_like(score) * -1e9, score)

        alignment = tf.nn.softmax(score, axis=-1)
        context = tf.matmul(alignment, value)
        context = tf.transpose(context, [0, 2, 1, 3])
        context = tf.reshape(context, [batch_size, -1, self.key_size * self.h])

        heads = self.wo(context)
        # heads has shape (batch, decoder_len, model_size)
        return heads, alignment

temp_mha = MultiHeadAttention(512, 8) # model_size & head (h)
y = tf.random.uniform((1, 60, 512))  # (batch_size, encoder_sequence, d_model)
out, attn = temp_mha(y, y, None)
out.shape, attn.shape
```

Gambar 3. 21 Kode program pada tahap *multi-head attention*

Berikur *output* yang dihasilkan dari fungsi *multi-head attention* dapat dilihat pada **Gambar 3.22**

```
(TensorShape([1, 60, 512]), TensorShape([1, 8, 60, 60]))
```

Gambar 3. 22 Output dari tahap *multi-head attention*

Encoder-Decoder layer Attention

Keluaran dari setiap lapisan adalah *LayerNorm(x+sublayer(x)).* Normalisasi dilakukan pada *d_model* dan N lapisan *encoder* di *transformer.*

Proses Encoder layer Attention

Meskipun topologi semua *encoder* sama, bobot untuk setiap lapisan tidak dibagi. Vaswani (2017) menyatakan bahwa *encoder* memiliki N = 6 lapisan, 8 lapisan *head*, dan 1024 ruang dimensi. Dalam penelitian ini, *hyperparameter* diubah menjadi lebih kecil agar relatif lebih cepat selama pelatihan. Proses pembuatan *encoder* ini menggunakan masukan *embeddings, positional encoding*, dan lapisan *encoder* N. Masuk ke lapisan *encoder* melalui *embeddings* yang digabungkan dengan *positional encoding* sebelum masuk ke lapisan berikutnya. Teks dikonversi menjadi vektor angka berkat penggabungan ini. Keluarannya dijadikan masukan ke lapisan *encoder*. Dua sublapisan *encoder* di setiap lapisan terdiri dari mekanisme *self-attention multi-head* dan jaringan *feed forward* sederhana yang sepenuhnya bergantung pada posisi. Dengan menggunakan koneksi residual di masing-masing sub-lapisan,

lapisan normalisasi akan dilanjutkan. Oleh karena itu, *output encoder* ini dimasukkan ke tahap *decoder*. Kode program pada tahap ini akan ditunjukkan pada gambar berikut.

```
class Encoder(tf.keras.Model):

    def __init__(self, vocab_size, model_size, num_layers, h):
        super(Encoder, self).__init__()
        self.model_size = model_size
        self.num_layers = num_layers
        self.h = h
        self.embedding = tf.keras.layers.Embedding(vocab_size, model_size)
        self.embedding_dropout = tf.keras.layers.Dropout(0.1)
        self.attention = [MultiHeadAttention(model_size, h) for _ in range(num_layers)]
        self.attention_dropout = [tf.keras.layers.Dropout(0.1) for _ in range(num_layers)]

        self.attention_norm = [tf.keras.layers.LayerNormalization(
            epsilon=1e-6) for _ in range(num_layers)]

        self.dense_1 = [tf.keras.layers.Dense(
            MODEL_SIZE * 4, activation='relu') for _ in range(num_layers)]
        self.dense_2 = [tf.keras.layers.Dense(
            model_size) for _ in range(num_layers)]
        self.ffn_dropout = [tf.keras.layers.Dropout(0.1) for _ in range(num_layers)]
        self.ffn_norm = [tf.keras.layers.LayerNormalization(
            epsilon=1e-6) for _ in range(num_layers)]

    def call(self, sequence, training=True, encoder_mask=None):
        """ Forward pass for the Encoder
        Args:
            sequence: source input sequences
            training: whether training or not (for Dropout)
            encoder_mask: padding mask for the Encoder's Multi-Head Attention

