PENERJEMAHAN BAHASA LAMPUNG-BAHASA INDONESIA MENGGUNAKAN *LONG SHORT-TERM MEMORY*

(Skripsi)

Oleh

MOHAMMED RAIHAN AKBAR

NPM 1917051061

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS LAMPUNG

BANDAR LAMPUNG

2024

PENERJEMAHAN BAHASA LAMPUNG-BAHASA INDONESIA MENGGUNAKAN *LONG SHORT-TERM MEMORY*

Oleh

MOHAMMED RAIHAN AKBAR

Skripsi

Sebagai Salah Satu Syarat untuk Mencapai Gelar

SARJANA KOMPUTER

Pada

Jurusan Ilmu Komputer

Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung

FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM

UNIVERSITAS LAMPUNG

BANDAR LAMPUNG

2024

ABSTRAK

PENERJEMAHAN BAHASA LAMPUNG-BAHASA INDONESIA MENGGUNAKAN *LONG SHORT-TERM MEMORY*

Oleh

MOHAMMED RAIHAN AKBAR

Penelitian ini bertujuan untuk mengembangkan model penerjemahan otomatis dari bahasa Lampung Api ke bahasa Indonesia menggunakan *long short-term memory* (LSTM). Mengingat bahasa Lampung merupakan bahasa daerah yang terancam punah, penelitian ini fokus pada penerapan model LSTM untuk mengatasi tantangan dalam menerjemahkan bahasa tersebut. Model dilatih dengan data yang telah dibersihkan dan dilakukan *tokenisasi*, lalu dievaluasi menggunakan analisis kuantitatif berupa plot *loss* dan metrik BLEU dan analisis kualitatif berupa contoh hasil penerjemahan. Hasil evaluasi pada set latih menunjukkan skor BLEU 1-gram sebesar 0,7900, namun menurun pada n-gram lebih tinggi: 2-gram sebesar 0,7025, 3-gram sebesar 0,6205, dan 4-gram sebesar 0,5290. Pada set uji, skor BLEU 1-gram turun menjadi 0,5118, dengan penurunan pada n-gram berikutnya. Model menunjukkan kesulitan dalam menangani konteks kompleks, terutama pada kalimat panjang, di mana skor 4-gram turun menjadi 0,0532 di set uji pada kalimat 10 kata. Temuan ini mengindikasikan keterbatasan LSTM dalam menangani *long-range dependencies*. Penambahan mekanisme *attention* dapat dipertimbangkan untuk perbaikan. Penelitian ini diharapkan berkontribusi pada pengembangan teknologi penerjemahan bahasa daerah di Indonesia dan mendukung pelestarian Bahasa Lampung.

Kata kunci: penerjemahan mesin, bahasa Lampung, bahasa Indonesia, *long short-term memory*, pemrosesan bahasa alami, pelestarian bahasa

ABSTRACT

LONG SHORT-TERM MEMORY FOR LAMPUNG-INDONESIA TRANSLATION

BY

MOHAMMED RAIHAN AKBAR

This research aims to develop an automatic translation model from the Lampung Api language to Indonesian using long short-term memory (LSTM). Considering that the Lampung language is an endangered regional language, this study focuses on applying the LSTM model to address the challenges in translating the language. The model was trained on cleaned and tokenized data and evaluated using quantitative analysis in the form of loss plots and BLEU metrics, as well as qualitative analysis through translation examples. Evaluation results on the training set showed a BLEU 1-gram score of 0.7900, but it decreased with higher n-grams: 2-gram at 0.7025, 3-gram at 0.6205, and 4-gram at 0.5290. On the test set, the BLEU 1-gram score dropped to 0.5118, with further declines in the subsequent n-grams. The model demonstrated difficulties in handling complex contexts, especially in long sentences, where the 4-gram score dropped to 0.0532 in the test set for sentences with 10 words. These findings indicate the limitations of LSTM in handling long-range dependencies. The addition of an attention mechanism may be considered for improvement. This research is expected to contribute to the development of regional language translation technology in Indonesia and support the preservation of the Lampung language.

Keywords: machine translation, Lampung language, Indonesian language, long short-term memory, natural language processing, language preservation

Judul Skripsi	:	**PENERJEMAHAN BAHASA LAMPUNG-BAHASA INDONESIA MENGGUNAKAN *LONG SHORT-TERM MEMORY***
Nama Mahasiswa	:	**Mohammed Raihan Akbar**
Nomor Pokok Mahasiswa	:	**1917051061**
Program Studi	:	**Ilmu Komputer**
Fakultas	:	**Matematika dan Ilmu Pengetahuan Alam**

MENYETUJUI

1. Komisi Pembimbing

Dr. Ir. Kurnia Muludi, M.S.Sc.
NIDN. 0216066401

2. Ketua Jurusan Ilmu Komputer

Dwi Sakethi, S.Si., M.Kom.
NIP. 196806111998021001

MENGESAHKAN

1. Tim Penguji

   Ketua : **Dr. Ir. Kurnia Muludi, M.S.Sc.** .....

   Penguji Pembahas : **Dr. rer. nat. Akmal Junaidi, M.Sc** .....

   Penguji Pembahas : **Favorisen R. Lumbanraja, Ph.D.** .....

2. Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam

Dr. Eng. Heri Satria, S.Si., M.Si.

NIP. 197110012005011002

Tanggal Lulus Ujian Skripsi: **7 Agustus 2024**

PERNYATAAN

Saya yang bertanda tangan di bawah ini:

Nama : Mohammed Raihan Akbar
NPM : 1917051061

Dengan ini menyatakan bahwa skripsi saya yang berjudul "PENERJEMAHAN BAHASA LAMPUNG-BAHASA INDONESIA MENGGUNAKAN *LONG SHORT-TERM MEMORY*" adalah benar hasil karya saya sendiri dan bukan karya orang lain. Seluruh tulisan yang tertulis dalam skripsi ini telah mengikuti kaidah penulisan karya tulis ilmiah Universitas Lampung. Jika di kemudian hari terbukti skripsi saya adalah hasil penjiplakan atau dibuat oleh orang lain, maka saya bersedia menerima sanksi sesuai dengan peraturan yang berlaku.

Bandar Lampung, 5 September 2024

Penulis,

Mohammed Raihan Akbar

NPM. 1917051061

RIWAYAT HIDUP

Penulis dilahirkan di Kota Bandar Lampung pada tanggal 24 Juni 2000, sebagai anak pertama dari dua bersaudara dari pasangan Bapak Dr. Nirwan Syarif, S.Si., M.Si. dan Ibu Yenny, S.E.. Penulis memiliki seorang adik perempuan bernama Maliqa Raissa Annayla.

Penulis menyelesaikan pendidikan Sekolah Dasar (SD) di SD Islam Az-Zahrah Palembang (Hanya kelas 1 SD), lalu pindah ke SD Kartika II-5 Bandar Lampung dan selesai pada tahun 2012. Kemudian penulis melanjutkan pendidikan Sekolah Menengah Pertama (SMP) di SMP Darma Bangsa Bandar Lampung yang selesai pada tahun 2015, lalu penulis melanjutkan pendidikan Sekolah Menengah Atas (SMA) di SMA Darma Bangsa Bandar Lampung, selesai pada Tahun 2018.

Penulis juga sempat menjadi mahasiswa program studi Kimia di Institut Teknologi Sumatera pada tahun 2018 sebelum menjadi mahasiswa program studi Ilmu Komputer di Universitas Lampung pada tahun 2019. Selama menjadi mahasiswa, penulis melakukan beberapa kegiatan antara lain.

1. Melaksanakan Kerja Praktik (KP) periode 2021/2022 di Badan Pengelolaan Keuangan dan Aset Daerah Provinsi Lampung.
2. Mengikuti kursus Aplikasi Android dengan Flutter program Kredensial Mikro Mahasiswa Indonesia (KMMI) pada tahun 2021.
3. Melaksanakan Kuliah Kerja Nyata (KKN) pada tahun ajaran 2022/2023 di Desa Sumberhadi, Kecamatan Melinting, Kabupaten Lampung Timur, Lampung.

4. Mengikuti ujian sertifikasi dan mendapatkan sertifikat *Junior Web Developer* yang diselenggarakan oleh Badan Nasional Sertifikasi Profesi (BNSP) pada tahun 2022.

MOTTO

“Eppur si muove.”

(Galileo Galilei)

“Ja sagen.”

(Friedrich Nietzsche)

“To improve is to change, so to be perfect is to have changed often.”

(Winston Churchill)

“In life there's roadblocks. Don't let nothing stop you.”

(Khaled Mohammed Khaled)

“The only real change come from inside.”

(Jermaine Lamarr Cole)

PERSEMBAHAN

Alhamdulillahirabbil'alamin telah Engkau ridai Ya Allah langkah hamba-Mu, sehingga pada akhirnya skripsi ini pada akhirnya dapat diselesaikan. Teriring shalawat serta salam kepada Nabi Muhammad SAW. Semoga kelak skripsi ini dapat memberikan ilmu yang bermanfaat.

Aku persembahkan karya sederhana ini kepada:

Kedua orang tua tercinta

Dr. Nirwan Syarif, S.Si., M.Si. dan Yenny, S.E.

Serta adikku tersayang

Maliqa Raissa Annayla

Terima kasih atas dukungan serta doa yang telah diberikan.

Terima kasih untuk teman-teman seperjuangan di Jurusan Ilmu Komputer, semoga amal kebaikan yang telah dilakukan mendapat balasan dari Allah SWT.

Almamater yang penulis cintai dan banggakan,

Universitas Lampung

SANWACANA

Puji syukur *alhamdulillah* ke hadirat Allah SWT Yang Maha Pengasih lagi Maha Penyayang, karena atas rahmat dan hidayah-Nya penulis dapat menyelesaikan skripsi ini yang berjudul **"PENERJEMAHAN BAHASA LAMPUNG-BAHASA INDONESIA MENGGUNAKAN *LONG SHORT-TERM MEMORY*".** Tak lupa shalawat serta salam tercurah limpahkan kepada junjungan besar Nabi Muhammad SAW sebagai suri teladan yang baik dan pemimpin bagi kaumnya.

Skripsi ini merupakan salah satu syarat untuk mencapai gelar Sarjana Komputer di Universitas Lampung. Penulis menyadari bahwa masih terdapat kekurangan dalam penulisan skripsi ini karena keterbatasan kemampuan dan pengetahuan yang penulis miliki. Tanpa bantuan dari berbagai pihak, skripsi ini mustahil dapat terwujud dengan baik. Dalam kesempatan ini penulis mengucapkan terima kasih kepada:

1. Ibu Prof. Dr. Ir. Lusmeilia Afriani, D.E.A.IPM, ASEAN Eng., selaku Rektor Universitas Lampung;
2. Bapak Dr. Eng. Heri Satria, S.Si., M.Si., selaku Dekan Fakultas Matematika dan Ilmu Pengetahuan Alam Universitas Lampung;
3. Bapak Dwi Sakethi, S.Si., M.Kom., selaku Ketua Jurusan Ilmu Komputer Universitas Lampung;
4. Ibu Anie Rose Irawati, S.T., M.Cs., selaku Sekretaris Jurusan Ilmu Komputer Universitas Lampung;
5. Bapak Dr. Ir. Kurnia Muludi, M.S.Sc. selaku Dosen Pembimbing Skripsi yang telah memberikan ide, arahan, serta masukan kepada Penulis sehingga dapat menyelesaikan skripsi ini dengan baik;

6. Bapak Dr. rer. nat. Akmal Junaidi, M.Sc., selaku Dosen Pembahas I yang telah memberikan saran dan masukan yang bermanfaat dalam proses perbaikan skripsi ini;
7. Bapak Favorisen R. Lumbanraja, Ph.D. selaku Dosen Pembahas II yang telah memberikan saran dan masukan yang bermanfaat dalam proses perbaikan skripsi ini;
8. Bapak Rizky Prabowo, M.Kom., Ardiansyah,, S.Kom., M.Kom., Ridho Sholehurrohman, M. Mat, Rahman Taufik, M.Kom selaku Dosen Pembimbing Akademik yang telah membantu dalam hal akademik selama proses perkuliahan di Jurusan Ilmu Komputer Universitas Lampung;
9. Ibu Ade Noera Maela, selaku Admin Jurusan Ilmu Komputer yang telah membantu penulis dalam mengurus segala administrasi di Jurusan Ilmu Komputer Universitas Lampung;
10. Bang Zainudin, Mas Sam, dan Mas Nofal yang telah membantu penulis dalam mengurus segala jenis perizinan di Jurusan Ilmu Komputer Universitas Lampung;
11. Bapak dan Ibu Dosen Jurusan Ilmu Komputer Universitas Lampung yang telah membantu penulis dalam proses menimba ilmu yang bermanfaat untuk penulis selama perkuliahan di Jurusan Ilmu Komputer Universitas Lampung;
12. Kedua orang tua dan adik penulis yang selalu mendoakan, menyemangati, menasehati, membiayai serta mendukung penulis baik secara moral maupun material;
13. Teman satu bimbingan skripsi, yaitu Irfan Yadi yang selalu mendukung, menyemangati, dan membantu selama proses penyusunan skripsi ini hingga selesai;
14. Teman dekat di jurusan, yaitu Irfan Yadi, Abbie Syeh Nahri dan Devi Ramadhia Fitri yang telah menjadi teman yang baik dan selalu memberikan banyak dukungan dan bantuan dalam hal apapun;

15. Teman-teman seperjuangan Ilmu Komputer 2019, yang tidak dapat disebut satu persatu yang telah mendukung dan membantu selama proses perkuliahan di Jurusan Ilmu Komputer Universitas Lampung;
16. *Validator* data penelitian, Ibu Nurul Fitrianisa, S.Pd., terima kasih sudah melakukan validasi data sehingga skripsi ini bisa dapat diselesaikan.

Terima kasih sekali lagi penulis ucapkan kepada semua pihak yang terlibat dalam proses penyelesaian skripsi ini, baik yang tertulis maupun tidak tertulis namanya, yang tidak saya sebutkan satu persatu, tanpa mengurangi rasa hormat dalam sanwacana ini. Penulis berharap semoga Allah SWT membalas semua kebaikan kalian. Akhir kata semoga skripsi ini bisa bermanfaat bagi keberlangsungan proses pembelajaran, penelitian, dan pengabdian khususnya di bidang ilmu komputer.

