DETEKSI KALIMAT BAHASA LAMPUNG DIALEK NYO DENGAN PENDEKATAN NEURAL NETWORK

Sentence Detection of Lampung Nyo Dialect with A Neural Network Approach

Acc Revisi

Pembimbing

31/5 2023

Skripsi

Untuk memenuhi sebagian persyaratan
Mencapai derajat S-1

Diajukan oleh:
RAHMAT ALAMSYAH
17312224

Acc Kemkr,
29/6 2023
[illegible], M.Pd.

Acc Cetak.
31/5-2023

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK DAN ILMU KOMPUTER
UNIVERSITAS TEKNOKRAT INDONESIA
BANDAR LAMPUNG

2023

LEMBAR PENGESAHAN
SKRIPSI

DETEKSI KALIMAT BAHASA LAMPUNG DIALEK NYO DENGAN PENDEKATAN NEURAL NETWORK

Dipersiapkan dan disusun oleh :

RAHMAT ALAMSYAH
17312224

Telah dipertahankan di depan Dewan Penguji
pada tanggal 23 Mei 2023

Pembimbing,	Penguji,
Zaenal Abidin, S.Si., S.Kom., M.T.	**Permata, S.Si., M.Si.**
NIK. 022 08 10 03	NIK. 022 09 03 03

Skripsi ini telah diterima sebagai salah satu persyaratan
untuk memperoleh gelar sarjana
tanggal 31 Mei 2023

Fakultas Teknik dan Ilmu Komputer Dekan,	Program Studi S1 Informatika Ketua,
Dr. H. Mahathir Muhammad, S.E., M.M.	**Dyah Ayu Megawaty, M.Kom.**
NIK.023 05 00 09	NIK.022 09 03 05

LEMBAR PERNYATAAN

Yang bertanda tangan dibawah ini :

Nama : Rahmat Alamsyah

NPM : 17312224

Program Studi : Informatika

Dengan ini menyatakan bahwa tugas akhir :

Judul : Deteksi Kalimat Bahasa Lampung Dialek Nyo Dengan Pendekatan Neural Network

Pembimbing : Zaenal Abidin, S.Si., S.Kom., M.T

Belum pernah diajukan untuk diuji sebagai persyaratan untuk memperoleh gelar akademik pada berbagai tingkatan di universitas/perguruan tinggi manapun. Tidak ada bagian dalam skripsi ini yang pernah dipublikasikan oleh pihak lain, kecuali bagian yang digunakan sebagai referensi, berdasarkan kaidah penulisan ilmiah yang benar.

Apabila dikemudian hari ternyata laporan tugas akhir yang saya tulis terbukti hasil saduran/plagiat, maka saya akan bersedia menanggung segala resiko yang akan saya terima.

Demikian pernyataan ini dibuat dengan sebenar-benarnya.

Bandar Lampung, 23 Mei 2023

Yang menyatakan,

Rahmat Alamsyah

17312224

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI
SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia, saya yang bertanda tangan di bawah ini:

Nama : Rahmat Alamsyah

NPM : 17312224

Program Studi : Informatika

Jenis karya : Skripsi/Tesis

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia, **Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty-Free Right*)** atas karya ilmiah saya yang berjudul :

"Deteksi Kalimat Bahasa Lampung Dialek Nyo Dengan Pendekatan Neural Network"

Beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas **Royalti Noneksklusif** ini Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia berhak menyimpan, mengalihmedia/*formatkan*, mengelola dalam bentuk pangkalan data (*database*), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Bandar Lampung

Pada tanggal : 27 Mei 2023

Yang menyatakan,

Rahmat Alamsyah

NPM. 17312224

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya, penulis dapat menyelesaikan skripsi ini. Penulisan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar sarjana pada Program Studi S1 Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia.

Penulis rnenyadari bahwa, tanpa bantuan dan bimbingan dari berbagai pihak, dari masa perkuliahan sampai pada penyusunan skripsi ini, sangatlah sulit bagi penulis untuk menyelesaikan laporan ini. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Bapak Dr. H.M. Nasrullah Yusuf, S.E., M.B.A., selaku Rektor Universitas Teknokrat Indonesia.
2. Bapak Dr. H. Mahathir Muhammad, S.E., M.M., selaku Dekan Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia
3. Ibu Dyah Ayu Megawaty, S.Kom., M.Kom., selaku Ketua Program Studi S1 Informatika, Fakultas Teknik dan Ilmu Komputer, Universitas Teknokrat Indonesia.
4. Bapak Zaenal Abidin, S.Si., S.Kom., M.T., selaku Dosen Pembimbing yang telah meluangkan waktu untuk membimbing penulis menyelesaikan skripsi ini
5. Bapak Permata, S.Si., M.Si., selaku Dosen Penguji yang telah menyediakan waktu untuk menguji demi kelancaraan jalannya sidang

Akhir kata, penulis berharap semoga Allah SWT berkenan membalas segala kebaikan semua pihak yang telah membantu dan sernoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Bandarlampung, 23 Mei 2023