        Returns:
            The output of the Encoder (batch_size, length, model_size)
            The alignment (attention) vectors for all layers
        """
        embed_out = self.embedding(sequence)

        embed_out *= tf.math.sqrt(tf.cast(self.model_size, tf.float32))
        embed_out += pes[:sequence.shape[1], :]
        embed_out = self.embedding_dropout(embed_out)

        sub_in = embed_out
        alignments = []

        for i in range(self.num_layers):
            sub_out, alignment = self.attention[i](sub_in, sub_in, mask=encoder_mask)
            sub_out = self.attention_dropout[i](sub_out, training=training)
            sub_out = sub_in + sub_out
            sub_out = self.attention_norm[i](sub_out)

            alignments.append(alignment)
            ffn_in = sub_out

            ffn_out = self.dense_2[i](self.dense_1[i](ffn_in))
            ffn_out = self.ffn_dropout[i](ffn_out, training=training)
            ffn_out = ffn_in + ffn_out
            ffn_out = self.ffn_norm[i](ffn_out)

            sub_in = ffn_out

        return ffn_out, alignments

H = 8
NUM_LAYERS = 4
vocab_size = len(id_tokenizer.word_index) + 1
ml_vocab_size = len(ml_tokenizer.word_index) + 1
id_vocab_size = len(id_tokenizer.word_index) + 1

encoder = Encoder(vocab_size, MODEL_SIZE, NUM_LAYERS, H)

print(vocab_size)

sequence_in = tf.constant([[1, 2, 3, 0, 0]])
encoder_output, _ = encoder(sequence_in)
encoder_output.shape
```

Gambar 3. 23 Kode program pada tahap *encoder layer attention*

Berikut *output* yang dihasilkan dari kode program pada **Gambar 3.23**

```
9105
TensorShape([1, 5, 512])
```

Gambar 3. 24 *output* pada tahap *encoder layer attention*

Proses Decoder layer Attention

Decoder adalah tahap yang menghasilkan keluaran berupa prediksi atau kemungkinan kata tergantung pada prosedur yang telah dilakukan pada setiap time_step. Embeddings,

positional encoding, dan N *decoder* layers adalah tiga output dari proses decoding ini. Setiap sub-lapisan memiliki koneksi residual, yang diikuti oleh lapisan normalisasi (Vaswani et.al., 2017). Sub-lapisan perhatian diri tumpukan *decoder* telah diubah untuk mencegah posisi yang telah ditetapkan sebelumnya berpindah ke posisi berikutnya. *Token* dimasukkan ke dalam kalimat sasaran melalui *embeddings*, yang kemudian digabungkan dengan *positional encoding*. Lapisan *decoder* menerima masukan dari *output* penjumlahan lapisan *encoder*. *Embeddings* berfungsi sebagai representasi *token* dalam ruang d-dimensi. dimana *token* yang memiliki arti yang sama ditempatkan dalam kelompok yang sama. *Embeddings* tidak mengkodekan posisi *token* dalam kalimat. Namun, berdasarkan kesamaan makna dan tempatnya dalam kalimat, *token* akan lebih dekat satu sama lain dalam ruang d-dimensi berdasarkan *positional encoding*. Variabel *bot_sub_out* dan *mid_sub_out* menyimpan masukan dari penjumlahan melalui *embeddings* yang terpisah menjadi dua lapisan *vektor* perhatian. *Vektor* lapisan yang dihasilkan dari penggabungan kedua lapisan ini dimasukkan ke lapisan *feed forward network* yang bergantung pada posisi. Setelah itu, lapisan di masing-masing dari dua sublapisan akan dinormalisasi dengan menggunakan koneksi residual. *batch_size, target_seq_len*, dan *d_model* adalah bentuk keluaran perhatian *decoder*. Kode program pada proses *decoder* dapat dilihat pada **Gambar 3.25** berikut.