Bandar Lampung, 5 September 2024

Penulis

Mohammed Raihan Akbar

DAFTAR ISI

Halaman

DAFTAR TABEL

Tabel Halaman

DAFTAR GAMBAR

Gambar Halaman

DAFTAR KODE PROGRAM

Kode Program Halaman

I. PENDAHULUAN

1.1. Latar Belakang dan Masalah

Indonesia adalah negara kepulauan yang terletak di Asia Tenggara. Negara ini memiliki lebih dari 17.000 pulau yang tersebar di wilayah Samudra Hindia dan Samudra Pasifik. Luas wilayahnya sekitar 1,9 juta kilometer persegi dan memiliki populasi lebih dari 275 juta jiwa. Karena kondisi tersebut, Indonesia memiliki lebih dari 300 etnis dan bahasa yang beragam. Termasuk bahasa resmi yaitu bahasa Indonesia, dan berbagai bahasa daerah lainnya. Bahasa daerah merupakan sekumpulan bahasa yang digunakan oleh masyarakat di suatu wilayah tertentu. Ada banyak bahasa daerah yang digunakan di Indonesia, seperti bahasa Lampung, Jawa, Sunda, Minangkabau, dan bahasa-bahasa lainnya. Bahasa daerah ini sering kali digunakan dalam komunikasi sehari-hari dan memiliki peran penting dalam budaya dan tradisi masyarakat setempat.

Bahasa Lampung adalah bahasa daerah yang digunakan oleh masyarakat Lampung, provinsi di sebelah selatan Sumatra, Indonesia. Bahasa ini merupakan bahasa Austronesia yang memiliki beberapa dialek, yaitu dialek Lampung Api, Lampung Nyo. Bahasa Lampung memiliki beberapa karakteristik unik, seperti sistem pengejaan yang kompleks dan kosakata yang berbeda-beda antar dialek. Bahasa Lampung memiliki peran penting dalam budaya dan tradisi masyarakat Lampung. Beberapa cerita rakyat, lagu, dan tarian tradisional hanya dapat diterjemahkan atau dimengerti dengan menggunakan bahasa Lampung. Meskipun bahasa Indonesia adalah bahasa nasional yang digunakan sebagai bahasa pengantar di sekolah dan media massa, bahasa Lampung tetap diperlukan dan dihormati sebagai bagian

penting dari kekayaan budaya Lampung. Seiring dengan perkembangan zaman dan era globalisasi, eksistensi bahasa Lampung kini terancam. Di Provinsi Lampung sendiri bahasa Lampung sendiri semakin ditinggalkan. Keanekaragaman yang ada pada masyarakat kota Bandar Lampung mengakibatkan bahasa daerah Lampung mengalami pergeseran (Putri, 2018).

Pakar bahasa mengestimasi bahwa dalam 40 hingga 50 tahun ke depan, bahasa Lampung berisiko menghilang (Putri, 2018). Program transmigrasi yang dijalankan selama era Presiden Soeharto telah secara signifikan mengubah demografi di Lampung. Bukan lagi suku asli Lampung yang mayoritas, namun para pendatang. Situasi ini berimbas pada perubahan preferensi bahasa di kalangan masyarakat, yang mengakibatkan bahasa Lampung semakin jarang digunakan.

Salah satu upaya untuk melestarikan bahasa Lampung adalah dengan mengembangkan sistem penerjemahan dari bahasa Lampung ke bahasa Indonesia. Ini memfasilitasi pemahaman antara pembicara bahasa daerah dan bahasa nasional (Fauziyah dkk., 2022). Penerjemahan mesin, yang merupakan perangkat lunak untuk mengonversi teks dari satu bahasa ke bahasa lainnya, sangat penting dalam konteks pelestarian bahasa (Suryani dkk., 2016). Penerjemahan ini dapat dilakukan secara otomatis dari bahasa sumber ke bahasa sasaran (Sujaini, 2018). Misalnya, dari bahasa asing ke bahasa Indonesia atau sebaliknya (Faqih, 2018). Namun, ada berbagai tantangan dalam penerjemahan, baik oleh manusia maupun mesin, terutama yang berbasis konteks, yang disebabkan oleh perbedaan geografis, budaya, kebiasaan, dan leksikon (Shen dkk., 2020).

Metode penerjemahan mesin telah berkembang pesat, dimulai dari *rule-based machine translation* (RBMT) kemudian *statistical machine translation* (SMT), dan kini *neural machine translation* (NMT) yang menggunakan jaringan saraf buatan. NMT melibatkan struktur model dan lapisan proses yang kompleks, menggunakan model *sequence to sequence* (Seq2seq) (Sutskever dkk., 2014) yang terdiri dari

encoder dan *decoder* (Cho dkk., 2014). *Encoder* memproses bahasa sumber, sementara *decoder* menghasilkan terjemahan yang kemudian diubah menjadi bahasa sasaran (Garg dkk., 2019). Proses ini menggunakan lapisan *encoder-decoder* yang memanfaatkan jaringan saraf *recurrent neural network* (RNN) (Britz dkk., 2017), yang mampu memproses urutan teks secara kontekstual dan berulang.

Long short-term memory (LSTM) adalah pengembangan dari RNN, yang memiliki kemampuan dalam memproses *input* dan *output* secara kontekstual. Berdasarkan keunggulan ini, penelitian ini bertujuan untuk menerapkan LSTM dalam penerjemahan Bahasa Lampung Api ke Bahasa Indonesia serta mengevaluasi kualitas hasil terjemahan yang diperoleh. Penelitian ini difokuskan pada penggunaan LSTM dan diharapkan dapat membantu melestarikan Bahasa Lampung, memfasilitasi komunikasi antarbudaya, memperkuat identitas budaya masyarakat Lampung, meningkatkan aksesibilitas informasi, dan menyediakan referensi untuk penelitian serupa di masa depan.

Penerjemahan mesin, yang merupakan bagian dari *natural language processing* (NLP), dikenal dengan istilah *machine translation* (MT). *Google translate* adalah salah satu mesin penerjemah yang terkenal, namun bahasa Lampung belum tersedia di sana. *Neural machine translation* (NMT) kini menjadi teknik yang banyak diterapkan untuk penerjemahan mesin (Klein dkk., 2018). Berdasarkan keunggulan LSTM dalam memproses teks secara kontekstual, penelitian ini akan menggunakan LSTM untuk penerjemahan bahasa Lampung ke bahasa Indonesia. Penelitian ini dilakukan untuk mendukung keberlangsungan bahasa Lampung dengan mengembangkan mesin penerjemah yang dapat memfasilitasi komunikasi dan pelestarian budaya.

Penelitian ini menggunakan dialek Lampung Api dari bahasa Lampung. Pemilihan dialek api didasarkan pada pertimbangan dialek api lebih banyak digunakan di komunitas penutur bahasa Lampung dibandingkan dialek nyo. Ditotalkan dari

jumlah penutur seluruh ragam bahasa Lampung di *Ethnologue*, berdasarkan data sensus tahun 2000. Di mana terdapat 3.219.000 penutur untuk ragam api (kode ethnolog ljp), dan 1.800.000 untuk nyo (kode ethnolog abl) (Lewis, 2009).

1.2. Rumusan Masalah

Rumusan masalah pada penelitian ini adalah sebagai berikut:

1. Apakah LSTM dapat diterapkan pada penerjemahan bahasa Lampung-bahasa Indonesia?
2. Bagaimana kualitas hasil penerjemahan yang didapatkan dari penerjemahan bahasa Lampung-bahasa Indonesia dengan LSTM tersebut?

1.3. Tujuan Penelitian

Adapun tujuan yang ingin dicapai dalam penelitian ini adalah:

1. Menerapkan LSTM dalam penerjemahan bahasa Lampung-bahasa Indonesia.
2. Mengetahui kualitas hasil penerjemahan LSTM yang telah diterapkan dalam penerjemahan bahasa Lampung-bahasa Indonesia dengan LSTM.

1.4. Batasan Masalah

Penelitian ini dibatasi oleh beberapa hal, yaitu:

1. Penelitian ini hanya menggunakan LSTM.
2. Penelitian ini hanya dibuat untuk penerjemahan bahasa Lampung berdialek Api - bahasa Indonesia.

1.5. Manfaat Penelitian

Adapun yang menjadi manfaat dalam penelitian ini adalah sebagai berikut.

1. Melindungi dan melestarikan bahasa: Penelitian ini dapat membantu dalam usaha pelestarian bahasa Lampung dengan memfasilitasi pemahaman dan akses kepada bahasa tersebut bagi penutur bahasa Indonesia.
2. Memfasilitasi komunikasi antarbudaya: Dengan menerjemahkan bahasa Lampung ke Indonesia, penelitian dapat membantu penduduk lokal dan pendatang berkomunikasi lebih efektif, yang berpotensi memperkuat hubungan komunitas.
3. Penguatan identitas budaya: Menyediakan teknologi yang dapat menerjemahkan dari bahasa Lampung bisa memperkaya identitas budaya dan kebanggaan etnis masyarakat setempat.
4. Meningkatkan aksesibilitas informasi: Memungkinkan penutur Lampung yang tidak fasih dalam bahasa Indonesia untuk mengakses berbagai informasi dan layanan yang mayoritas disediakan dalam bahasa Indonesia dan juga sebaliknya.
5. Hasil dari penelitian ini dapat digunakan sebagai referensi untuk penelitian-penelitian yang akan datang yang memiliki kesamaan topik dengan penelitian ini.

II. TINJAUAN PUSTAKA

2.1. Penelitian Terdahulu

Tujuan dari mencari penelitian terdahulu adalah untuk membandingkan penelitian yang telah dilakukan sebelumnya dengan penelitian yang akan dilakukan selanjutnya. Selain itu, hasil penelitian tersebut akan digunakan sebagai bahan pembelajaran. Penelitian tersebut dapat dilihat Tabel 1.

Tabel 1. Penelitian Terdahulu

No	Judul Penelitian	Metode	Data	Hasil
1	*Recurrent Neural Network Language Model for English-Indonesian Machine Translation: Experimental Study* (Hermanto dkk., 2015)	1. *Recurrent neural network* 2. *Statistical based network with n-gram model*	Korpus BPPT bahasa Inggris-bahasa Indonesia.	1. Model RNN menghasilkan *perplexity*, BLEU, dan RIBES yang lebih baik dibandingkan model statistik. 2. Hasil ini tidak signifikan karena *dataset* yang digunakan kecil. 3. Pelatihan model RNN memerlukan waktu lebih lama karena kompleksitas algoritmanya.
2	*Japanese-to-English Machine Translation Using Recurrent Neural Networks* (Greenstein dan Penner, 2015)	1. Model *RNNsearch* oleh (Bahdanau dkk., 2015). 2. Model RNN *Encoder-Decoder* (Cho dkk., 2014).	*1. Kyoto wiki corpus* *2. TED corpus* *3. Tanaka corpus* 4. Korpus paralel buatan tangan	1. Model *RNNsearch* diimplementasikan dan dilatih pada berbagai *dataset*. 2. Model dapat menerjemahkan kalimat di luar sampel dengan akurasi tinggi. 3. Hasil menunjukkan bahwa model ini dapat memberikan prediksi baik jika

No	Judul Penelitian	Metode	Data	Hasil
		3. *Encoder* BiRNN (Schuster dan Paliwal, 1997). 4. *Activation Function* GRU (Cho dkk., 2014). 5. *Decoder* RNN dengan GRU.		dilatih pada *dataset* yang lebih besar.
3	Mesin Penterjemah Bahasa Indonesia-Bahasa Sunda Menggunakan *Recurrent Neural Networks* (Fauziyah dkk., 2022)	1. LSTM + *Attention* 2. LSTM 3. GRU	Korpus bahasa Sunda-bahasa Indonesia.	1. Pengujian arsitektur RNN menunjukkan GRU dengan akurasi 99,17%. 2. Model dengan *Attention* memperoleh akurasi 99,94%. 3. Model optimasi *Adam* menghasilkan akurasi 99,35%. 4. BLEU mencapai 92,63% dengan *brevity penalty* 0,929. 5. Mesin penerjemah berfungsi baik dengan kalimat dari korpus, namun kurang akurat dengan kalimat baru karena keterbatasan data.
4	*English to Bangla Machine Translation Using Recurrent Neural Network* (Siddique dkk., 2020)	1. Model *sequential* dengan GRU 2. Model *sequential* dengan LSTM	Korpus bahasa Inggris-bahasa Bangla.dari artikel.	Metode RNN (GRU dan LSTM) memberikan hasil lebih baik dibandingkan metode lain.
5	Penerapan *Neural Machine Translation* untuk Eksperimen Penerjemahan secara Otomatis pada Bahasa	RNN	Korpus bahasa Lampung-bahasa Indonesia.	1. Penerjemahan Lampung-Indonesia dengan NMT menghasilkan BLEU 41,79 (kalimat tunggal) dan 37,5 (kalimat majemuk) tanpa OOV. 2. NMT mampu menangani makna

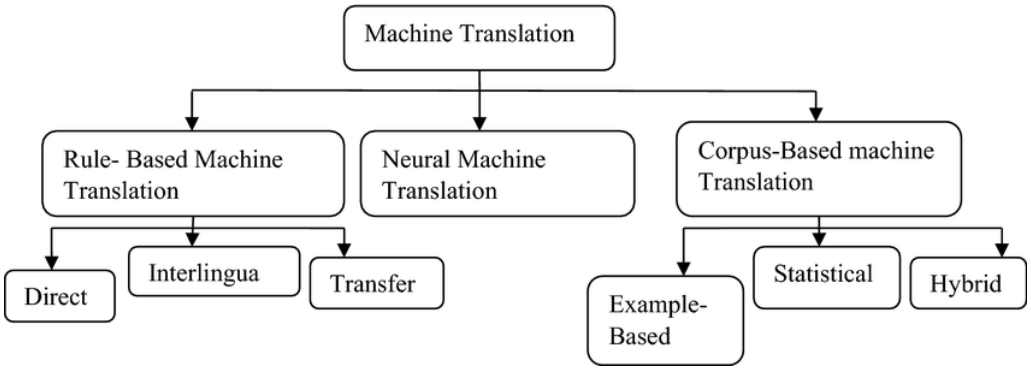
No	Judul Penelitian	Metode	Data	Hasil
	Lampung – Indonesia (Abidin, 2017)			kontekstual dalam bahasa Lampung, seperti kata *wai*, *lapah*, *sai*. 3. Kata-kata yang jarang muncul pada korpus terdeteksi sebagai OOV dan menyebabkan kegagalan terjemahan. 4. Penelitian selanjutnya dapat fokus pada kalimat yang mengandung OOV.