Penulis

DAFTAR ISI

DAFTAR TABEL

Hal

DAFTAR GAMBAR

Hal

DAFTAR LAMPIRAN

Hal

ABSTRAK

DETEKSI KALIMAT BAHASA LAMPUNG DIALEK NYO DENGAN PENDEKATAN NEURAL NETWORK

Sentence Detection of Lampung Dialect Nyo with a Neural Network approach

Oleh

Rahmat Alamsyah
17312224

Bahasa Lampung adalah bahasa asli yang digunakan masyarakat Lampung itu sendiri. Namun seiring dengan perkembangan zaman, penggunaannya didaerah perkotaan sangatlah minim dikarenakan keberagaman dan perkembangannya lebih dominan menggunakan bahasa Indonesia sehingga bahasa Lampung terancam punah. Oleh karena itu perlu dilakukan pelestarian, salah satu upaya tersebut adalah membuat mesin penerjemah bahasa Lampung ke bahasa Indonesia dengan meningkatkan akurasi menggunakan *Direct Machine Translation* (DMT) dan *Bilingual Evaluation Understudy* (BLEU). Tetapi pada penelitian sebelumnya belum ada yang menerapkan algoritma Neural network untuk mendeteksi bahasa Lampung sebagai pendukung *Machine Translation* (MT). Pada penelitian ini penulis ingin melihat bagaimana cara kerja algoritma *Back-Propagation Neural network* dalam mendeteksi bahasa Lampung dan bahasa Indonesia, yang akan diterapkan melalui bahasa pemrograman python sehingga menghasilkan suatu engine deteksi. Dimana engine tersebut akan diuji menggunakan *Performance Evaluation Measure* (PEM) untuk melihat *Accuracy, Precision, Recall* dalam bentuk persentase.

Kata Kunci : Bahasa Lampung, *Back-Propagation Neural Network, Python, Performance Evaluation Measure, Accuracy, Precision, Recall*

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dari 34 provinsi di Indonesia, Lampung merupakan salah satu Provinsi Indonesia yang letaknya berada di ujung Selatan Pulau Sumatera. Provinsi Lampung terkenal dengan julukan “Sai Bumi Ruwa Jurai” dengan arti satu bumi yang di huni oleh dua aliran (kelompok) budaya yang berbeda. Adapun aliran masyarakat budaya tersebut adalah Pepadun dan Saibatin, dapat diartikan pendatang dan pemukim. Yang mana masyarakat Pepadun mendiami daerah pedalaman atau daratan yang tidak dekat dengan laut, seperti daerah Way Kanan, Pubian, Sungkai, Abung, dan Tulang Bawang. Sedangkan masyarakat Saibatin mendiami daerah pesisir pantai, seperti pesisir Krui, pesisir Semaka, pesisir Rajabasa, Labuhan Maringgai, dan Belalau.

Bahasa Lampung merupakan bahasa asli yang digunakan masyarakat Lampung itu sendiri. Seperti halnya masyarakat Lampung yang terdapat dua aliran adat, bahasa Lampung juga terbagi menjadi dua yaitu dialek Nyow (O) digunakan oleh masyarakat Pepadun dan dialek Api (A) digunakan oleh masyarakat Saibatin. Namun dalam penggunaan Bahasa lampung pada daerah perkotaan sangatlah minim dikarenakan keberagaman dan perkembangan masyarakat perkotaan lebih dominan menggunakan bahasa Indonesia. Semakin turunnya pemakaian bahasa Lampung pada masyarakat serta cara penyampaian dalam pembelajaran bahasa lampung yang terkesan statis, merupakan beberapa faktor yang dapat mempengaruhi berkurangya minat masyarakat dalam mempelajari tentang pentingnya bahasa lampung.

Oleh sebab itu Bahasa lampung terancam punah, maka perlu dilakukan pelestarian terutama pada kalangan generasi muda. Dalam Ardiyatno 2021, Abidin dkk menyatakan bahwa dalam Upaya yang telah dilakukan oleh pemerintah daerah provinsi Lampung adalah dibuatnya Peraturan Gubernur Nomor 39 Tahun 2014 yaitu menjadikan mata pelajaran bahasa dan Aksara Lampung wajib sebagai muatan lokal mulai dari SD hingga SMA. Upaya lain dalam pelestarian bahasa Lampung juga datang dari akademisi yaitu penelitian tentang mesin penerjemah bahasa Lampung dialek Api dengan *stemming* menggunakan 2000 kata berimbuhan ke bahasa Indonesia menggunakan *Brute-Force* dan Pemrograman C# (Abidin, Wijaya dan Pasha, 2021) dan upaya meningkatkan akurasi mesin penerjemah bahasa Lampung ke bahasa Indonesia menggunakan *Direct Machine Translation* (DMT) dan *Bilingual Evaluation Understudy* (BLEU) (Ardiyatno, 2021).