```
""" Class for the Decoder
    Args:
        model_size: d_model in the paper (depth size of the model)
        num_layers: number of layers (Multi-Head Attention + FNN)
        h: number of attention heads
        embedding: Embedding layer
        embedding_dropout: Dropout layer for Embedding
        attention_bot: array of bottom Multi-Head Attention layers (self attention)
        attention_bot_dropout: array of Dropout layers for bottom Multi-Head Attention
        attention_bot_norm: array of LayerNorm layers for bottom Multi-Head Attention
        attention_mid: array of middle Multi-Head Attention layers
        attention_mid_dropout: array of Dropout layers for middle Multi-Head Attention
        attention_mid_norm: array of LayerNorm layers for middle Multi-Head Attention
        dense_1: array of first Dense layers for FFN
        dense_2: array of second Dense layers for FFN
        ffn_dropout: array of Dropout layers for FFN
        ffn_norm: array of LayerNorm layers for FFN
        dense: Dense layer to compute final output
    """

class Decoder(tf.keras.Model):

    def __init__(self, vocab_size, model_size, num_layers, h):
        super(Decoder, self).__init__()
        self.model_size = model_size
        self.num_layers = num_layers
        self.h = h
        self.embedding = tf.keras.layers.Embedding(vocab_size, model_size)
        self.embedding_dropout = tf.keras.layers.Dropout(0.1)
        self.attention_bot = [MultiHeadAttention(model_size, h) for _ in range(num_layers)]
        self.attention_bot_dropout = [tf.keras.layers.Dropout(0.1) for _ in range(num_layers)]
        self.attention_bot_norm = [tf.keras.layers.LayerNormalization(
            epsilon=1e-6) for _ in range(num_layers)]
        self.attention_mid = [MultiHeadAttention(model_size, h) for _ in range(num_layers)]
        self.attention_mid_dropout = [tf.keras.layers.Dropout(0.1) for _ in range(num_layers)]
        self.attention_mid_norm = [tf.keras.layers.LayerNormalization(
            epsilon=1e-6) for _ in range(num_layers)]

        self.dense_1 = [tf.keras.layers.Dense(
            MODEL_SIZE * 4, activation='relu') for _ in range(num_layers)]
        self.dense_2 = [tf.keras.layers.Dense(
            model_size) for _ in range(num_layers)]
        self.ffn_dropout = [tf.keras.layers.Dropout(0.1) for _ in range(num_layers)]
        self.ffn_norm = [tf.keras.layers.LayerNormalization(
            epsilon=1e-6) for _ in range(num_layers)]