2.2. Uraian Tinjauan Pustaka

2.2.1. Penerjemahan

Menurut Basnett-McGuire dalam buku yang ditulis (Wuryantoro, 2018), penerjemahan adalah pengalihan teks bahasa sumber (TBSu) ke dalam teks bahasa sasaran (TBSa) untuk menjamin (1) kedua makna lahir kira-kira akan sama dan (2) struktur bahasa sumber (BSu) akan dipertahankan sedekat mungkin namun tidak sama persis sehingga struktur bahasa sasaran (BSa) akan benar-benar berubah. Dengan kata lain, terjemahan adalah pengalihan makna dari bahasa sumber (BSu) ke bahasa sasaran (BSa) sesuai dengan struktur gramatikal dan konteks budaya bahasa sasaran. Bahasa sumber adalah bahasa pertama yang akan diterjemahkan, sedangkan bahasa sasaran adalah bahasa hasil terjemahan dari bahasa sumber. Dalam hal ini, hasil terjemahan yang dihasilkan dari penerjemahan bahasa sumber ke bahasa sasaran tidak sepenuhnya diterjemahkan secara kata per kata, tetapi dapat diterjemahkan sesuai dengan budaya bahasa sasaran. Kunci kegiatan menerjemahkan ialah dengan mengalihkan makna, bukan kata (Muam dan Nugraha, 2020).

2.2.2. Penerjemahan Mesin (*Machine Translation*)



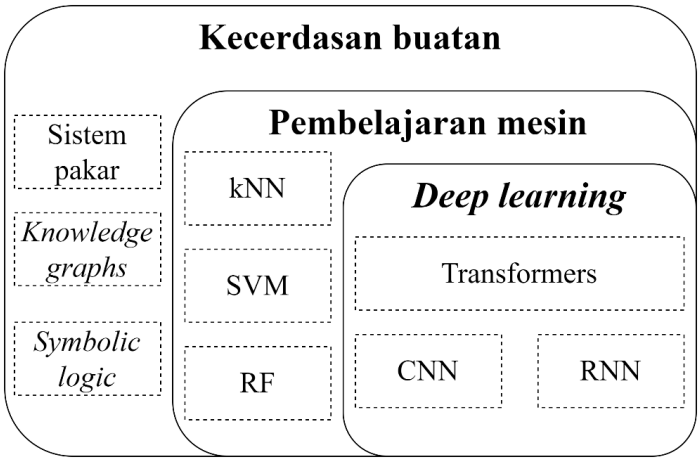
Gambar 1. Taksonomi Studi Penerjemahan Mesin (Kahlon dan Singh, 2023).

Penerjemahan mesin atau *machine translation* (MT) adalah cabang linguistik komputasional yang mempelajari penggunaan perangkat lunak komputer untuk menerjemahkan teks atau ucapan dari satu bahasa sumber ke bahasa lainnya (Prima, 2022). Tujuan utama dari penerjemahan mesin adalah untuk menyediakan sistem yang dapat menerjemahkan bahasa sumber ke bahasa sasaran dengan arti yang sama dan akurat. Penerjemahan mesin telah berkembang melalui beberapa pendekatan, termasuk pendekatan dengan aturan (*rule-based*), pendekatan dengan model statistik (*statistical-based)*, pendekatan hybrid, dan pendekatan dengan menggunakan contoh (*example-based*). Penerjemahan mesin juga telah dikembangkan dengan menggunakan teknologi *natural language processing* (NLP) dan *neural machine translation* (NMT). NLP memungkinkan sistem untuk menerima masukan dalam bentuk suara dan melakukan penerjemahan pada beberapa bahasa, seperti Indonesia, Inggris, Spanyol, dan Prancis. NMT menggunakan jaringan saraf tiruan untuk menerjemahkan teks dan memungkinkan sistem untuk belajar dari korpus teks dan meningkatkan kualitas penerjemahan.

2.2.3. *Neural Machine Translation*

Neural machine translation (NMT) menggunakan jaringan saraf tiruan untuk menerjemahkan teks dan memungkinkan sistem untuk belajar dari korpus teks dan meningkatkan kualitas penerjemahan. NMT merepresentasikan sebuah paradigma terkini dalam evolusi MT, yang memperlihatkan perbaikan substansial dibandingkan pendekatan-pendekatan konvensional sebelumnya. Penelitian yang dilakukan dalam konteks NMT telah menunjukkan potensi model ini sebagai mekanisme yang secara signifikan lebih efektif dalam menjalankan proses penerjemahan. (Abidin, 2017). (Junczys-Dowmunt dkk.) telah membuktikan performa NMT lebih baik dibanding SMT pada eksperimen 30 jenis penerjemahan. Berbeda dengan sistem penerjemahan berbasis frasa tradisional yang terdiri dari berbagai sub-komponen kecil yang disetel secara terpisah, NMT berupaya membangun dan melatih satu jaringan saraf yang besar dan terintegrasi, yang mampu membaca sebuah kalimat secara menyeluruh dan menghasilkan terjemahan yang akurat sebagai keluaran. (Bahdanau dkk., 2015).

2.2.4. Kecerdasan Buatan (*Artificial Intelligence*)



Gambar 2. Taksonomi Metodologi *Artificial Intelligence* (Cau dkk., 2023).

Kecerdasan buatan atau *artificial intelligence* (AI), adalah pengembangan sistem komputer yang dapat melakukan tugas-tugas yang biasanya memerlukan kecerdasan manusia (Purnomo, 2024). seperti pembelajaran, pemecahan masalah, pengambilan keputusan, dan lain-lain. Sistem AI dirancang untuk meniru proses berpikir dan perilaku manusia, sehingga memungkinkannya berinteraksi dengan lingkungan, mengenali pola, dan beradaptasi dengan situasi baru. Istilah "*artificial intelligence*" diciptakan oleh ilmuwan komputer John McCarthy pada tahun 1956, dan sejak saat itu, AI telah berevolusi secara signifikan, didorong oleh kemajuan dalam perangkat keras, perangkat lunak, dan penyimpanan data. AI telah diterapkan di berbagai bidang.

2.2.5. Pembelajaran Mesin (*Machine Learning*)

Pembelajaran mesin atau *machine learning* (ML), sebuah sub bidang dari kecerdasan buatan, telah mendapatkan perhatian yang signifikan dalam beberapa tahun terakhir karena potensinya untuk merevolusi berbagai industri. Bidang ini melibatkan pelatihan algoritme untuk belajar dari data dan membuat prediksi atau keputusan tanpa diprogram secara eksplisit. Konsep pembelajaran mesin berakar pada gagasan bahwa mesin dapat dirancang untuk mengenali pola dan membuat keputusan berdasarkan data yang diberikan. Menurut IBM, metode ML merupakan cabang dari kecerdasan buatan dan ilmu komputer yang berfokus pada penggunaan data dan algoritma untuk meniru cara manusia belajar dan secara bertahap dapat meningkatkan akurasinya. Semakin bagus algoritma *machine learning* yang digunakan maka akan semakin baik pula keputusan yang keluar (Faiza dkk., 2022).

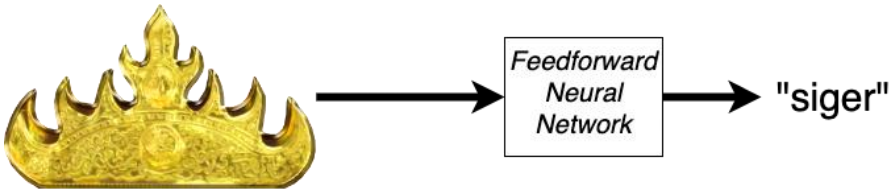
2.2.6. *Deep Learning*

Deep learning merupakan percabangan dari bidang ML yang menggunakan saraf tiruan atau disebut dengan *artificial neural networks* (ANN) yang memiliki beberapa *layers* dalam memproses atau mempelajari suatu data. *Deep learning*

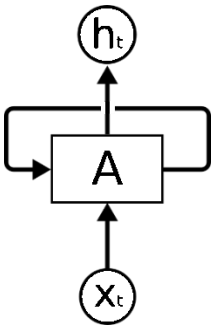
memungkinkan suatu komputer untuk belajar secara otomatis melalui pengalaman yang diberikan agar sistem tersebut dapat mengidentifikasi pola yang kompleks dalam data. *Deep learning* memiliki model jaringan saraf tiruan yang mirip dengan struktur jaringan saraf manusia, di mana model tersebut terdapat beberapa lapisan neuron buatan yang terhubung satu sama lain. Neuron tersebut berperan dalam menerima suatu *input* data, melakukan komputasi matematika dan mengirim *output* tersebut ke neuron yang lain. (Jiantono, 2023).

2.2.7. *Recurrent Neural Network*

Recurrent neural network (RNN) adalah model dari keluarga *deep learning,* RNN bertujuan untuk membuat prediksi pada data sekuensial dengan mengandalkan arsitekturnya yang berbasis memori. RNN berbeda dengan *neural network* normal, sebuah *neural network* normal (juga disebut *feedforward*) bertindak seperti fungsi pemetaan, di mana sebuah *input* tunggal diasosiasikan dengan *output* tunggal. Pada arsitektur normal, tidak ada dua *input* yang mengetahui satu sama lain, masing-masing dari mereka bergerak hanya satu arah, mulai dari *input nodes*, melewati *hidden nodes*, dan berhenti pada *output nodes*.

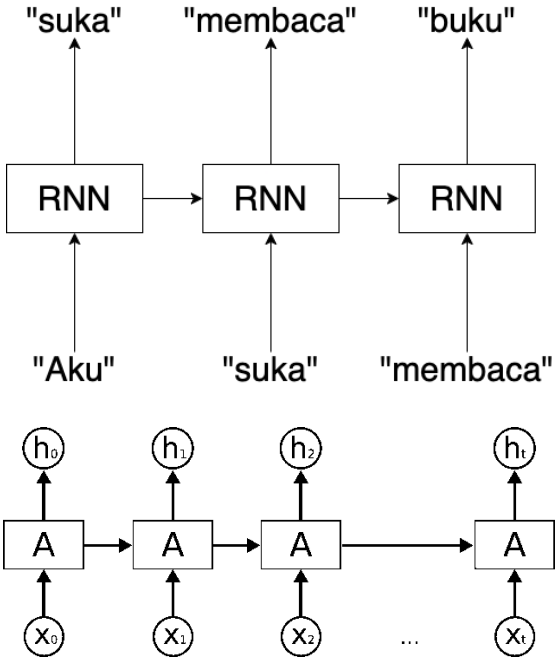


Gambar 3. *Feedforward Neural Network*.



Gambar 4. *Recurrent Neural Network* (Manu, 2021).

Kebalikannya, *recurrent* (bisa juga disebut *feedback*) *neural network* menggunakan sesuatu yang disebut *memory-state*. Ketika *input* A_1, (kata aku) ditambahkan, jaringan menghasilkan *output* B_1, (kata suka) dan menyimpan informasi tentang *input* A_1 pada *memory-state*. Ketika *input* selanjutnya A_2 (kata suka) ditambahkan, jaringan memproduksi *output* B_2 yang terkait (kata membaca) dengan bantuan *memory-state*. Lalu, *memory-state* diperbarui menggunakan informasi dari *input* A_2 baru. Operasi ini diulang pada setiap *input*:



Gambar 5. *Unrolled Recurrent Neural Network* (Du dkk., 2019).

Dengan metode ini prediksi yang kita lakukan tidak hanya bergantung pada *input* sekarang, tapi juga pada data sebelumnya. Inilah mengapa RNN adalah salah satu model yang paling mutakhir untuk melakukan *machine translation* (Kostadinov, 2018).

RNN memiliki komponen utama sebagai berikut: *Input layer* adalah lapisan pertama dari RNN dalam menerima *input* sekuens, sebagai contoh, dalam kasus data teks, setiap *input* dapat berupa satu kata atau karakter. *Hidden layer* adalah tempat di mana bagian "*recurrent*" dari RNN berperan. Lapisan *hidden layer* mengambil *input* dari *time step* sebelumnya dan meneruskannya dengan *input* saat

ini ke *time step* selanjutnya. Hal ini memungkinkan RNN untuk menyimpan memori dari *input* sebelumnya dan menggunakannya untuk memprediksi *output* untuk *input* saat ini. *Output layer*, sebagai lapisan terakhir dari RNN, menghasilkan *output* berdasarkan urutan *input* dan *hidden state*. Contoh *output* dapat berupa klasifikasi, regresi, atau distribusi probabilitas dari sekumpulan *output* yang memungkinkan. Fungsi aktivasi seperti *softmax*, *sigmoid*, *tanh*, atau ReLU diterapkan untuk menambah non-linearitas pada model.

Recurrent connection memungkinkan *hidden state* untuk diteruskan dari suatu *time step* ke *time step* selanjutnya. Hal ini dicapai dengan menambahkan suatu *loop* dalam jaringan yang memungkinkan *output* dari *time step* sebelumnya untuk dimasukkan kembali sebagai *input* pada *time step* saat ini. *Time step* adalah durasi satu *input* dalam sekuens, RNN memproses satu *input* pada satu waktu dan meneruskan *hidden state* ke *time step* berikutnya, dan memori RNN, yang bertumpu pada *hidden state*, memfasilitasi penyimpanan dan pemanfaatan informasi dari *input* yang telah lalu untuk pengambilan keputusan prediktif pada masa kini.

Secara keseluruhan, arsitektur RNN dirancang untuk menangani data sekuensial dengan mempertahankan memori *input* sebelumnya melalui penggunaan *hidden state* dan *recurrent connections*. Hal ini memungkinkan RNN untuk membuat prediksi berdasarkan konteks *input sequence*, yang berguna untuk tugas seperti penerjemahan bahasa. Dalam penerjemahan bahasa, kata-kata dalam kalimat sering kali memiliki makna yang bergantung pada konteks kalimat sebelumnya (Shelf, 2024), dan RNN dapat menangkap hubungan ini melalui *internal-state* miliknya. Salah satu alasan utama mengapa RNN cocok untuk penerjemahan bahasa adalah kemampuannya untuk menangani urutan data dengan panjang yang bervariasi. Model RNN dapat memahami konteks dari bagian teks yang sudah diproses sebelumnya dan menggunakan informasi itu untuk memprediksi bagian berikutnya dari teks yang akan diterjemahkan. Dalam konteks penerjemahan bahasa, ini berarti bahwa RNN dapat menangkap nuansa dan struktur bahasa yang lebih baik daripada model lain yang tidak memiliki mekanisme memori internal yang serupa. Beberapa

penelitian telah mendukung penggunaan RNN dalam penerjemahan bahasa. Contohnya adalah studi oleh (Cho dkk., 2014) yang menemukan RNN meningkatkan kinerja terjemahan secara keseluruhan dalam hal skor BLEU dibandingkan dengan model lainnya.