Pada penelitian yang dilakukan oleh Pray Cristanto (2021) menggunakan label Lampung dan Indonesia dari 2998 korpus dalam tesisnya tentang deteksi dialek Lampung Api (A) dengan bahasa pemrograman C# menggunakan algoritma *Naïve Bayes* yang nilai akurasinya sebesar 96 %, dan Joseph Frans Sijabat (2021) menggunakan dialek Lampung Nyow (O) dan Api (A) dari 7998 korpus menggunakan bahasa pemrograman *Python* dan algoritma *Naïve Bayes* dengan tingkat akurasi sebesar 98%. Pada penelitian sebelumnya tersebut belum ada yang menerapkan algoritma Neural network untuk mendeteksi bahasa Lampung sebagai pendukung penelitian mengenai *Machine Translation* (*MT*).

Neural network adalah sebuah jaringan yang dirancang untuk menyerupai otak manusia yang bertujuan untuk melaksanakan suatu tugas tertentu

(Haykin, 2009). Neural network dibuat berdasarkan model saraf manusia tetapi dengan bagian-bagian yang lebih sederhana. Komponen terkecil dari Neural network adalah *unit* atau yang biasa disebut dengan *neuron*, dimana *neuron* tersebut akan mentransformasikan informasi yang diterima menuju *neuron* lainnya (Shukla, dkk, 2010)

Pada penelitian ini algoritma Neural network yang digunakan adalah Back-Propagation. Algoritma ini dipilih karena memungkinkan untuk menghindari kesulitan yang dijelaskan menggunakan aturan belajar (A. Wanto, 2018). Algoritma Back-Propagation Neural Network akan diterapkan dan dikembangkan pada engine dengan menggunakan bahasa pemrograman *python* serta diimplementasikan pada *tools Google Collab* yang nantinya menjadi sebuah program pendeteksi kalimat bahasa Lampung.

1.2 Rumusan Masalah

Berdasarkan uraian yang ada di latar belakang, maka permasalahan yang akan diselesaikan pada penelitian ini adalah bagaimana cara kerja dari algorima Back-Propagation Neural network untuk mendeteksi sebuah kalimat bahasa Lampung dialek Nyow (O) dan kalimat bahasa Indonesia ?

1.3 Batasan Masalah

Sesuai dengan judul yang tertera diatas, penulis membatasi pembahasan penelitian agar ruang lingkup proyek ini sesuai dan tidak menyimpang dari objek pembahasan yang diinginkan. Adapun ruang lingkup pembatasan pada penelitian ini adalah sebagai berikut :

1. Penelitian ini hanya berfokus pada pendeteksian antara bahasa Lampung dialek Nyow (O) dan bahasa Indonesia.
2. Bahasa Lampung yang digunakan dalam penelitian ini menggunakan bahasa Lampung dialek Nyow (O).
3. Data yang digunakan pada penelitian ini adalah korpus yang berasal dari penelitian sebelumnya dengan jumlah 9077 kalimat bahasa Indonesia dan 9077 kalimat bahasa Lampung. Sehingga totalnya yaitu 18154, dengan pembagian 80% untuk data training dan 20% atau untuk data testing
4. Model yang dibangun ini hanya dapat mendeteksi bahasa dari data yang sudah ada.
5. Jumlah iterasi yang digunakan pada model adalah 1000 iterasi.

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk merancang sebuah *engine detection language* atau mesin pendeteksi bahasa pada sebuah kalimat dengan mendeteksi apakah kalimat tersebut berbahasa Lampung dialek Nyow (O) atau berbahasa Indonesia dengan metode Back-Propagation Neural network.

1.5 Manfaat atau Kontribusi Penelitian

Manfaat yang dapat diharapkan dalam penelitian ini adalah :

a. Untuk mengetahui bagaimana algoritma Back-Propagation Neural network dapat mendeteksi sebuah kalimat bahasa Lampung dialek Nyow (O) dan bahasa Indonesia.
b. Mendukung *Direct Machine Translation* (DMT) berbasis kamus.
c. Membantu melestarikan Bahasa Lampung yang kurang popular belakangan ini

BAB II
LANDASAN TEORI

2.1 Tinjauan Pustaka

Dalam penelitian ini akan digunakan enam tinjauan studi yang nantinya dapat mendukung penelitian. Berikut ini merupakan tinjauan studi yang digunakan dalam penelitian, dapat dilihat pada Tabel 2.1 Daftar Literatur.

Tabel 2.1 Daftar Literatur

No.	Penulis	Judul	Metode dan Hasil
01	Omar Hussein, Hachem Sfar, Jelena Mitrovic, Michael Granitzer (2020)	*NLP_Passau at SemEval-2020 Task 12: Multilingual Neural network for Offensive Language Detection in English, Danish and Turkish*	Algoritma CNN dan RNN dengan hasil akurasi Turki 83.34%, Denmark 88.89%, Inggris 77.87%; presisi Turki 74%, Denmark 88%, Inggris 75%; dan recall Turki 65%, Denmark 59%, Inggris 75%
02	Crisanadenta Wintang Kencana, Erwin Budi Setiawan, Isman Kurniawan (2020)	*Hoax Detection on Twitter using Feed-forward and Back-Propagation Neural networks Method*	Algoritma BNN dengan hasil akurasi 78.76%, presisi 73.40% dan recall 75%
03	Theresia Hendrawati, Christina Purnama Yanti (2021)	*Analysis of Twitter Users Sentiment against the Covid-19 Outbreak Using the Backpropagation Method with Adam*	Algoritma BNN dengan Adam Optimization dengan akurasi 70%, presisi 65% dan recall 67 %