        self.dense = tf.keras.layers.Dense(vocab_size)
```

Analisis Hasil

Berikut ini merupakan analisis hasil pengujian yang telah dilakukan:

1. Pengujian otomatis dengan metode penambahan jumlah epoch secara konsisten 10 epoch, dimulai dari epoch 50 hingga epoch 90 dengan 500 korpus kalimat uji dan 4.500 korpus kalimat latih yang digunakan pada pengujian pertama sampai pengujian kelima, kemudian didapatlah nilai akurasi dari kedua jenis korpus paralel yang menunjukkan nilai tertinggi berdasarkan nilai skor BLEU yaitu pada korpus sebelum optimasi menghasilkan akurasi 45,04% pada epoch 70, sedangkan pada korpus setelah optimasi menghasilkan akurasi 51,31% pada epoch 60.
2. Pada pengujian berdasarkan jumlah epoch, dapat dilihat jumlah epoch sangat mempengaruhi hasil terjemahan pada kedua jenis korpus paralel dan juga mempengaruhi jumlah kalimat yang dapat diterjemahkan.
3. Pada pengujian berdasarkan k-fold cross validation, sebelumnya dapat dilihat jumlah epoch 70 memberikan akurasi terbaik pada korpus sebelum optimasi sedangkan jumlah epoch 60 memberikan akurasi terbaik pada korpus setelah optimasi, sehingga jumlah epoch tersebut akan diterapkan pada kedua model NMT, yang masing-masing akan dilakukan pengujian berdasarkan k-fold cross validation. Hasil nilai rata-rata k-fold cross validation adalah 41,56% untuk korpus sebelum optimasi dan 44,95% untuk korpus setelah optimasi. Peningkatan rata-rata nilai akurasi kurang lebih sebesar 8,15%.
4. Pada pengujian yang dilakukan oleh Siti Tirta Dinar (2020) menggunakan mesin penerjemah statistik, menggunakan metode optimasi korpus mendapatkan peningkatan rata-rata nilai BLEU sebesar 4%. Dengan Sistem *baseline* menghasilkan rata-rata nilai BLEU sebesar 61,25%. Sistem korpus setelah optimasi menghasilkan rata-rata nilai BLEU sebesar 63,70%. Sedangkan pengujian ini dengan menggunakan *Neural Machine Translation* dengan metode yang sama hanya bisa mencapai rata-rata nilai akurasi sebesar adalah 41,56% untuk korpus sebelum optimasi dan 44,95%.

Kesimpulan dan Saran

Kesimpulan

Berdasarkan hasil analisis dan pengujian yang sudah dilakukan dapat diambil kesimpulan sebagai berikut:

1. Berdasarkan pengujian dengan metode penambahan jumlah epoch secara konsisten, jenis korpus terbaik adalah korpus yang telah dioptimasi dengan nilai akurasi sebesar 51,13% tanpa out-of-vocabulary (OOV) pada epoch 60 dan korpus sebelum optimasi dengan nilai akurasi 45,04% pada epoch 70 tanpa OOV.
2. Berdasarkan pengujian k-fold cross validation menunjukkan korpus setelah optimasi memiliki nilai akurasi diatas korpus sebelum optimasi pada seluruh data uji dengan nilai rata-rata akurasi sebesar 44,95% tanpa OOV sedangkan 41,56% tanpa OOV pada korpus sebelum optimasi. Terdapat selisih nilai akurasi sebesar 3,39%, sehingga membuat metode optimasi korpus terbukti dapat meningkatkan rata-rata nilai akurasi.
3. Peralihan mesin penerjemah dari penelitian sebelumnya yaitu milik Siti Tirta Dinar (2020) terbukti bahwa mesin penerjemah statistik memiliki rata-rata nilai akurasi yang lebih tinggi yaitu sebesar 61,25% untuk korpus sebelum optimasi dan korpus setelah optimasi menghasilkan rata-rata nilai BLEU sebesar 63,70% dengan peningkatan rata-rata nilai BLEU sebesar 4%. Sedangkan untuk *Neural Machine Translation* hanya mendapatkan rata-rata nilai akurasi sebesar 44,95% untuk korpus setelah optimasi dan 41,56% untuk korpus sebelum optimasi dengan peningkatan rata-rata nilai akurasi kurang lebih sebesar 8,15%. Keduanya diuji dengan menggunakan jumlah korpus dan metode yang sama yaitu 5000 korpus paralel bahasa Indonesia dan bahasa Melayu Ketapang dan menggunakan metode optimasi korpus.