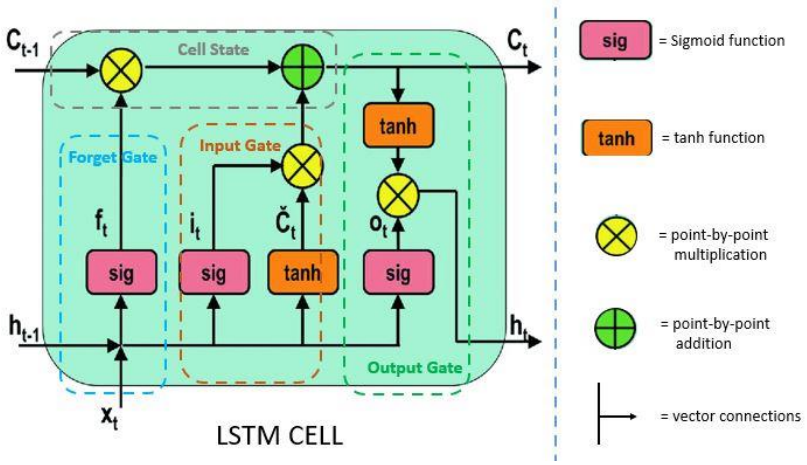
2.2.8. *Long Short-Term Memory*

Long short-term memory (LSTM), yang pertama kali diperkenalkan oleh Sepp Hochreiter dan Juergen Schmidhuber pada tahun 1997, merupakan sebuah inovasi signifikan dalam arsitektur RNN yang dirancang khusus untuk mengatasi masalah *vanishing gradient* dalam pemrosesan data sekuensial jangka panjang (Husada dan Toba, 2020). Dalam model RNN tradisional, tantangan utama yang sering muncul adalah ketidakmampuan untuk mempertahankan informasi jangka panjang secara efektif. Ketika jaringan saraf memproses urutan data yang panjang, memori lama cenderung tergantikan oleh informasi baru, menyebabkan fenomena yang dikenal sebagai *vanishing gradient*, di mana gradien yang dibutuhkan untuk pembaruan bobot menjadi sangat kecil, sehingga sulit bagi model untuk belajar dan menyimpan informasi dari tahap-tahap awal urutan.

LSTM mengatasi masalah ini melalui pengenalan *memory cell* yang menggantikan neuron pada *hidden layer* RNN (Qiu dkk., 2020), elemen utama dalam arsitektur LSTM yang dirancang untuk menyimpan informasi penting dalam jangka waktu yang lama. *Memory cell* ini tidak hanya menyimpan informasi, tetapi juga memiliki mekanisme kontrol yang canggih berupa *input gate*, *output gate*, dan *forget gate* yang secara selektif mengatur aliran informasi masuk, keluar, dan yang perlu dilupakan. *Forget gate* menentukan informasi mana yang harus dibuang dari *memory cell*, sementara *input gate* mengontrol informasi baru yang perlu disimpan, dan *output gate* mengatur kapan informasi dalam *cell* harus digunakan untuk memprediksi *output*. Dengan demikian, LSTM memberikan fleksibilitas yang lebih besar dalam mengelola ketergantungan jangka panjang dibandingkan dengan RNN tradisional, memungkinkan model untuk mempertahankan konteks dan informasi

kritis dalam urutan data yang panjang, yang sangat penting dalam berbagai aplikasi seperti penerjemahan mesin, pengenalan suara, dan analisis teks.

Struktur sel dari LSTM dapat dilihat pada gambar berikut.



Gambar 6. *Long Short-Term Memory* (Smagulova dan James, 2020).

1. *Forget Gate*

   Gerbang pertama dalam LSTM disebut dengan *forget gate*. *Forget gate* adalah komponen pada model LSTM yang berfungsi untuk memutuskan apakah informasi dalam *cell state* perlu dihapus atau tidak dengan mengabaikan informasi yang tidak relevan dan tidak lagi diperlukan oleh sistem. Dengan demikian, LSTM mampu menyediakan informasi yang lengkap namun tetap relevan dengan kebutuhan terkini. *Forget gate* berfungsi untuk menghapus informasi yang tidak lagi diperlukan dalam *cell*. Caranya adalah dengan melakukan proses perhitungan yang di mana, proses perhitungan ini mengevaluasi *output* biner menggunakan data *output* dari waktu sebelumnya (h_{t-1}) dan *input* saat ini (x_t) dikalikan dengan matriks bobot (W_f) kemudian ditambahkan dengan nilai bias (b_f). Nilai yang diperoleh kemudian diproses melalui fungsi aktivasi dan menghasilkan *output* biner. Jika *output* bernilai 0, informasi dianggap tidak berguna dan dapat dihapus. Sebaliknya, jika *output* bernilai 1, informasi tersebut disimpan untuk digunakan. Persamaan untuk *forget gate* dirumuskan sebagai berikut pada persamaan (1).

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

Di mana:

f_t : *Forget gate*.

σ : Fungsi *sigmoid*.

W_f : Nilai *weight* untuk *forget gate*.

h_{t-1} : Nilai *output* sebelum urutan ke-t.

x_t : Nilai *input* pada urutan ke-t.

b_f : Nilai bias pada *forget gate*.

Selanjutnya adapun rumus nilai *weight* adalah sebagai berikut pada persamaan (2).

$W = \left(-\frac{1}{\sqrt{d}}, \frac{1}{\sqrt{d}}\right)$.....(2)

Di mana:

W : *Weight*.

d : Jumlah data.

2. *Input Gate*

Selanjutnya, terdapat gerbang kedua yang disebut *input gate*, yang bertugas memasukkan informasi penting untuk meningkatkan akurasi data. *Input gate* berfungsi menambahkan informasi yang telah diseleksi sebelumnya oleh *forget gate*. Gerbang ini tidak ada pada RNN, yang hanya memungkinkan satu *input* data untuk setiap *output* data. Dalam *input gate*, dikenal juga istilah *input modulation gate*, yang sering kali tidak disebutkan dalam beberapa ulasan tentang LSTM. Sesuai dengan namanya, *input modulation gate* berfungsi untuk memodulasi informasi yang ada, sehingga dapat mengurangi kecepatan konvergensi dari data *zero-mean*. *Input gate* menambahkan informasi penting ke *cell state*. Pertama, informasi diatur dengan fungsi *sigmoid* untuk menyaring nilai yang akan disimpan, mirip dengan proses pada *forget gate* yang menggunakan *input* h_{t-1} dan x_t. Setelah itu, vektor dibentuk menggunakan fungsi *tanh* yang menghasilkan output dari -1 hingga +1, yang mencakup semua kemungkinan nilai dari h_{t-1} dan x_t. Terakhir, nilai-nilai dalam vektor dan nilai-

nilai yang diatur dikalikan untuk mendapatkan informasi yang relevan. Persamaan *input gate* dirumuskan sebagai berikut pada persamaan (3).

$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i)$.....(3)

Di mana:

i_t : *Input gate*.

σ : Fungsi *sigmoid*.

W_i : Nilai *weight* untuk *input gate*.

h_{t-1} : Nilai *output* sebelum urutan ke-t.

x_t : Nilai *input* pada urutan ke-t.

b_i : Nilai bias pada *input gate*.

Adapun persamaan untuk kandidat baru pada *input gate* dirumuskan sebagai berikut pada persamaan (4).

$\tilde{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_C)$.....(4)

Di mana:

$\tilde{C}_t$: Nilai baru yang ditambahkan ke *cell state*.

$tanh$: Fungsi tangen hiperbolik.

W_C : Nilai *weight* untuk *cell state*.

h_{t-1} : Nilai *output* sebelum urutan ke-t.

x_t : Nilai *input* pada urutan ke-t.

b_C : Nilai bias pada *cell state*.

Setelah itu, *cell state* yang lama akan diperbarui menjadi *cell state* yang baru dengan mengalikan *state* lama dengan *forget gate* (f_t) untuk menghapus informasi yang telah dipilih oleh *forget gate*. Kemudian, nilai tersebut akan ditambahkan dengan $i_t * \tilde{C}_t$ yang merupakan nilai baru untuk memperbarui state, sehingga diperoleh persamaan *cell state* sebagai berikut pada persamaan (5).

$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$.....(5)

Di mana:

C_t : *Cell state*.

f_t : *Forget gate*.

C_{t-1} : *Cell state* sebelum urutan ke-t.

i_t : *Input gate*.

$\tilde{C}_t$: Nilai baru yang dapat ditambahkan ke *cell state*.

3. *Output Gate*

Gerbang terakhir adalah *output gate* yang berperan dalam menentukan dan menghasilkan informasi data yang komplet dan aktual yang berdasarkan *input* dan memori blok. Gerbang ini dapat menjadi tahap akhir untuk suatu informasi atau hanya menjadi bagian dari langkah awal, sebelum informasi diproses lebih lanjut melalui *input gate* di *cell* berikutnya. *Output gate* bertugas mengekstraksi informasi penting dari *cell state* saat ini untuk disajikan sebagai *output*. Pertama, sebuah vektor dibentuk dengan menerapkan fungsi *tanh* pada *cell*. Kemudian, informasi tersebut disesuaikan menggunakan fungsi *sigmoid*, yang menyaring nilai-nilai untuk disimpan menggunakan *input* h_{t-1} dan x_t. Terakhir, nilai vektor dan nilai yang telah diatur dikalikan untuk dikirim sebagai *output* dan *input* ke *cell* berikutnya. Persamaan *output gate* dirumuskan sebagai berikut pada persamaan (6).

$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \quad (6)$$

Di mana:

o_t : *Output gate*.

σ : Fungsi *sigmoid*.

W_o : Nilai *weight* untuk *output gate*.

h_{t-1} : Nilai *output* sebelum urutan ke-t.

x_t : Nilai *input* pada urutan ke-t.

b_o : Nilai bias pada *output gate*.

Setelah mendapatkan nilai dari *output gate*, *cell state* kemudian melewati *tanh* layer. Setelah itu, hasilnya dikalikan dengan *output gate* dan *sigmoid* layer. Persamaan untuk *output* pada urutan ke-t dirumuskan sebagai berikut pada persamaan (7).

$h_t = o_t * tanh(C_t)$.....(7)

Di mana:

h_t : Nilai *output* urutan ke-t.

o_t : *Output gate*.

$tanh$: Fungsi tangen hiperbolik.

C_t : *Cell state*.

2.2.9. *Bilingual Evaluation Understudy*

Dalam penerjemahan mesin biasanya dilakukan langkah evaluasi berupa cara untuk menilai apakah teks yang diterjemahkan oleh mesin setara dengan teks sumber yang diterjemahkan oleh manusia. Ada berbagai metrik evaluasi, seperti BLEU, METEOR, dan TER. Metrik-metrik ini digunakan untuk menilai segmen yang diterjemahkan oleh mesin berdasarkan kemiripannya dengan terjemahan referensi (Smartling, 2024). Menurut Papineni pada buku yang ditulis oleh (Wołk, 2019), *bilingual evaluation understudy* (BLEU) adalah sebuah metrik yang digunakan untuk mengevaluasi kualitas *output* dari *machine translation*. BLEU mengukur tingkat kemiripan antara terjemahan yang dihasilkan oleh mesin dan terjemahan yang dihasilkan oleh manusia. BLEU mengukur nilai dari 0 sampai 1, di mana 1 menunjukkan kemiripan yang sempurna antara terjemahan mesin dan terjemahan manusia.

Proses penggunaan BLEU dalam penelitian penerjemahan mesin biasanya melibatkan langkah-langkah berikut; membangun model penerjemahan mesin menggunakan berbagai teknik, seperti LSTM. Setelah model penerjemahan dibangun, peneliti mengevaluasi kinerjanya menggunakan data dari pasangan kalimat yang diterjemahkan secara manual oleh manusia. Sistem penerjemahan menerjemahkan kalimat-kalimat ini dan hasilnya dibandingkan dengan terjemahan manusia, BLEU menghitung presisi n-gram yang cocok antara terjemahan sistem dan referensi. Presisi ini kemudian diberi bobot berdasarkan panjang n-gram,

dengan n-gram yang lebih panjang memiliki bobot yang lebih tinggi. Semakin tinggi skor BLEU, semakin baik kualitas terjemahan sistem. Biasanya, skor BLEU rata-rata dihitung untuk keseluruhan *dataset* uji, memberikan indikasi umum tentang kinerja sistem terjemahan dalam menangani berbagai jenis kalimat. Skor BLEU digunakan untuk membandingkan kinerja model terjemahan dengan sistem lainnya atau dengan metode lain dalam penelitian yang sama. Analisis ini membantu peneliti dalam mengevaluasi keefektifan model terjemahan yang mereka kembangkan.

BLEU merupakan metrik evaluasi yang banyak digunakan karena relatif mudah dihitung dan dapat diterapkan pada berbagai macam pasangan bahasa. Namun, BLEU memiliki beberapa keterbatasan, seperti tidak memperhitungkan makna atau kefasihan, dan bias terhadap terjemahan yang menggunakan struktur bahasa dan kosakata yang mirip dengan terjemahan referensi (Santhosh, 2023). Terlepas dari keterbatasannya, BLEU tetap menjadi perangkat yang berharga untuk membandingkan berbagai sistem penerjemahan mesin dan melacak progresnya dari waktu ke waktu. BLEU memberikan metrik evaluasi yang lebih objektif daripada penilaian manusia karena menggunakan perbandingan statistik antara terjemahan sistem dan referensi manusia. Hal ini membantu peneliti untuk secara konsisten dan obyektif mengevaluasi kinerja model terjemahan. BLEU juga relatif mudah diimplementasikan dan diinterpretasikan. Peneliti dapat dengan cepat menghitung skor BLEU untuk setiap terjemahan dan memahami seberapa baik sistem mereka dalam menghasilkan terjemahan yang cocok dengan referensi manusia. Studi telah menunjukkan bahwa skor BLEU memiliki korelasi yang cukup baik dengan penilaian manusia terhadap kualitas terjemahan. Meskipun tidak sempurna, BLEU sering dianggap sebagai indikator yang dapat diandalkan tentang kualitas terjemahan sistem. BLEU telah menjadi standar *de facto* dalam penelitian *machine translation* dan telah digunakan secara luas dalam berbagai studi dan kompetisi terkait terjemahan mesin. Hal ini membuatnya menjadi pilihan yang umum dan mudah dibandingkan antara penelitian satu dengan lainnya.