No.	Penulis	Judul	Metode dan Hasil
		Optimization	
04	Euis Saraswati, Yuyun Umaidah, Apriade Voutama (2021)	Penerapan Algoritma *Artificial Neural network* untuk Klasifikasi Opini Publik Terhadap Covid-19	Algoritma BNN dengan hasil akurasi 88.62%, presisi 91.5% dan recall 95.73%
05	Mhd. Denry Aruna Nasution, Jaya Tata Hardinata, Irfan Sudahri Damanik (2019)	Jaringan Syaraf Tiruan *Backpropagation* untuk Klasifikasi Data Tilang Berdasarkan Jenis Pelanggaran	Algoritma BNN dengan hasil akurasi 95%, presisi 87%, dan recall 93%
06	Conor O'Sullivan (2020)	*Deep* Neural network *Language Identification*	Algoritma DNN dan n-gram karakter dengan hasil akurasi 98.26%, presisi 95% dan recall 97%

2.1.1 Literatur 1

Makalah ini menjelaskan model Neural network (NN) yang digunakan untuk berpartisipasi dalam *OffensEval*, Tugas 12 dari lokakarya *SemEval* 2020. Tujuan dari tugas ini adalah untuk mengidentifikasi pidato ofensif di media sosial, khususnya dalam tweet. Model yang kami gunakan, *C-BiGRU*, terdiri dari *Convolutional* Neural network *(CNN)* bersama dengan *Bidirectional Recurrent* Neural network *(RNN)*. Representasi numerik multi dimensi (*embedding*) untuk setiap kata dalam tweet, yaitu digunakan oleh model, ditentukan menggunakan *fastText*. Ini digunakan dengan kumpulan data berlabel tweet untuk melatih model dalam mendeteksi kombinasi kata yang dapat menyampaikan serangkaian arti.

Dalam makalah ini, sistem yang kami gunakan untuk berpartisipasi dalam *OffensEval* 2020 ditampilkan dan penjelasan rinci diberikan tentang arsitektur model *C-BiGRU* dan *pre-processing* data yang digunakan. Adapun *dataset* untuk pelatihan yang digunakan adalah 14.100 untuk bahasa Inggris, 2.961 untuk bahasa Denmark, dan 31.277 bahasa Turki. Model ini mencapai skor F1 rata-rata makro sebesar 90,882%, 76,76%, dan 76,70% untuk bahasa Inggris, Turki, dan Denmark masing-masing pada pelabelan *dataset OffensEval 2*.

2.1.2 Literatur 2

Media sosial adalah salah satu cara untuk menghubungkan setiap individu di dunia. Itu juga digunakan oleh orang-orang yang tidak bertanggung jawab untuk menyebarkan *hoax*. *Hoax* adalah berita bohong yang dibuat seolah-olah benar. Hal ini dapat menimbulkan kecemasan dan kepanikan di masyarakat. Hal ini dapat mempengaruhi kondisi sosial dan politik. Pada era saat ini, media sosial yang paling populer adalah Twitter. Ini adalah tempat untuk berbagi informasi dan pengguna di seluruh dunia dapat berbagi dan menerima berita dalam pesan singkat atau disebut tweet. Deteksi *hoax* mendapatkan minat yang signifikan dalam dekade terakhir.

Metode pendeteksian hoax yang ada saat ini berbasis konten berita atau konteks sosial dengan menggunakan fitur berbasis pengguna. Dalam studi ini, kami menghadirkan deteksi *hoax* berdasarkan jaringan saraf *Feed Forward & Back Propagation*. Dalam pengembangannya, kami menggunakan dua metode *vektorisasi, TF-IDF* dan *Word2Vec*. Model kami dirancang untuk mempelajari fitur klasifikasi berita hoax secara otomatis melalui beberapa *hidden layer* dibangun ke dalam jaringan saraf. Jaringan syaraf sebenarnya menggunakan

kemampuan otak manusia yang mampu memberikan rangsangan, proses, dan *output*. Ini dikerjakan oleh *neuron* untuk memproses setiap informasi yang masuk, kemudian diproses melalui koneksi jaringan, dan akan terus belajar untuk menghasilkan kemampuan melakukan klasifikasi.

Model yang kami usulkan akan membantu untuk memberikan solusi yang lebih baik untuk mendeteksi *hoax*. Pengumpulan data diperoleh melalui *crawling* menggunakan Twitter API dan mengambil data sesuai dengan kata kunci dan tagar dari tanggal 13 sampai 16 Maret 2020. Didapatkan 50.646 tweet yang mana sebanayak 25.021 *class hoax* dan 25.624 *class non-hoax*. Dari *dataset* tersebut dibagi menjadi 80% data *train* dan 20% data *test*. Jaringan saraf akurasi tertinggi diperoleh menggunakan *TF-IDF* dengan hasil 78,76% dibandingkan menggunakan *Word2Vec* dengan hasil 67,38%. Kami juga menemukan bahwa kualitas data memengaruhi kinerja.