Saran

Adapun saran yang dapat diberikan sebagai pengembangan dari penelitian ini adalah :

1. Jumlah epoch dalam pelatihan model NMT mempengaruhi nilai akurasi hasil terjemahan tetapi tidak mempengaruhi jumlah kalimat yang dapat diterjemahkan, karena kuantitas dan kualitas korpus data latihlah yang berkontribusi pada kosakata model NMT, sehingga perlu dilakukan penelitian lebih lanjut untuk mengatasi kalimat yang tidak dapat diterjemahkan karena dianggap memiliki kata yang tidak diketahui *(unknown word)* yang disebabkan kata tersebut tidak terdaftar di dalam kosakata data latih.
2. Selain penggunaan korpus teks paralel bahasa sumber dan bahasa target dalam pelatihan, perlu juga memanfaatkan kamus diluar korpus seperti kamus tesaurus bahasa indonesia yang bisa dijadikan bahan untuk mengatasi OOV yang sering ditemukan pada mesin penerjemah, sehingga dapat meningkatkan akurasi yang dihasilkan.
3. Perlunya penelitian lebih jauh dengan menggunakan metode lain, arsitektur lain maupun bahasa lainnya sehingga dapat mengetahui lebih jauh perbandingan antara kedua mesin yang telah diuji pada penelitian ini yaitu *Neural Machine Translation* dan mesin penerjemah statistik.

DAFTAR PUSTAKA

Abidin, Z., Sucipto, A., & Budiman, A. (2018). *Penerjemahan Kalimat Bahasa Lampung-Indonesia Dengan Pendekatan Neural Machine Translation Berbasis Attention Translation of Sentence Lampung-Indonesian Languages With Neural Machine Translation Attention Based. Jurnal Kelitbangan*, *06*(02), 191–206.

Amalia, F. (2007). *Peningkatan kemampuan menerjemahkan bahasa prancis ke dalam bahasa indonesia melalui model penerjemahan pedagogis*. Tesis, tidak diterbitkan. Bandung: FPBS UPI.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Bahdanau, D., Cho, K. H., & Bengio, Y. (2015). Neural machine Foldanslation by jointly learning to align and Foldanslate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Foldack Proceedings*, 1–15.

Chaer, Abdul. (2011). Tata Bahasa Praktis Bahasa Indonesia. Jakarta: Rineka Cipta.

Chen, K., Wang, R., Utiyama, M., Liu, L., Tamura, A., Sumita, E., & Zhao, T. (n.d.). Neural Machine Translation with Source Dependency Representation.

Dharmawan, E., Sujaini, H., & Muhardi, H. (2020). Perbandingan Nilai Akurasi Terhadap Penggunaan Part of Speech Set pada Mesin Penerjemah Statistik. Jurnal Sistem Dan Teknologi Informasi (Justin), 8(3), 250. https://doi.org/10.26418/justin.v8i3.39810

Dinar, S. T., Sujaini, H., & Safriadi, N. (2020). *Optimasi Korpus untuk Meningkatkan Nilai Akurasi Mesin Penerjemah Statistik (Studi Kasus Bahasa Indonesia ke Bahasa Melayu Ketapang) Corpus Optimizing To Increase Value Of The Accuracy For Statistical Machine Foldanslation (Study Case Indonesia Language*. https://doi.org/10.26418/justin.v

Dio, M. G. A. 2018. *Perbaikan Kualitas Korpus untuk Meningkatkan Kualitas Mesin Penerjemah Statistik (Studi Kasus : Bahasa Indonesia – Jawa Krama)*. Skripsi Pada Fakultas Teknik Prodi Teknik Informatika Universitas Tanjungpura: Pontianak.

Fauziyah, Y., Ilyas, R., Kasyidi, F., Achmad Yani, J., Sains dan Informatika, F., Jenderal Achmad Yani, U., & Terusan Sudirman, J. (2022). MESIN PENTERJEMAH BAHASA INDONESIA-BAHASA SUNDA MENGGUNAKAN RECURRENT

NEURAL NETWORKS (Vol. 16, Issue 2). https://ejurnal.teknokrat.ac.id/index.php/teknoinfo/index

Fbk, L. B., Italy, T., Bisazza, A., & Fbk, M. F. (n.d.). Neural versus Phrase-Based Machine Translation Quality: a Case Study. http://www.ted.com/

Ginting, A. (2016). Penerjemah Dua Arah Bahasa Indonesia Ke Bahasa Daerah (Karo) Menggunakan Teknik Statistical Machine Translation (Smt) Sebagai Fitur Pada Situs Web Untuk Meningkatkan Web Traffic. *Telematika MKOM*, *4*(1), 60–72.