2.2.10. *Softmax*

Fungsi aktivasi adalah komponen kunci dalam jaringan saraf tiruan yang mengubah *output* dari neuron menjadi bentuk yang dapat digunakan oleh lapisan berikutnya. Salah satu fungsi aktivasi yang paling sering digunakan dalam konteks klasifikasi adalah fungsi aktivasi *softmax*. Fungsi *softmax* umum digunakan tugas-tugas klasifikasi multi-kelas (Wathani dan Hidayati, 2023) seperti pada penelitian ini, di mana model harus memprediksi probabilitas distribusi dari kelas-kelas yang ada. Fungsi *softmax* adalah fungsi aktivasi yang menghasilkan distribusi probabilitas di antara beberapa kelas. Secara matematis, fungsi ini didefinisikan sebagai: $\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$. di mana z_i adalah *output* dari neuron ke-i dan K adalah jumlah total kelas. Fungsi ini memastikan bahwa jumlah dari semua *output* adalah satu, sehingga *output* dapat ditafsirkan sebagai probabilitas.

Keunggulan fungsi *softmax* terletak pada beberapa aspek penting, yaitu interpretasi probabilitas, normalisasi, dan diferensiasi yang mudah. *Output* dari fungsi *softmax* dapat diinterpretasikan sebagai probabilitas, memungkinkan penggunaan langsung dalam banyak algoritma pembelajaran mesin yang memerlukan *output* probabilitas. Fungsi ini juga menormalisasi *output* sehingga jumlahnya menjadi satu, yang membuatnya ideal untuk tugas klasifikasi multi-kelas.

Fungsi *softmax* sering digunakan dalam lapisan *output* dari jaringan saraf tiruan, terutama dalam model klasifikasi seperti *long short-term memory* (LSTM), *recurrent neural network* (RNN), *convolutional neural network* (CNN), dan model *transformer*. Misalnya pada penelitian penerjemahan bahasa, *softmax* digunakan untuk memilih kata berikutnya dalam *output* teks berdasarkan distribusi probabilitas yang dihasilkan oleh model.

2.2.11. *Adam*

Optimizer adalah komponen kunci dalam pelatihan model pembelajaran mesin dan jaringan saraf tiruan. Fungsinya adalah untuk memperbarui bobot model selama proses pelatihan dengan tujuan meminimalkan *loss function*. Dalam penelitian ini, *optimizer* berperan penting dalam melatih model *long short-term memory* (LSTM) yang digunakan untuk menerjemahkan bahasa Lampung dialek Api ke bahasa Indonesia. Dengan menggunakan *optimizer* yang tepat, model dapat belajar lebih efisien dan menghasilkan terjemahan yang lebih akurat. *Adam* adalah *optimizer* yang dapat menggantikan metode klasik *stochastic gradient descent* untuk pembaruan bobot secara iteratif berdasarkan data pelatihan. Metode ini merupakan gabungan antara *RMSprop* dan *stochastic gradient descent* dengan momentum. Diperkenalkan oleh (Kingma dan Ba, 2017) dalam makalah ICLR 2015 berjudul "*Adam: A Method for Stochastic Optimization*", *adam* adalah metode dengan laju pembelajaran adaptif yang menghitung laju pembelajaran secara individual untuk setiap parameter. Nama "*adam*" berasal dari singkatan "*adaptive moment estimation*" karena algoritma ini memanfaatkan estimasi gradien momen pertama dan kedua untuk menyesuaikan laju pembelajaran setiap bobot pada jaringan saraf. *Adam* adalah *optimizer* yang populer dalam *deep learning* karena mampu mencapai hasil yang baik dengan cepat. Dalam makalah aslinya, *adam* secara empiris menunjukkan bahwa konvergensinya sesuai dengan harapan analisis teoritis.

2.2.12. *Categorical Cross-Entropy*

Fungsi *loss cross-entropy* digunakan untuk menyesuaikan bobot model selama pelatihan, dengan tujuan untuk meminimalkan *loss*. Semakin kecil *loss*, semakin baik kinerja model. Fungsi *loss* mengukur seberapa jauh prediksi model dari yang benar. Fungsi *loss* tidak hanya memberikan gambaran statis tentang kinerja model, tetapi juga menjadi dasar untuk menilai seberapa akurat algoritma dalam mencocokkan data. Sebagian besar algoritma pembelajaran mesin menggunakan

fungsi *loss* selama fase optimasi, yang melibatkan pemilihan parameter optimal untuk data.

Categorical cross-entropy dikenal juga sebagai *softmax loss*. Fungsi *loss categorical cross-entropy* adalah kombinasi antara aktivasi *softmax* dan *loss cross-entropy* yang digunakan untuk klasifikasi multi-kelas. Dengan *loss* ini, kita dapat melatih *convolutional neural network* untuk menghasilkan probabilitas untuk setiap kelas N pada setiap gambar.

Dalam klasifikasi multi-kelas, *output* mentah dari jaringan saraf diproses melalui aktivasi *softmax*, yang kemudian menghasilkan vektor probabilitas prediksi untuk kelas-kelas *input*.

Dalam kasus khusus (dan umumnya) klasifikasi multi-kelas, labelnya adalah *one-hot*, sehingga hanya kelas positif yang memiliki *term* dalam *loss*. Hanya ada satu elemen dalam vektor target yang berbeda dari nol. Dengan mengabaikan elemen-elemen penjumlahan yang bernilai nol karena label target, kita dapat menuliskan: $CE = -log\left(\frac{e^{s_p}}{\Sigma_j^c e^{s_j}}\right)$. (Shah, 2023)

2.2.13. *Overfitting*

Overfitting adalah masalah mendasar dalam *supervised machine learning* yang mencegah kita menggeneralisasi model secara sempurna agar sesuai dengan data yang diamati pada data pelatihan, serta data yang tidak terlihat pada set pengujian. (Ying, 2019). *Overfitting* merupakan masalah umum dalam pembelajaran mesin di mana model mempelajari detail dan kebisingan dalam data pelatihan hingga mempengaruhi performa pada data baru atau data uji, sehingga model menunjukkan akurasi tinggi pada data pelatihan namun berkinerja buruk pada data yang belum

pernah dilihat sebelumnya. Hal ini sering terjadi ketika model terlalu kompleks, memiliki terlalu banyak parameter, atau dilatih pada data pelatihan yang terbatas. Model yang terlalu rumit atau pelatihan yang terlalu lama dapat menyebabkan model mempelajari informasi yang tidak relevan dari data pelatihan, sedangkan data yang mengandung kebisingan atau jumlah yang tidak memadai juga berkontribusi pada masalah ini. Untuk mencegah *overfitting*, beberapa teknik dapat diterapkan, seperti *cross-validation* untuk memastikan kinerja model yang konsisten, regularisasi untuk mengurangi kompleksitas model, serta *dropout* dalam jaringan saraf untuk mencegah ketergantungan berlebihan pada neuron tertentu. Mengumpulkan lebih banyak data pelatihan dan menggunakan *earlystopping* untuk menghentikan pelatihan berdasarkan performa pada data validasi juga merupakan metode yang efektif. Dalam berbagai aplikasi seperti pengenalan gambar atau penerjemahan bahasa, *overfitting* dapat diatasi dengan teknik-teknik ini untuk meningkatkan kemampuan model dalam menggeneralisasi pada data baru, sehingga menghasilkan model yang lebih andal dan efektif.

2.2.14. Google Drive

Google Drive adalah layanan penyimpanan *file* secara *online* yang dikembangkan oleh Google. Layanan ini memungkinkan pengguna untuk menyimpan, mengunduh, dan berbagi *file* secara *online*. Google Drive juga memungkinkan pengguna untuk membuat folder, membuat *file*, dan mengedit *file* secara *online*. Google Drive akan digunakan untuk di-*mounting* ke Google Colab.

Mounting Google Drive ke Google Colab adalah proses yang memungkinkan pengguna untuk mengakses *file* dan folder di Google Drive dari dalam Google Colab. Proses ini memungkinkan pengguna untuk mengunduh *file* dan folder dari Google Drive ke Google Colab dan mengeditnya secara *online*. Untuk *mounting* Google Drive ke Google Colab, pengguna harus terlebih dahulu menghubungkan akun Google Drive dengan Google Colab. Pengguna dapat melakukan ini dengan

mengeklik tombol "*Connect to Google Drive*" di Google Colab dan memasukkan *authorization* yang diperlukan.

Setelah pengguna menghubungkan akun Google Drive dengan Google Colab, pengguna dapat mengakses folder dan *file* di Google Drive dari dalam Google Colab. Pengguna dapat mengunduh *file* dan folder dari Google Drive ke Google Colab dan mengeditnya secara *online*. *Mounting* Google Drive ke Google Colab memungkinkan pengguna untuk mengakses *file* dan folder di Google Drive dari dalam Google Colab dan mengeditnya secara *online*. Proses ini memungkinkan pengguna untuk mengunduh *file* dan folder dari Google Drive ke Google Colab dan mengeditnya secara *online*. (Rahmadya, 2020)

Gambar 7. Logo Google Drive.

2.2.15. Google Colaboratory

Google Colaboratory, atau Colab adalah produk dari Google Research. Colab adalah *environment* Jupyter Notebook gratis yang disediakan oleh Google di mana kita bisa menggunakan GPU dan TPU gratis yang bisa menyelesaikan masalah *data science*. Colab memiliki hampir semua modul yang dibutuhkan untuk analisis *data science*. Alat-alat ini termasuk tetapi tidak terbatas pada Numpy, Scipy, Pandas, dll. Bahkan kerangka kerja *deep learning*, seperti Tensorflow, Keras, dan Pytorch juga disertakan. Colab memungkinkan siapa saja untuk menulis dan mengeksekusi kode python melalui *browser*, dan sangat cocok untuk *machine learning* dan *data analysis*. Colab adalah layanan *cloud* gratis yang dimiliki oleh Google untuk mendorong penelitian *machine learning* dan *artificial intelligence*, di mana sering kali menjadi halangan adalah daya komputasi yang besar. Sedangkan untuk komputer yang memiliki GPU atau komputer yang bagus, pembuatan *local environment* dengan anaconda dan penginstalan paket serta menyelesaikan masalah

instalasi adalah hal yang bisa merepotkan (Naik, 2023), di mana *ease of use* dari Colab bisa menjadi sebuah keunggulan.

Gambar 8. Logo Google Colab.

2.2.16. Python

Python dikembangkan oleh Guido von Rossum pada tahun 1990-an di CWI, Amsterdam sebagai kelanjutan dari bahasa pemrograman ABC. Python merupakan salah satu bahasa pemrograman dinamis, Python telah digunakan oleh banyak *programmer* untuk mengembangkan berbagai macam sistem (Huda dan Ardi, 2020). Keunggulan Python dalam *machine learning* adalah, Python adalah bahasa yang memiliki banyak kemampuan (*powerful*) namun mudah digunakan (*accessible).* Python telah menjadi bahasa pemrograman terpopuler untuk *data science* karena Python mengizinkan kita untuk melupakan bagian-bagian yang membosankan dari pemrograman dan menawarkan kita *environment* dimana kita bisa langsung menjalankan ide-ide dan konsep-konsep yang kita miliki (Raschka dan Mirjalili, 2019).

Gambar 9. Logo Python.

2.2.17. NumPy

NumPy adalah pustaka inti yang digunakan dalam banyak perpustakaan Python lainnya. NumPy menyediakan struktur data dan fungsi berkinerja tinggi yang tidak tersedia dalam paket dasar Python. Pustaka ini mendefinisikan struktur data khusus yang disebut *ndarray*, yaitu larik N-dimensi. Pengetahuan tentang NumPy sangat penting untuk perhitungan numerik karena penggunaan yang tepat dapat meningkatkan kinerja komputasi secara signifikan. (Nelli, 2023)

Gambar 10. Logo NumPy.

2.2.18. Keras

Keras adalah API jaringan saraf tingkat tinggi yang dapat berjalan di atas beberapa kerangka kerja tingkat rendah termasuk TensorFlow, CNTK, atau Theano. API ini dikembangkan dengan fokus untuk memungkinkan eksperimen yang cepat dan menyediakan API yang konsisten dan sederhana yang meminimalkan jumlah tindakan pengguna yang diperlukan untuk kasus penggunaan umum, menawarkan umpan balik yang jelas dan dapat ditindaklanjuti setelah kesalahan pengguna. Keras dirancang agar mudah digunakan dan sangat cocok untuk pemula dan ahli. Ia mampu berjalan di berbagai perangkat, termasuk CPU, Xeon Phi, Google TPU, dan perangkat seperti GPU atau GPU yang mendukung OpenCL. Singkatnya, Keras adalah kerangka kerja *deep learning* yang kuat dan serbaguna yang menyediakan API yang sederhana dan konsisten untuk membangun dan melatih jaringan saraf. Kemampuannya untuk berjalan di beberapa kerangka kerja tingkat rendah dan dukungannya untuk berbagai perangkat membuatnya menjadi pilihan populer di kalangan peneliti dan pengembang. (Rorro, 2019)

Gambar 11. Logo Keras.

2.2.19. NLTK

Natural Language Toolkit (NLTK) adalah pustaka Python sumber terbuka yang digunakan secara luas untuk pemrosesan bahasa alami (NLP). *Library* ini dirancang untuk memfasilitasi studi, pemrosesan, dan analisis data bahasa alami. NLTK sangat berguna untuk tugas-tugas yang melibatkan pemrosesan dan analisis data teks dalam jumlah besar, seperti analisis sentimen, pengenalan entitas bernama, dan pemodelan topik. NLTK menawarkan berbagai fungsi, termasuk dukungan untuk berbagai korpora dan pembaca korpora, yang memungkinkan pengguna untuk dengan mudah mengakses dan memanipulasi koleksi data teks yang besar. NLTK banyak digunakan di berbagai bidang, termasuk linguistik, ilmu komputer, dan teknologi informasi, dan sangat populer di kalangan peneliti dan pengembang yang mengerjakan proyek NLP. NLTK tidak lebih dari sebuah pustaka Python yang di dalamnya terdapat banyak alat yang dikhususkan untuk pemrosesan dan analisis data teks. NLTK dibuat pada tahun 2001 untuk tujuan pendidikan, kemudian seiring berjalannya waktu dikembangkan sedemikian rupa sehingga menjadi alat analisis yang nyata. Di dalam perpustakaan NLTK, ada juga koleksi besar contoh teks, yang disebut korpora. Koleksi teks ini sebagian besar diambil dari literatur dan sangat berguna sebagai dasar untuk penerapan teknik-teknik yang dikembangkan dengan perpustakaan NLTK. (Nelli, 2023)

2.2.20. Pandas

Pandas adalah salah satu pustaka utama untuk analisis data dalam Python. Konsep dasar dari paket ini adalah *DataFrame*, sebuah struktur data tabular dua dimensi

dengan label baris dan kolom. *DataFrame* memudahkan manipulasi data seperti yang biasanya dilakukan dalam *spreadsheet* atau *database* relasional (database SQL). Pandas mengadopsi kemampuan kinerja tinggi dari pustaka NumPy untuk mempercepat operasi data. Dengan menggunakan pengindeksan yang canggih, maka akan mudah untuk melakukan banyak operasi pada struktur data seperti pembentukan ulang, pengirisan, penggabungan, dan pemilihan *subset* data. (Nelli, 2023)

Penggunaan Pandas sangat relevan karena alat ini memungkinkan peneliti untuk mengelola dan menganalisis data penelitian dengan lebih efektif. Misalnya, untuk proyek yang melibatkan analisis data besar atau pemrosesan data kompleks, Pandas dapat mengurangi waktu yang dibutuhkan untuk membersihkan dan memanipulasi data. Hal ini dapat mempercepat tahap pra-pemrosesan dan analisis data, yang meningkatkan efisiensi keseluruhan dari penelitian. Pengetahuan yang baik tentang Pandas tidak hanya meningkatkan kinerja komputasi tetapi juga memberikan fleksibilitas dalam mengolah data sesuai kebutuhan penelitian.