2.1.3 Literatur 3

Penelitian ini memanfaatkan data dari Twitter dengan menganalisis tweet berbahasa Indonesia yang membahas tentang wabah virus Covid-19 untuk mengetahui pendapat pengguna Twitter tentang wabah virus Covid-19. Penelitian ini mencoba menganalisis sentimen untuk melihat opini pada pengguna tweet tentang virus Covid-19 dengan hasil sentimen Positif, Negatif atau Netral menggunakan *Multi-layer Perceptron* (MLP) dan *Backprogragation* dengan optimasi Adam. Pada penelitian ini mengumpulkan 200 dokumen (tweet) dalam bahasa Indonesia tentang Covid-19 yang di-tweet sejak November 2019 sampai dengan 30 Agustus 2020.

Setelah melalui 470 *epochs*, Model yang dihasilkan berhasil mendapatkan akurasi hingga 70% dengan skor f1 untuk kelas positif, negatif, dan netral masing-masing 0,77, 0,75, dan 0,5 dari nilai maksimum 1. Hal ini menunjukkan bahwa model pada penelitian ini cukup berhasil dalam melakukan proses klasifikasi sentimen untuk tweet berbahasa Indonesia dengan Tema Covid-19.

2.1.4 Literatur 4

Pada penelitian ini membuat sebuah sistem yang dapat melakukan klasifikasi opini publik terhadap Covid-19 menggunakan metode Backpropagation Neural network. Metode ini digunakan karena dapat melakukan klasifikasi data dengan jumlah fitur yang banyak dan beragam.

Penelitian ini berdasarkan data opini publik tentang virus yang telah beredar di banyak media sosial, termasuk media sosial Twitter. Di Twitter, ada beberapa diskusi terkait virus Covid-19. Perasaan positif atau negatif dapat ditemukan dalam opini di tweet. Sentimen sebuah tweet dapat diperhitungkan dan dievaluasi oleh pihak berwenang saat menangani virus Covid-19. Berdasarkan permasalahan tersebut, diperlukan klasifikasi analisis sentimen untuk mengetahui bagaimana virus Covid-19 dipersepsikan oleh masyarakat luas.

Algoritma Artificial Neural network (ANN) dan metode Backpropagation digunakan dalam penelitian ini. Pada pengujian tersebut, dihasilkan nilai presisi 91,5%, recall 95,73%, dan akurasi 88,62%. Hasilnya menunjukkan bahwa model Neural network sangat efektif untuk pengklasifikasian *text mining.*

2.1.5 Literatur 5

Bukti pelanggaran (tilang) diberikan kepada setiap orang yang melakukan aktivitas pelanggaran lalu lintas. Karena tingginya tingkat pelanggaran yang

terjadi menyebabkan data tilang yang tersimpan pada Kejaksaan Negeri Simalungun menjadi hambatan seperti kelebihan kapasitas ruang sidang. Dalam berkas tilang, ada beberapa poin penting yang menjelaskan tentang tilang secara mendetail seperti jenis barang bukti, kendaraan, pasal, kehadiran di persidangan, dan jenis operasi yang dilakukan. Klasifikasi dapat digunakan sebagai pilihan alternatif dalam pembagian jadwal sidang tilang. Klasifikasi tilang dalam penelitian ini menggunakan metode algoritma *Backpropagation*.

Susunan perhitungan *backpropagation* terdiri dari 2 (dua) bagian, yaitu bagian persiapan dan pegangan pengujian. Terdapat 100 unit informasi yang dipisahkan menjadi 2 (dua) yaitu 50 (lima puluh) unit informasi persiapan dan 50 (lima puluh) unit informasi uji. Informasi yang ada dimasukkan ke dalam desain jaringan klasifikasi yang terdiri dari input layer, hidden layer, output layer, learning rate dan tingkat errror. Adapun Tingkat akurasi terbaik yang diperoleh dari penelitian ini adalah sebesar 95.0%, dan presisi 87% dan recall 93%.

2.1.6 Literatur 6

Ada beberapa pendekatan berbeda untuk identifikasi bahasa dan, dalam artikel ini, kita akan membahasnya secara mendetail. Yaitu menggunakan Neural network dan karakter n-gram sebagai fitur. Pada akhirnya, penelitian ini menunjukkan bahwa akurasi lebih dari 98% dapat dicapai dengan pendekatan ini. Penelitian ini menggunakan 6.872.356 kalimat dalam 328 bahasa unik dengan pembagian 70% pelatihan, 20% validasi dan 10% untuk pengujian. Model yang digunakan adalah DNN yang memiliki 3 hidden layer dengan 250 node, dan 1 layer output untuk setiap bahasa dengan 6 node. Hingga menghasilkan tingkat akurasi sebesar 98.26%, presisi 95% dan recall 97%.

2.2 Artificial Neural network

Artificial Neural Network (ANN) adalah sebuah jaringan yang dirancang untuk menyerupai otak manusia yang bertujuan untuk melaksanakan suatu tugas tertentu (Haykin, 2009). Dalam pengimplentasian jaringan ini, biasanya menggunakan komponen elektronik atau disimulasikan pada aplikasi komputer.