Grimes, B. F. Ed. 1988. *Ethnologue: languages of the world. Dallas*, Texas: Summer Institute of Linguistics, Inc

Gunawan. W, Sujaini. H, Tursina., 2021 "*Analisis Perbandingan Nilai Akurasi Mekanisme Attention Bahdanau dan Luong pada Neural Machine Foldanslation Bahasa Indonesia ke Bahasa Melayu Ketapang dengan Arsitektur Recurrent Neural Network*". Jurnal Edukasi dan Penelitian Informatika, Vol. 7, No. 3

Hunston, S. 2002. *Corpora in Applied Linguistics*. Cambridge: Cambridge University Press.

Kishore Papineni, Salim Roukos, Todd Ward, dan Wei-Jing Zhu. 2002. BLEU: a Method for Automatic Evaluation of Machine Foldanslation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*.pp. 311-318.

Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Israd, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G. Murray, Benoit Stiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, dan Xiaoqing Zheng. 2016. TensorFlow: A system for large-scale machine learning. In *12th USENIX symposium on operating systems design and implementation* (OSDI 16) (pp. 265-283).

Meilinda. D.A, Sujaini. H, Tursina. *Pivot Language Bahasa Melayu Pontianak ke Bahasa Bugis Menggunakan Neural Machine Translation.* Jurnal Edukasi dan Penelitian Informatika, Vol. 7, No. 3

Mohamed, Y. A., Khanan, A., Bashir, M., Mohamed, A. H. H. M., Adiel, M. A. E., & Elsadig, M. A. (2024). The Impact of Artificial Intelligence on Language Translation: A Review. IEEE Access, 12, 25553–25579. https://doi.org/10.1109/ACCESS.2024.3366802

Mulyanto, Agus dkk. Penerapan Convolutional Neural Network (CNN) pada Pengenalan Aksara Lampung Berbasis Optical Character Recognition (OCR). https://colab.research.google.com. JEPIN (Jurnal Edukasi dan Penelitian Informatika)

Philipp Koehn. 2017. Neural machine Foldanslation. *arXiv preprint arXiv:1709.07809.*

Sennrich, R., Haddow, B., & Birch, A. (2015). Improving Neural Machine Translation Models with Monolingual Data. http://arxiv.org/abs/1511.06709

Sujaini, H. & Bijaksana, A., 2014. *SFoldategi Memperbaiki Kualitas Korpus untuk Meningkatkan Kualitas Mesin Penerjemah Statistik*. Seminar Nasional Teknologi Informasi XI.

Sujaini, H., Cahyawijaya, S., & Putra, A. B. (2023). Analysis of Language Model Role in Improving Machine Translation Accuracy for Extremely Low Resource Languages. Journal of Advances in Information Technology, 14(5), 1073–1081. https://doi.org/10.12720/jait.14.5.1073-1081

Sujaini, H. 2018. *Peningkatan Akurasi Penerjemah Bahasa Daerah dengan Optimasi Korpus Paralel*. Jurnal Nasional Teknik ElekFoldo dan Teknologi Informasi (JNTETI). Vol 7, No. 1.

Sulissusiawan, A. 1998. *Struktur Bahasa Melayu Dialek Ketapang*. Jakarta, Pusat Pembinaan Dan Pengembangan Bahasa.

Tanuwijaya, Hansel. 2009. *Penerjemahan Inggris-Indonesia Menggunakan Mesin Penerjemah Statistik Dengan Word Reordering dan Phrase Reordering*. Jakarta, Jurnal ilmu Komputer dan Informasi Vol 2 No 1, hal 17-24, 2009

Wang, H., Wu, H., He, Z., Huang, L., & Church, K. W. (2022). Progress in Machine Translation. In Engineering (Vol. 18, pp. 143–153). Elsevier Ltd. https://doi.org/10.1016/j.eng.2021.03.023

Wardhana, H., Yadi Dharma, I. M., Marzuki, K., & Syarif Hidayatullah, I. (2024). Implementation of Neural Machine Translation in Translating from Indonesian to Sasak Language. MATRIK : Jurnal Manajemen, Teknik Informatika Dan Rekayasa Komputer, 23(2), 465–476. https://doi.org/10.30812/matrik.v23i2.3465

Zhixing Tan, Shuo Wang, Zonghan Yang, Gang Chen, Xuancheng Huang, Maosong Sun, dan Yang Liu. 2020. Neural Machine Foldanslation: A review of Methods, Resources, and Tools. *AI Open*, 1, 5-21.