Gambar 12. Logo Pandas.

2.2.21. Matplotlib

“Matplotlib adalah sebuah pustaka *plotting* yang kuat dalam Python yang digunakan untuk membuat visualisasi statis, animasi, dan interaktif. Matplotlib dirancang untuk menyediakan alat dan fungsionalitas bagi pengguna untuk merepresentasikan data secara grafis, membuatnya lebih mudah untuk dianalisis dan dipahami. Matplotlib pada awalnya dikembangkan oleh John D. Hunter pada tahun 2003 dan sekarang dikelola oleh komunitas pengembang yang besar. Matplotlib dikenal karena keserbagunaannya, menawarkan berbagai macam plot

termasuk plot garis, plot sebar, plot batang, histogram, diagram lingkaran, dan banyak lagi. Ini juga menyediakan opsi penyesuaian yang luas untuk mengontrol setiap aspek plot, seperti gaya garis, warna, penanda, label, dan anotasi. Selain itu, Matplotlib terintegrasi secara mulus dengan NumPy, membuatnya mudah untuk melakukan plot *array* data secara langsung. Matplotlib banyak digunakan dalam komunitas ilmiah, teknik, dan ilmu data untuk eksplorasi, analisis, dan presentasi data karena fleksibilitas, keserbagunaan, dan kemampuannya yang luas.” (Kattamuri, 2018)

Gambar 13. Logo Matplotlib.

III. METODOLOGI PENELITIAN

3.1. Waktu dan Tempat Penelitian

Penelitian dilakukan pada semester genap 2022/2023 di Gedung Ilmu Komputer, Jurusan Ilmu Komputer, Fakultas Matematika dan Ilmu Pengetahuan Alam, Universitas Lampung yang beralamatkan di Jalan Prof. Dr. Ir. Sumantri Brojonegoro No.1, Gedong Meneng, Kec. Rajabasa, Kota Bandar Lampung, Lampung dan di kediaman penulis di Bandar Lampung yaitu di Pengajaran, Kecamatan Teluk Betung Utara, Kota Bandar Lampung, Lampung.

3.2. Perangkat Penelitian

Penelitian ini menggunakan kombinasi antara perangkat keras (*hardware*) dan perangkat lunak (*software*), di antaranya adalah sebagai berikut.

3.2.1. Perangkat Keras

Dalam penelitian ini perangkat keras yang digunakan adalah PC dengan spesifikasi sebagai berikut:

- *Processor* : Intel(R) Core(TM) i5-4460 CPU @ 3,20GHz
- RAM : 16,0 GB DDR3 1600 MHz
- GPU : PowerColor TurboDuo R9 270 OC

- Penyimpanan : SSD 500 GB

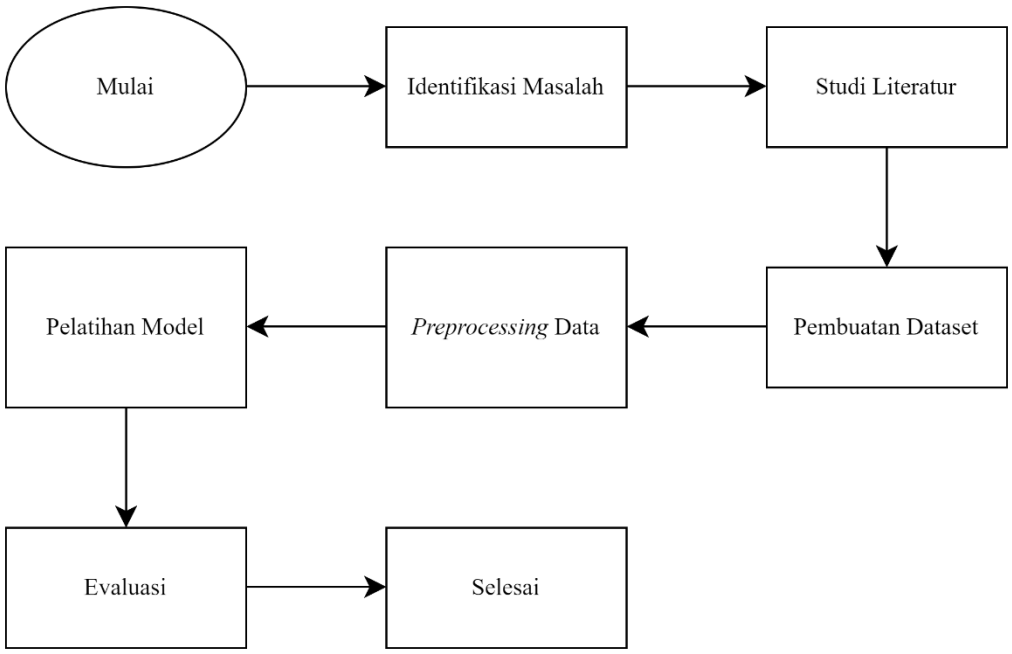
3.2.2. Perangkat Lunak

Dalam penelitian ini perangkat lunak yang digunakan adalah sebagai berikut:

- Windows 10
- Google Drive
- Google Colab Pro
- Python
- NumPy, Keras, NLTK, Pandas, Matplotlib

3.3. Alur Penelitian

Alur penelitian ini adalah sebagai berikut:



Gambar 14. Alur Penelitian.

Gambar di atas menjelaskan tahapan-tahapan yang dilakukan dalam penelitian ini. Penjelasan mengenai tahapan penelitian ini adalah sebagai berikut:

3.3.1. Identifikasi Masalah

Bahasa Lampung merupakan salah satu bahasa daerah di Indonesia yang menghadapi risiko kepunahan karena semakin jarangnya penggunaan dalam komunikasi sehari-hari. Faktor-faktor seperti urbanisasi, migrasi, dan dominasi bahasa Indonesia sebagai bahasa nasional telah menyebabkan penurunan signifikan dalam jumlah penutur bahasa Lampung. Keadaan ini diperburuk oleh kurangnya dukungan teknologi yang memadai untuk memfasilitasi penggunaan dan pelestarian bahasa tersebut.

Seiring berjalannya waktu, semakin sedikit generasi muda yang mampu berbicara atau memahami bahasa Lampung, sehingga mempercepat proses kepunahan. Upaya pelestarian bahasa ini memerlukan pendekatan yang inovatif, salah satunya adalah dengan mengembangkan teknologi penerjemahan otomatis yang mampu mendukung penggunaan bahasa Lampung dalam berbagai konteks, termasuk pendidikan dan budaya.

Berdasarkan analisis permasalahan yang ada, pada tahapan ini penelitian yang akan dilakukan adalah bagaimana cara membuat sebuah mesin penerjemahan berbasis *long short-term memory* yang dapat menerjemahkan bahasa Lampung ke bahasa Indonesia. Hasil dari penelitian ini diharapkan dapat diciptakannya alat yang membantu preservasi bahasa Lampung dan dapat turut membantu bahasa Lampung dapat lebih mudah dimengerti oleh masyarakat yang lebih luas.

3.3.2. Studi Literatur

Langkah kedua dalam penelitian ini adalah melakukan studi literatur, di mana peneliti mengumpulkan bahan bacaan yang relevan dengan masalah yang akan diteliti. Tujuan dari langkah ini adalah untuk memahami secara mendalam penelitian sebelumnya yang telah dilakukan, baik yang berkaitan dengan topik

maupun metode penelitian. Informasi dari teori dan penelitian sebelumnya sangat penting karena menjadi landasan untuk menganalisis dengan tepat sesuai dengan kerangka berpikir ilmiah yang ditetapkan. Dalam konteks penelitian, telah dilakukan banyak penelitian mengenai topik penerjemahan mesin dan juga metode penerjemahannya. Data tentang penelitian sebelumnya akan dimasukkan ke dalam tabel sebagai bagian dari studi literatur yang relevan dengan penelitian ini. Hal ini bertujuan untuk memberikan pemahaman yang lebih menyeluruh dan komprehensif terhadap topik yang sedang diteliti.

3.3.3. Pembuatan *Dataset*

3.3.3.1. Pengumpulan Korpus Bahasa Indonesia

Dalam langkah pengumpulan data, langkah awal yang dilakukan adalah mencari berbagai kumpulan teks yang tersedia dalam bahasa Indonesia. Setelah itu, teks-teks tersebut akan diterjemahkan ke dalam bahasa Lampung oleh peneliti. Langkah ini diambil untuk memastikan bahwa data yang diperoleh dapat mencakup beragam konteks dan varian bahasa yang relevan dengan penelitian. Dengan demikian, akan tercipta basis data yang representatif dan komprehensif untuk analisis lebih lanjut dalam konteks bahasa Lampung.

3.3.3.2. Penerjemahan Manusia untuk Pembentukan Korpus Paralel

Pada langkah penerjemahan manusia, peneliti akan melakukan proses penerjemahan secara manual dari bahasa Indonesia yang selanjutnya akan diterjemahkan ke bahasa Lampung, hal ini dilakukan guna membuat data korpus paralel yang akan menjadi dasar utama dalam penelitian ini. Data kata-kata berbahasa Lampung dialek api dikumpulkan dari berbagai sumber seperti kamus, cerita rakyat, dan teks sastra. Data tersebut digunakan penelitian ini untuk melatih dan menguji model LSTM yang akan menerjemahkan bahasa Lampung ke bahasa

Indonesia secara otomatis. Dengan memperoleh data yang berkualitas dan representatif dari proses penerjemahan manual ini, diharapkan model dapat belajar dengan baik dan menghasilkan hasil terjemahan yang akurat dan konsisten dalam konteks bahasa Lampung.

Tabel 2. Contoh Data Korpus Paralel

Terus jalan lurus hingga tiba di gereja tersebut.	Teghus lapah lughus hingga tigoh di gegheja tesebut.
Apa kamu pernah melihat laba-laba membuat sarangnya?	Api niku peghnak ngeliyak lawah ngeguwai saghangni?
Dia tidak memiliki kemampuan untuk menjadi presiden.	Ia mak ngedok kemampuan guwai jadi pghesiden.

3.3.4. *Preprocessing* Data

Preprocessing data adalah langkah penting dalam penelitian ini untuk memastikan bahwa data yang digunakan untuk melatih dan mengevaluasi model LSTM bersih, konsisten, dan dalam format yang sesuai. Langkah-langkah *preprocessing* yang dilakukan dalam penelitian ini meliputi pembagian *dataset*, normalisasi teks, dan *tokenisasi* teks. Berikut ini adalah penjelasan langkah-langkah tersebut:

3.3.4.1. Pembagian *Dataset*

Dataset dibagi menjadi dua bagian utama yaitu data latih dan data uji. Pembagian ini penting untuk memastikan bahwa model dapat dilatih dan dievaluasi dengan benar. Proporsi pembagian yang digunakan adalah sebagai berikut:

1. **Data Latih (*Training Set*)**: 80% dari total data digunakan untuk melatih model. Data ini digunakan untuk memperbarui bobot model melalui proses pelatihan.
2. **Data Uji (*Test Set*)**: 20% dari total data digunakan untuk evaluasi akhir model. Data ini hanya digunakan sekali setelah model dilatih untuk memberikan estimasi kinerja model pada data yang benar-benar baru.

3.3.4.2. Normalisasi Teks

Normalisasi teks adalah proses untuk menghilangkan variasi dalam teks yang tidak relevan untuk tugas penerjemahan. Langkah-langkah normalisasi yang dilakukan meliputi:

1. ***Lowercasing***: Mengubah semua huruf dalam teks menjadi huruf kecil untuk menghilangkan perbedaan antara huruf besar dan kecil. Misalnya, kata "Lampung" dan "lampung" akan dianggap sama.
2. **Penghapusan Karakter Khusus**: Menghapus karakter khusus seperti tanda baca (misalnya, titik, koma, tanda tanya) dan simbol yang tidak diperlukan. Hal ini dilakukan untuk menghindari gangguan dalam proses *tokenisasi*.
3. **Penghapusan Spasi Berlebih**: Menghapus spasi tambahan yang tidak diperlukan, baik di awal, di akhir, maupun di antara kata-kata.

Contoh hasil normalisasi teks:

- Sebelum: "Lampung adalah sebuah provinsi, di Sumatera. "
- Sesudah: "lampung adalah sebuah provinsi di sumatera"

3.3.4.3. *Tokenisasi*

Tokenisasi adalah proses memecah teks menjadi unit-unit kecil yang disebut *token*. Dalam konteks penerjemahan mesin, *token* bisa berupa kata, karakter, atau sub-kata tergantung pada pendekatan yang digunakan. Setiap kalimat dipecah menjadi deretan kata. Pemecahan ini biasanya dilakukan berdasarkan spasi dan tanda baca.

Contoh hasil *tokenisasi* teks:

- Sebelum: "lampung adalah sebuah provinsi di sumatera"
- Sesudah: ["lampung", "adalah", "sebuah", "provinsi", "di", "sumatera"]

Langkah-langkah *preprocessing* data ini esensial untuk memastikan bahwa data yang digunakan dalam penelitian ini berada dalam kondisi yang optimal untuk pelatihan dan evaluasi model RNN. Dengan melakukan pembagian *dataset*, normalisasi teks, dan *tokenisasi* teks secara sistematis, model dapat dilatih dengan data yang bersih dan konsisten, yang diharapkan akan meningkatkan performa dan akurasi penerjemahan bahasa Lampung ke bahasa Indonesia.

3.3.4.4. *Padding Sequences*

Padding sequences adalah langkah untuk memastikan bahwa semua urutan data memiliki panjang yang sama. Dalam banyak model, *input* harus memiliki ukuran yang konsisten. *Padding* dilakukan dengan menambahkan nilai khusus (biasanya nol) ke awal (*pre*) atau akhir (*post*) urutan yang lebih pendek sehingga semua urutan memiliki panjang yang seragam. Hal ini memastikan bahwa semua *input* memiliki dimensi yang sama, memudahkan pemrosesan dalam *batch*, dan memungkinkan model untuk menangani urutan dengan panjang yang bervariasi.