Pada tahun 1943, Warren McCulloch dan Walter Pitts memperkenalkan konsep model Neural network untuk komputasi. Ini menandai langkah pertama di bidang ilmu jaringan Neural. Model mereka menggabungkan beberapa unit pemrosesan sederhana untuk meningkatkan daya komputasi secara keseluruhan.

ANN terdiri dari banyak *neuron* di dalamnya. *Neuron - neuron* ini akan dikelompokkan ke dalam beberapa *layer*. Ada tiga tipe *node* (*neuron*) yaitu, *input, hidden* dan *output*. Setiap relasi menghubungkan dua buah *node* dengan bobot tertentu dan juga terdapat arah yang menujukkan aliran data dalam proses (Kusrini dan Luthfi, 2009).

Setiap neuron yang ada di setiap lapisan Neural network terhubung dengan neuron di lapisan lainnya, kecuali layer input dan output. Hal ini hanya berlaku untuk lapisan-lapisan yang berada di antara keduanya. Informasi yang diterima oleh layer input akan diproses satu per satu oleh layer-layer dalam Neural network hingga mencapai layer output terakhir. Lapisan yang terletak di antara input dan output dikenal sebagai hidden layer. Namun, tidak semua Neural network memiliki hidden layer karena ada beberapa jenis yang hanya terdiri dari layer input dan output saja. Model ANN ditunjukkan dengan kemampuannya dalam emulasi, analisis, *prediksi*, dan *asosiasi*. Kemampuan yang dimiliki ANN dapat digunakan

untuk belajar dan menghasilkan aturan atau operasi dari beberapa contoh *input* yang dimasukan dan membuat prediksi tentang kemungkinan *output* yang akan muncul atau menyimpan karakteristik input yang diberikan kepada ANN.

2.3 Algoritma Backpropagation

Algoritma *Backpropagation* merupakan salah satu dari model jaringan saraf tiruan (JST). Pada tahun 1974 Paul Werbos pertama kali merumuskan algoritma ini dan digunakan Neural network yang dipopulerkan oleh Rumelhart bersama McClelland. Metode *Backpropagation* pada awalnya dirancang untuk neural network *feedforward*, tetapi pada perkembangannya, metode ini diadaptasi untuk pembelajaran pada model Neural network lainnya (Astuti, 2009). Model ini digunakan karena dalam kebanyakan kasus mengenali pola diperlukan mengidentifikasi ketika fakta-fakta atau fenomena akan terjadi lagi sebelum terjadi.

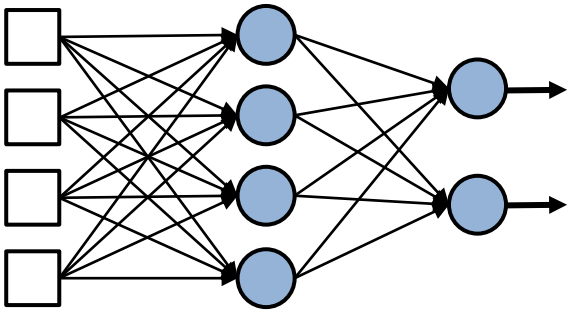
Pelatihan *Backpropagation* meliputi 3 fase:

i) Fase propagasi maju (*feedforward*) pola pelatihan masukan. Pola masukan dihitung maju mulai dari *layer* masukan hingga *layer* keluaran dengan fungsi aktivasi yang ditentukan,

ii) Fase propagasi mundur (*backpropagation*) dari *error* yang terkait. Selisih antara keluaran dan target merupakan kesalahan yang terjadi. Kesalahan tersebut dipropagasi mundur, dimulai dari garis yang berhubungan langsung dengan unit-unit dilayer keluaran;

iii) Fase modifikasi bobot.

Fase propagasi maju, Selama propagasi maju, sinyal masukan (X_1) dipropagasikan ke *layer* tersembunyi menggunakan fungsi aktivasi yang ditentukan. Keluaran dari unit tersembuyi (Z_j) tersebut selanjutnya dipropagasi maju lagi ke *layer* tersembunyi berikutnya dengan fungsi aktivasi yang telah ditentukan. Dan seterusnya hingga menghasilkan keluaran jaringan (Y_k).



Gambar 2.1 Algoritma Backpropagation

2.4 Bahasa pemrograman *Python*

Python berasal dari pengembangan bahasa pemrograman ABC oleh Guido van Rossum pada tahun 1990-an di Amsterdam. *Python* adalah bahasa pemrograman yang berorientasi objek dengan model skrip sehingga dapat dikatakan *Python* bersifat multifungsi karena dapat dipakai dalma pengembangan suatu perangkat lunak serta mampu berjalan pada semua sistem operasi.

Dalam penggunaannya *Python* memungkinkan para penggunanya untuk dapat lebih mengutamakan pengembangan aplikasi yang akan dibuat dikarenakan penulisan kode yang simpel dan mudah untuk diterapkan.

Karena bersifat *opensource* dan *freeware*, *Python* memiliki *library* yang dapat kembangkan dan dimengerti dengan mudah sehingga tidak ada batasan

dalam penggunaannya. Salah satu fitur yang tersedia pada *python* sebagai bahasa pemrograman dinamis yaitu *python* disempurnakan dengan adanya sistem manajemen memori otomatis.