3.3.4.5. *One-Hot Encoding*

One-hot encoding adalah teknik untuk mengubah label kategori menjadi format numerik yang dapat diproses oleh model. Dalam *one-hot encoding* setiap label kategori diubah menjadi vektor biner di mana hanya satu elemen yang bernilai 1 (aktif), dan semua elemen lainnya bernilai 0 (non-aktif). Misalnya, jika ada tiga kategori, label "kategori 1" akan diwakili sebagai [1, 0, 0], label "kategori 2" sebagai [0, 1, 0], dan label "kategori 3" sebagai [0, 0, 1]. Teknik ini memungkinkan model untuk memproses setiap kategori secara terpisah tanpa mengasumsikan adanya hubungan ordinal antara kategori-kategori tersebut.

3.3.5. Pelatihan Model

3.3.5.1. Desain Model

Model yang digunakan dalam penelitian ini adalah model *seq2seq* dengan *long short-term memory. Sequence to sequence* (seq2seq) adalah arsitektur model yang dirancang untuk tugas-tugas di mana data *input* dan *output* berbentuk urutan, seperti dalam terjemahan bahasa, pemrosesan bahasa alami, atau sintesis ucapan. Model seq2seq terdiri dari dua bagian utama: *encoder* dan *decoder*. *Encoder* bertugas untuk memproses urutan *input* dan mengubahnya menjadi representasi vektor tetap yang dikenal sebagai konteks atau *state*. *Decoder* kemudian menggunakan representasi ini untuk menghasilkan urutan *output*. Arsitektur ini memungkinkan model untuk menangani *input* dan *output* dengan panjang yang berbeda secara efektif. Sedangkan, *long short-term nemory* (LSTM) adalah jenis jaringan saraf berulang (*recurrent neural network*, RNN) yang dirancang untuk mengatasi masalah *vanishing gradient* yang sering terjadi pada RNN tradisional. LSTM memiliki unit penyimpanan yang dikenal sebagai sel memori, yang memungkinkan model untuk mempertahankan informasi jangka panjang. Setiap sel memori dalam LSTM terdiri dari beberapa komponen: *input gate*, *forget gate*, dan *output gate*, yang mengontrol aliran informasi masuk dan keluar dari sel memori. Dengan demikian, LSTM dapat mengingat informasi penting untuk periode waktu yang lebih lama dan lebih efektif dalam memproses urutan data yang kompleks dan panjang. Model ini menggunakan beberapa *layer*:

1. ***Embedding Layer***: *Layer* ini mengubah kata-kata yang direpresentasikan sebagai indeks integer menjadi vektor berdimensi tetap. Fungsi utamanya adalah untuk menangkap makna semantik kata-kata. Selama proses pelatihan, vektor ini dipelajari dan disesuaikan untuk mencerminkan hubungan antar kata dalam ruang vektor berdimensi lebih rendah. Dengan demikian, *layer* ini membantu model memahami dan mengolah kata-kata berdasarkan konteks mereka.

2. **LSTM *Layer***: *Layer* ini adalah komponen kunci dari model yang dirancang . LSTM menggunakan sel memori dengan tiga gerbang utama *input gate*, *forget gate*, dan *output gate* untuk mengontrol aliran informasi. *Input gate* menentukan berapa banyak informasi baru yang akan ditambahkan ke sel memori, *forget gate* memutuskan informasi mana yang harus dihapus, dan *output gate* mengatur informasi mana yang akan diteruskan ke langkah berikutnya. Dengan cara ini, LSTM mampu mempertahankan informasi penting dalam urutan data yang panjang dan kompleks.

3. ***Dropout Layer***: *Layer* ini berfungsi untuk mengurangi *overfitting* dengan secara acak menonaktifkan beberapa unit selama pelatihan. Setiap *epoch*, neuron-neuron tertentu dipilih secara acak untuk dinonaktifkan dengan probabilitas tertentu, yang memaksa model untuk tidak bergantung pada neuron tertentu dan meningkatkan kemampuan generalisasi. Selama inferensi, *dropout* tidak diterapkan, dan semua neuron aktif untuk menghasilkan prediksi akhir.

4. ***RepeatVector Layer***: *Layer* ini berfungsi sebagai adaptor dengan mengubah *context vectors* yang dihasilkan oleh *encoder* menjadi urutan dengan panjang yang diperlukan oleh *decoder*. Setelah *encoder* menghasilkan vektor konteks berdimensi tertentu, *RepeatVector* menggandakannya menjadi urutan dengan panjang yang sesuai. Setiap elemen dalam urutan tersebut adalah salinan dari *context vector*, memungkinkan *decoder* untuk memproses informasi ini di setiap langkah waktu dalam urutan *output*. Ini memungkinkan *decoder* untuk menggunakan representasi tetap dari urutan *input* untuk menghasilkan *output* yang sesuai.

5. ***TimeDistributed Dense Layer* (Fungsi Aktivasi *Softmax*)**: *TimeDistributed dense layer* membungkus *layer dense* dan menerapkannya pada setiap *timestep* dari urutan *input* secara terpisah. Ini memungkinkan model untuk memproses setiap elemen dalam urutan secara individual, yang sangat berguna untuk tugas-tugas sekuensial seperti terjemahan bahasa. Setelah *layer dense* menerapkan transformasi pada setiap *timestep*, fungsi *softmax* digunakan pada *output* dari *layer dense* untuk mengubah nilai *logit* menjadi probabilitas. Fungsi *softmax* mengonversi *output* menjadi distribusi probabilitas di mana setiap elemen

menunjukkan seberapa besar kemungkinan setiap kata dalam kosakata adalah bagian dari *output* pada *timestep* tertentu. Dengan cara ini, model dapat memilih kata dengan probabilitas tertinggi pada setiap langkah waktu dalam urutan *output*, memungkinkan pembuatan prediksi kata yang akurat dan sesuai.

Implementasi model dilakukan menggunakan *library* Keras. Model dilatih menggunakan parameter sebagai berikut:

Tabel 3. Pengaturan Parameter

No	Parameter	Nilai
1	Unit	1024
2	*Dropout*	0,8
3	Fungsi Aktivasi	*Softmax*
4	*Optimizer*	*Adam*
5	Fungsi *Loss*	*Categorical Cross-Entropy*
6	*Epoch*	50
7	Ukuran *Batch*	64
8	Persentase *Validation*	10

3.3.6. Evaluasi

3.3.6.1. *Loss*

Salah satu cara untuk mengevaluasi performa model pembelajaran mesin, khususnya pada penelitian ini dalam tugas penerjemahan mesin, adalah dengan menggunakan grafik *loss*. Grafik *loss* memberikan visualisasi bagaimana model belajar selama proses pelatihan dan seberapa baik model dapat menggeneralisasi pengetahuan yang diperolehnya pada data yang tidak terlihat (data validasi). Grafik *loss* menampilkan dua garis utama: *loss* pada data pelatihan (*training loss*) dan *loss* pada data validasi (*validation loss*). *Loss* dalam konteks ini mengacu pada kesalahan yang dibuat oleh model ketika mencoba memprediksi keluaran yang diinginkan.

1. ***Training Loss:*** *Training loss* menggambarkan tingkat kesalahan model saat mencoba memprediksi hasil dari data pelatihan. Selama pelatihan, kita

mengharapkan *training loss* akan menurun secara bertahap. Penurunan ini menunjukkan bahwa model semakin mampu memahami pola yang ada dalam data pelatihan dan mengurangi kesalahan prediksinya. Grafik yang menunjukkan penurunan yang konsisten pada *training loss* menunjukkan bahwa model sedang belajar dengan baik dari data pelatihan.

2. ***Validation Loss:*** *Validation loss* mengukur kesalahan model pada data validasi, yaitu data yang tidak digunakan dalam pelatihan. Tujuannya adalah untuk mengevaluasi kemampuan model dalam menggeneralisasi pola yang dipelajari dari data pelatihan ke data yang tidak pernah dilihat sebelumnya. Idealnya, *validation loss* juga akan menurun seiring dengan *training loss*. Penurunan ini menandakan bahwa model tidak hanya belajar dari data pelatihan tetapi juga mampu membuat prediksi yang baik pada data baru.
3. **Interpretasi Grafik *Loss*:** Ketika kedua grafik *loss* (*training* dan *validation*) menunjukkan penurunan yang serupa dan cenderung konvergen (mendekati nilai yang hampir sama), ini menunjukkan bahwa model tidak hanya cocok dengan data pelatihan tetapi juga mampu menggeneralisasi dengan baik pada data validasi. Sedangkan, *overfitting* terjadi ketika *training loss* terus menurun, tetapi *validation loss* mulai meningkat atau stagnan. Ini menunjukkan bahwa model menjadi terlalu spesifik terhadap data pelatihan dan kehilangan kemampuannya untuk menggeneralisasi pada data baru. Lalu *underfitting* terjadi jika kedua grafik *loss* tetap tinggi atau hanya menurun sangat lambat, ini menandakan bahwa model belum cukup kompleks untuk menangkap pola yang ada dalam data, atau mungkin model tidak belajar dengan baik dari data pelatihan.

Dalam penelitian ini, grafik *loss* digunakan sebagai alat utama untuk memantau kinerja model selama proses pelatihan. Melalui analisis grafik *loss*, penyesuaian pada parameter pelatihan dapat dilakukan untuk meminimalkan *overfitting* dan mencapai performa terbaik dari model.

3.3.6.2. Hasil Penerjemahan

Pada tahap ini, penelitian ini menyajikan hasil penerjemahan teks bahasa Lampung ke dalam bahasa Indonesia yang dihasilkan oleh model. Untuk memberikan gambaran yang jelas dan komprehensif mengenai performa model, hasil terjemahan akan dibandingkan langsung dengan pasangan teks asli dari korpus paralel bahasa Lampung–bahasa Indonesia. Hasil penerjemahan yang disajikan dalam penelitian ini meliputi contoh-contoh kalimat atau teks yang dipilih secara acak dari korpus uji. Setiap contoh akan disajikan dalam bentuk tabel yang memudahkan perbandingan antara teks bahasa Lampung dari data korpus paralel, teks bahasa Indonesia dari data korpus paralel, dan teks terjemahan otomatis bahasa Indonesia yang dihasilkan oleh model. Penyajian ini memberikan kesempatan bagi pembaca untuk melihat secara langsung kualitas hasil terjemahan yang dihasilkan oleh model, serta membandingkannya dengan terjemahan referensi yang benar. Dengan demikian, pembaca dapat mengevaluasi akurasi dan kejelasan dari hasil terjemahan secara lebih mendalam.

3.3.6.3. *Bilingual Evaluation Understudy*

Dalam penelitian ini, digunakan metode evaluasi *bilingual evaluation understudy* (BLEU) untuk menilai sejauh mana hasil terjemahan mesin mendekati kualitas terjemahan yang dihasilkan oleh manusia. Metode BLEU mengevaluasi terjemahan dengan membandingkan kesamaan n-gram antara hasil terjemahan mesin dan terjemahan referensi yang dibuat oleh penerjemah manusia. Semakin banyak n-gram yang cocok, semakin baik kinerja model dalam menghasilkan terjemahan yang berkualitas. Berikut ini adalah cara melakukan perhitungan BLEU:

Skor BLEU dihitung dengan menggabungkan *precision n-gram* dan *brevity penalty*. *Precision n-gram* untuk ukuran n tertentu dihitung dengan rumus berikut pada persamaan (8).

$$\text{Precision}_n = \frac{\text{Jumlah n-gram yang cocok}}{\text{Jumlah n-gram dalam terjemahan mesin}} \quad \text{.....(8)}$$

Selanjutnya, (Papineni dkk., 2002) mengembangkan metode *modified precision n-gram* untuk mengatasi masalah *over-counting* dalam evaluasi terjemahan. Dalam metode ini, *precision* n-gram dihitung dengan melakukan penyesuaian terhadap n-gram yang berlebihan untuk setiap ukuran n. Dengan cara ini, *precision* yang dihitung tidak hanya mencerminkan akurasi kata yang tepat, tetapi juga menghindari peningkatan yang tidak realistis dalam skor akibat pengulangan n-gram yang sama. *Modified precision* mengurangi jumlah n-gram yang dihitung berdasarkan frekuensinya dalam referensi. Ini berarti bahwa jika sebuah n-gram muncul beberapa kali dalam kalimat prediksi tetapi hanya ada dalam jumlah terbatas dalam kalimat referensi, jumlahnya akan dibatasi sesuai dengan kemunculan maksimum di referensi. Dengan demikian, *precision* yang dimodifikasi memberikan gambaran yang lebih akurat tentang kinerja model dengan menghindari masalah *over-counting*.

Perlu dicatat, jenis *precision* yang telah dimodifikasi ini merupakan komponen penting dalam perhitungan Skor BLEU. BLEU menggunakan *precision* yang telah dimodifikasi untuk memberikan penilaian yang lebih adil terhadap kualitas terjemahan, memastikan bahwa n-gram yang berlebihan tidak memberikan kontribusi yang tidak semestinya pada skor akhir. *Modified n-gram precision* didefinisikan sebagaimana berikut pada persamaan (9).

$$\text{Modified Precision}_n = \frac{\text{Jumlah n-gram yang cocok}}{\text{Jumlah n-gram yang seharusnya}} \quad \text{.....(9)}$$

Selanjutnya, skor *precision* yang telah dimodifikasi digabungkan menggunakan formula berikut pada persamaan (10). Formula ini dapat menghitung *precision* untuk berbagai nilai N dan menerapkan bobot yang berbeda-beda. Pada umumnya, seperti pada *paper* asli (Papineni dkk., 2002) digunakan nilai N = 4, yang berarti perhitungan mencakup n-gram hingga ukuran empat kata (*four-gram*), dengan bobot seragam $w_n = 1 / N$.

$$\begin{aligned} Geometric\ Average\ Precision\ (N) &= exp(\sum_{n=1}^{N} w_n\ log\ p_n) \\ &= \prod_{n=1}^{N} {p_n}^{w_n} \\ &= (p_1)^{\frac{1}{4}} \cdot (p_2)^{\frac{1}{4}} \cdot (p_3)^{\frac{1}{4}} \cdot (p_4)^{\frac{1}{4}} \end{aligned} \quad \text{.....(10)}$$

Langkah berikutnya adalah menghitung *brevity penalty*. *Brevity penalty* digunakan untuk mencegah terjemahan yang terlalu singkat mendapatkan skor yang terlalu tinggi. Penalti ini diterapkan ketika panjang terjemahan mesin lebih pendek atau sama dengan panjang referensi, menggunakan rumus berikut pada persamaan (11).

$$\mathrm{BP} = \begin{cases} 1 & \text{jika } c > r \\ e^{(1-\frac{r}{c})} & \text{jika } c \leq r \end{cases} \quad \text{.....(11)}$$

Di mana r adalah panjang total dari korpus referensi, dan c adalah panjang total dari korpus yang diprediksi. Ini memastikan bahwa *brevity penalty* tidak melebihi 1, meskipun kalimat yang diprediksi jauh lebih panjang daripada sasaran. Sebaliknya, jika jumlah kata yang diprediksi sangat sedikit, nilai *brevity penalty* akan menjadi kecil. Terakhir, untuk menghitung skor BLEU, *brevity penalty* dikalikan dengan rata-rata geometrik dari skor *precision* seperti berikut pada persamaan (12).

$$Bleu\ (N) = Brevity\ Penalty \cdot Geometric\ Average\ Precision\ Scores\ (N) \quad \text{.....(12)}$$

Perlu diperhatikan bahwa dalam penghitungan skor BLEU untuk korpus, skor ini dihitung berdasarkan keseluruhan korpus, bukan pada kalimat-kalimat individual. Skor BLEU dihitung dengan mempertimbangkan keseluruhan teks dalam korpus yang diprediksi. Oleh karena itu, tidak mungkin untuk menghitung skor BLEU secara terpisah untuk setiap kalimat dalam korpus dan kemudian mengambil rata-rata skor tersebut dengan cara tertentu.

V. SIMPULAN DAN SARAN

5.1. Simpulan

Berdasarkan hasil penelitian yang telah dilakukan mengenai penerjemahan bahasa Lampung ke bahasa Indonesia menggunakan *long short-term memory* (LSTM), beberapa kesimpulan dapat diambil. Pertama, penerapan arsitektur LSTM terbukti efektif untuk penerjemahan bahasa Lampung ke bahasa Indonesia, terutama pada kalimat pendek yang terdiri dari 2 hingga 3 kata. Hasil penelitian menunjukkan bahwa model LSTM mampu menangkap hubungan n-gram dengan sangat baik pada kalimat pendek, menghasilkan nilai BLEU yang tinggi. Kedua, meskipun model LSTM menunjukkan performa yang baik pada kalimat pendek, performa model cenderung menurun seiring dengan meningkatnya panjang dan kompleksitas kalimat. Hal ini terjadi karena LSTM memiliki keterbatasan dalam menangani dependensi jarak jauh yang sering muncul pada kalimat panjang, yang berdampak pada akurasi penerjemahan n-gram yang lebih tinggi. Ketiga, penelitian ini juga menunjukkan bahwa model lebih rentan terhadap kesalahan penerjemahan pada kalimat yang lebih kompleks, yang mencerminkan tantangan dalam mempertahankan konsistensi konteks sepanjang kalimat panjang. Terakhir, potensi LSTM dalam penerjemahan bahasa Lampung ke bahasa Indonesia sangat signifikan, namun diperlukan pengembangan lebih lanjut untuk meningkatkan kemampuan model dalam menangani kalimat yang lebih panjang dan kompleks.

5.2. Saran

Berdasarkan hasil dan temuan penelitian ini, beberapa saran dapat diberikan untuk penelitian selanjutnya. Pertama, disarankan untuk memperluas dan mendiversifikasi *dataset* dengan menambahkan lebih banyak data teks dalam bahasa Lampung dan bahasa Indonesia, sehingga model dapat dilatih dengan lebih baik dan mampu menangkap berbagai variasi bahasa. Kedua, mekanisme *attention* dapat diintegrasikan ke dalam model LSTM guna meningkatkan kemampuan model dalam menangani kalimat panjang dan mempertahankan *long-range dependence*, yang dapat meningkatkan akurasi penerjemahan secara keseluruhan. Ketiga, eksplorasi model lain seperti *transformer* yang memiliki kemampuan *distributed attention* dapat menjadi alternatif yang lebih efektif dalam menangani kalimat panjang dan kompleks dibandingkan model LSTM. Keempat, perlu dilakukan pengujian lebih lanjut dalam konteks dunia nyata, seperti pada aplikasi penerjemahan otomatis atau alat bantu penerjemahan, untuk mendapatkan gambaran yang lebih akurat mengenai performa model dan bagaimana model tersebut dapat disesuaikan agar lebih bermanfaat dan efisien dalam penggunaan sehari-hari. Kelima, penelitian lanjutan dapat difokuskan pada optimasi parameter, struktur model LSTM, dan strategi pelatihan, guna meningkatkan performa model pada berbagai jenis kalimat. Dengan mengikuti saran-saran ini, diharapkan penelitian di masa depan dapat menghasilkan model penerjemahan yang lebih baik dan lebih akurat, memberikan kontribusi yang signifikan dalam pengembangan teknologi penerjemahan bahasa Lampung ke bahasa Indonesia, serta berkontribusi dalam pelestarian bahasa Lampung dan memfasilitasi komunikasi antarbudaya.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

Abidin, Z. 2017. Penerapan Neural Machine Translation untuk Eksperimen Penerjemahan secara Otomatis pada Bahasa Lampung – Indonesia

Ariyani, F., Sopari, S., Mulya, G. A., Murdani, D., dan Bahtiyar. 2015. *Kamus Dwi Bahasa Indonesia-Lampung Dialek Way Kanan*

Ariyani, F., Udin, N., Wetty, N. N., Hilal, I., dan H.M., J. 1999. *Kamus Bahasa Indonesia-Lampung dialek A*. Pusat Pembinaan dan Pengembangan Bahasa, Departemen Pendidikan dan Kebudayaan

Bahdanau, D., Cho, K., dan Bengio, Y. 2015. Neural Machine Translation by Jointly Learning to Align and Translate

Britz, D., Goldie, A., Luong, M.-T., dan Le, Q. 2017. Massive Exploration of Neural Machine Translation Architectures

Cau, R., Pisu, F., Muscogiuri, G., Mannelli, L., Suri, J. S., dan Saba, L. 2023. Applications of Artificial Intelligence-Based Models in Vulnerable Carotid Plaque. *Vessel Plus*. 2023

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., dan Bengio, Y. 2014. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation

Du, Lim, dan Tan. 2019. A Novel Human Activity Recognition and Prediction in Smart Home Based on Interaction. *Sensors*. Vol. 19. no. 20. hlm 4474

Faiza, I. M., Gunawan, G., dan Andriani, W. 2022. Tinjauan Pustaka Sistematis: Penerapan Metode Machine Learning untuk Deteksi Bencana Banjir. *Jurnal Minfo Polgan*. Vol. 11. no. 2. hlm 59–63

Faqih, A. 2018. Penggunaan Google Translate Dalam Penerjemahan Teks Bahasa Arab Ke Dalam Bahasa Indonesia. *ALSUNIYAT: Jurnal Penelitian Bahasa, Sastra, dan Budaya Arab*. Vol. 1. no. 2. hlm 88–97

Fauziyah, Y., Ilyas, R., dan Kasyidi, F. 2022. Mesin Penterjemah Bahasa Indonesia-Bahasa Sunda Menggunakan Recurrent Neural Networks. *Jurnal Teknoinfo*. Vol. 16. no. 2. hlm 313

Garg, S., Peitz, S., Nallasamy, U., dan Paulik, M. 2019. Jointly Learning to Align and Translate with Transformer Models

Greenstein, E. dan Penner, D. 2015. Japanese-to-English Machine Translation Using Recurrent Neural Networks. 2015

Herman. 2013. *Kamus Bahasa Lampung: 70 ribu kata*

Hermanto, A., Adji, T. B., dan Setiawan, N. A. 2015. Recurrent neural network language model for English-Indonesian Machine Translation: Experimental study. hlm. 132–136. dalam 2015 International Conference on Science in Information Technology (ICSITech) –, *2015 International Conference on Science in Information Technology (ICSITech)*, IEEE, Yogyakarta

Huda, A. dan Ardi, N. 2020. *Dasar-Dasar Pemrograman Berbasis Python*. UNP Press, Padang

Husada, I. N. dan Toba, H. 2020. Pengaruh Metode Penyeimbangan Kelas Terhadap Tingkat Akurasi Analisis Sentimen Pada Tweets Berbahasa Indonesia. *Jurnal Teknik Informatika dan Sistem Informasi*. Vol. 6. no. 2

Jiantono, A. C. 2023. Mengenal Deep Learning Beserta Contoh Penerapannya. *School of Information Systems*

Junczys-Dowmunt, M., Dwojak, T., dan Hoang, H. Is Neural Machine Translation Ready for Deployment? A Case Study on 30 Translation Directions

Kahlon, N. K. dan Singh, W. 2023. Machine Translation from Text to Sign Language: A Systematic Review. *Universal Access in the Information Society*. Vol. 22. no. 1. hlm 1–35

Kattamuri, M. 2018. *Introduction to Matplotlib*, GeeksforGeeks, https://www.geeksforgeeks.org/python-introduction-matplotlib/ , diakses tanggal 12 Mei 2024.

Kingma, D. P. dan Ba, J. 2017. Adam: A Method for Stochastic Optimization

Klein, G., Kim, Y., Deng, Y., Nguyen, V., Senellart, J., dan Rush, A. M. 2018. OpenNMT: Neural Machine Translation Toolkit

Kostadinov, S. 2018. *Recurrent Neural Networks with Python Quick Start Guide: Sequential Learning and Language Modeling with TensorFlow*. Packt Publishing Ltd, Birmingham

Lewis, M. P. (ed.). 2009. *Ethnologue: languages of the world*. SIL International, Dallas, Tex

Manu. 2021. *A simple overview of RNN, LSTM and Attention Mechanism*, https://medium.com/swlh/a-simple-overview-of-rnn-lstm-and-attention-mechanism-9e844763d07b , diakses tanggal 9 Mei 2024.

Muam, A. dan Nugraha, C. D. 2020. *Pengantar Penerjemahan*. Gadjah Mada University Press, Yogyakarta

Naik, P. G. 2023. *Conceptualizing Python in Google COLAB*. Shashwat Publication, Chhattisgarh

Nelli, F. 2023. *Python Data Analytics*. Apress, Berkeley, CA

Papineni, K., Roukos, S., Ward, T., dan Zhu, W.-J. 2002. Bleu: a Method for Automatic Evaluation of Machine Translation. hlm. 311–318. dalam Isabelle, P., Charniak, E., dan Lin, D. (ed.), ACL 2002 –, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, Philadelphia, Pennsylvania, USA

Prima, A. 2022. Alih Bahasa Melayu Belitung ke Bahasa Indonesia Dengan Pendekatan Berbasis Aturan

Purnomo, F. F. N. 2024. Pemanfaatan Artificial Intelligence pada Business Intelligence. *School of Information Systems*

Putri, N. W. 2018. Pergeseran Bahasa Daerah Lampung Pada Masyarakat Kota Bandar Lampung. *Prasasti: Journal of Linguistics*. Vol. 3. no. 1. hlm 83–97

Qiu, J., Wang, B., dan Zhou, C. 2020. Forecasting Stock Prices with Long-Short Term Memory Neural Network Based on Attention Mechanism. *PLOS ONE*. Vol. 15. no. 1. hlm e0227222

Rahmadya. 2020. Koneksi Google Drive dengan Google Colab. *Rahmadya Trias Handayanto*

Raschka, S. dan Mirjalili, V. 2019. *Python Machine Learning - Third Edition*. Packt Publishing

Rorro, M. 2019. Introduction to Keras TensorFlow. 2019

Santhosh, S. 2023. Understanding BLEU and ROUGE Score for NLP Evaluation. *Medium*

Schuster, M. dan Paliwal, K. K. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*. Vol. 45. no. 11. hlm 2673–2681

Shah, D. 2023. *Cross Entropy Loss: Intro, Applications, Code*, https://www.v7labs.com/blog/cross-entropy-loss-guide, https://www.v7labs.com/blog/cross-entropy-loss-guide , diakses tanggal 4 Agustus 2024.

Shelf. 2024. *Why Recurrent Neural Networks (RNNs) Dominate Sequential Data Analysis*, Shelf, https://shelf.io/blog/recurrent-neural-networks/ , diakses tanggal 9 Mei 2024.

Shen, J., Chen, P.-J., Le, M., He, J., Gu, J., Ott, M., Auli, M., dan Ranzato, M. 2020. The Source-Target Domain Mismatch Problem in Machine Translation

Siddique, S., Ahmed, T., Rifayet Azam Talukder, Md., dan Mohsin Uddin, Md. 2020. English to Bangla Machine Translation Using Recurrent Neural Network. *International Journal of Future Computer and Communication*. hlm 46–51

Smagulova, K. dan James, A. P. 2020. Overview of Long Short-Term Memory Neural Networks. hlm. 139–153. dalam James, A. P. (ed.), *Deep Learning Classifiers with Memristive Networks: Theory and Applications*, Springer International Publishing, Cham

Smartling. 2024. *What Goes Into Assessing Machine Translation Quality?*, https://www.smartling.com/resources/101/how-to-assess-machine-translation-quality/ , diakses tanggal 9 Mei 2024.

Sujaini, H. 2018. Peningkatan Akurasi Penerjemah Bahasa Daerah dengan Optimasi Korpus Paralel. *Jurnal Nasional Teknik Elektro dan Teknologi Informasi (JNTETI)*. Vol. 7

Suryani, A. A., Arieshanti, I., Yohanes, B. W., Subair, M., Budiwati, S. D., dan Rintyarna, B. S. 2016. Enriching English into Sundanese and Javanese translation list using pivot language. hlm. 167–171. dalam 2016 International Conference on Information & Communication Technology and Systems (ICTS) –, *2016 International Conference on Information & Communication Technology and Systems (ICTS)*

Sutskever, I., Vinyals, O., dan Le, Q. V. 2014. Sequence to Sequence Learning with Neural Networks

Wathani, M. R. dan Hidayati, N. 2023. Analisis Perbandingan Fungsi Aktivasi CNN Pada Pengelompokan Jenis Beras Berdasarkan Mutu Beras. *Brahmana : Jurnal Penerapan Kecerdasan Buatan*. Vol. 4. no. 2. hlm 144–153

Wołk, K. 2019. *Machine learning in translation corpora processing*. Taylor & Francis, Boca Raton, FL

Wuryantoro, A. 2018. *Pengantar Penerjemahan*. Deepublish, Sleman

Ying, X. 2019. An Overview of Overfitting and Its Solutions. *Journal of Physics: Conference Series*. Vol. 1168