2.5 Google Colaboratory

Sebuah aplikasi pengolahan data berbasis cloud dan gratis yang dikenal dengan nama Google Colaboratory atau Google Colab telah dikembangkan. Aplikasi ini dirancang menggunakan environment Jupyter dan mendukung berbagai library yang diperlukan dalam pengembangan teknologi kecerdasan buatan (AI), seperti yang telah dijelaskan oleh Wahyu pada tahun 2020.

Google Colab dibuat di atas environment Jupyter, sehingga mirip dengan Jupyter Notebook dan cara penggunaannya hampir sama. Satu-satunya perbedaan terletak pada media penyimpanan yang digunakan, yaitu Google Drive, dan alat ini berjalan pada sistem cloud. Selain itu, Google Colab juga menyediakan runtime Python 2 dan 3 yang telah dikonfigurasikan sebelumnya dengan berbagai *library*.

2.6 *Term Frequency – Inverse Document Frequency* (TF-IDF)

Term Frequency Inverse Text Frequency (TF-IDF) adalah sebuah metode yang digunakan untuk menghitung nilai frekuensi suatu kata dalam dokumen atau artikel dan sejauh mana kemunculan kata tersebut dalam beberapa publikasi. Algotima TF-IDF ini menentukan pentingnya sebuah kata dalam sebuah dokumen (Evan, 2014). Kamath pada tahun 2014 menjelaskan bahwa TF-IDF merupakan

salah satu algoritma yang sering digunakan dalam pengolahan data dalam skala besar.

TF menghitung seberapa sering sebuah kata muncul dalam dokumen. Sebuah kata mungkin bisa muncul berkali-kali dalam dokumen besar dibandingkan dengan yang kecil. Oleh karena itu, TF dihitung dengan membagi panjang dokumen. Dengan kata lain, TF dari sebuah kata dihitung dengan membaginya dengan jumlah kata dalam dokumen. TF dapat dihitung melalui persamaan berikut:

$$TF(t) = \frac{ft,d}{\sum t,d}$$

dimana *ft,d* merupakan frekuensi sebuah kata (*t*) muncul di dalam dokumen *d*, sedangkan $\sum t, d$ merupakan total keseluruhan kata yang terdapat di dalam dokumen d (Aizawa, 2002)

Untuk mengurangi dampak dari kata-kata umum seperti “apa,” “di,” dan seterusnya di korpus, makan digunkanlah TF-IDF. IDF memberikan bobot lebih pada kata-kata dengan frekuensi yang lebih tinggi atau lebih rendah. Sedangkan IDF diperoleh mealui persamaan berikut :

$$IDF_j = \log\left(\frac{|D|}{ft,D}\right)$$

|D| merupakan jumlah dokumen yang ada di dalam koleksi, sedangkan ft, merupakan jumlah dokumen dimana t muncul di dalam D (Salton & Buckley, 1988, Berger, et al, 2000). Dalam koleksi dokumen D, sebuah kata t dan dokumen individu d ϵ D

Kombinasi ini Metode TF dan IDF dikenal sebagai TF-IDF dan direpresentasikan secara matematis sebagai berikut:

$$TF\text{-}IDF(t,d,D) = TF(t,d) \times \log\left(\frac{D}{dft}\right)$$

di mana *t* menunjukkan istilah kata; *d* menunjukkan setiap dokumen; *D* mewakili kumpulan dokumen; dan d f_t menunjukkan jumlah dokumen dengan istilah *t* di dalamnya.

Jika *ft,d* bernilai besar dan *ft,D* bernilai kecil, maka nilai dari log (|D|/ft,D) cenderung tinggi dan juga TF-IDF(t) akan bernilai tinggi. Kata dengan nilai TF-IDF(t) yang tinggi berarti bahwa kata tersebut merupakan sebuah kata yang sangat penting di dalam dokumen *d* tetapi tidak di dalam kumpulan dokumen D.

2.7 *N-gram*

Dalam literaturnya William B. Cavnar and John M. Trenkle (2001) menyebutkan bahwa N-gram adalah sebuah potongan dari sebuah kalimat panjang dalam literaturnya mereka hanya menggunakan irisan yang berdekatan saja, satu potongan kalimat menghasilkan satu set N-gram yang tumpang tindih, di sistemnya mereka juga menyebutkan bahwa mereka menggunakan N-gram dengan ukuran yang berbeda terus-menerus, mereka juga menambahkan karakter kosong (dengan menggunakan karakter garis bawah "_" untuk mepresentasikan karakter kosong) di awal dan di akhir kalimat untuk membantu mencocokan kalimat di awal dan di akhir, contoh dari penelitian mereka menggunakan kata "TEKS" yang akan terdiri dari N-gram berikut:

uni-gram: T, E, K, S

bi-gram : _T, TE, EK, KS, S_

tri-gram : _TE, TEK, EKS, KS_, dan S_ _

Dapat disimpulkan bahwa untuk string berukuran *n* akan dimiliki pada n *unigram* dan n+1 *bigram*, n+1 *trigram* dan seterusnya. Penggunaan N-gram untuk *matching* kata memiliki keuntungan sehingga dapat diterapkan pada recovery pada input karakter ASCII yang terkena *noise*, interpretasi kode pos, *information retrieval* dan berbagai aplikasi dalam pemrosesan bahasa alami. Dengan model bahasa bigram memproses bahasa dengan memecah kalimat menjadi beberapa bagian dimana tiap bagian terdiri dari dua kata terurut dari kalimat. Jika suatu kalimat dinyatakan dalam notasi {w_1, w_2, w_3..., w_{i-1}, w_i} dan *i* menyatakan banyak kata dalam kalimat, maka nilai peluang bigram dapat dihitung melalui persamaan berikut,

$$\boldsymbol{P(w_i|w_{i-1}) = \frac{N(w_{i-1}, w_i)}{N(w_{i-1})}}$$

Notasi P(w_i|w_{i-1}) menyatakan nilai peluang bigram kata ke-i, didahului oleh kata ke (i-1). Notasi N(w_{i-1}, w_i) menyatakan banyaknya kemunculan pasangan kata ke-i didahului oleh kata ke (i-1) pada korpus. Notasi N($_{wi-1}$) menyatakan banyaknya kemunculan kata ke (i-1) pada korpus. Sebuah kalimat dapat dihitung nilai peluangnya menggunakan model statistik bigram dengan menerapkan aturan Chain yang dijabarkan dalam Persamaan berikut.

$$\boldsymbol{P(W_1^n) \approx \prod_{i=1}^{n} P(w_i/w_{i-1})}$$

Notasi (W_1^n) Menyatakan nilai peluang kalimat yang dihitung menggunakan model statistik bigram. Notasi n menyatakan banyaknya pasangan bigram kata dalam sebuah kalimat. Contoh penerapan persamaan adalah sebagai berikut.

W = Saya makan nasi padang

Maka peluang bigram dari kalimat tersebut dapat dihitung dengan rumus.

P(W) = P(makan | saya) * P(nasi | makan) * P(padang | nasi*)*

2.8 *Performance Evaluation Matrix* (PEM)

Ahmad Arif Budiman (2018) menggunakan *Performance Evaluation Measure* (PEM) atau dalam Bahasa Indonesia bisa disebut pengukuran evaluasi performa sebagai evaluasi pengukuran dalam penelitiannya, PEM sendiri adalah sebuah tahapan yang digunakan untuk mengukur performa suatu sistem. PEM dalam banyak kasus digunakan dalam *pelatihan data*, tujuannya untuk mengevaluasi model yang sudah dibuat. Ada banyak perhitungan untuk mendapatkan nilai PEM, biasanya diterapkan sebagai kombinasi atau juga secara parsial. Beberapa perhitungan dalam PEM antara lain :

a) Presisi.

   Presisi adalah tingkat ketepatan antara request pengguna dengan jawaban sistem;

   Rumus presisi (*pre*) : $\boldsymbol{pre} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}}$

b) Akurasi.

   Akurasi adalah perbandingan antara informasi yang dijawab oleh sistem dengan benar dengan keseluruhan informasi; dan

Rumus akurasi (*acc*) : $$\boldsymbol{acc} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+FP+TN+FN}}$$

c) *Recall*.

Recall adalah ukuran ketepatan antara informasi yang sama dengan informasi yang sudah pernah dipanggil sebelumnya.

Rumus *recall* (*rec*) : $$\boldsymbol{rec} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}}$$

PEM biasanya digambarkan dalam confusion matrix, yaitu berupa tabel yang berisi hasil pengujian model yang telah dibandingkan dengan *dataset*, terdiri dari kelas true dan false.

Tabel 2.2 Confusion Matrix

	Predict	***ed Class***
True Class	***Positive***	***Negative***
Positive	TP	FN
Negative	FP	TN

Keterangan :

TP (true positive) : contoh data bernilai positif yang diprediksi benar sebagai positif

TN (true negative) : contoh data bernilai negatif yang diprediksi benar sebagai negatif

FP (false positive) : contoh data bernilai negatif yang diprediksi salah sebagai positif

FN (false negative) : contoh data bernilai positif yang diprediksi salah sebagai negatif

BAB III
METODE PENELITIAN

3.1 Studi Literatur

Studi Literatur pada penelitian ini dilakukan dengan membaca dan mengutip dari jurnal dan buku-buku yang terkait dengan penelitian ini. Proses yang dilakukan adalah mempelajari jurnal, karya ilmiah, atau buku yang berkaitan dengan penelitian seperti informasi tentang *Language Identification* dan Neural network (NN).

3.2 Persiapan Alat Penelitian

Tahapan pertama penelitian adalah memepersiapkan alat dan juga bahan penelitian seperti perangkat lunak (*Software*) dan perangkat keras (*Hardware*) yang akan digunakan untuk penelitian dan pengembangan perangkat lunak. Adapun perangkat keras dan perangkat lunak yang akan penulis gunakan adalah sebagai berikut :

3.2.1 Hardware

Perangkat keras (*Hardaware*) adalah alat yang akan digunakan untuk menunjang dalam pembuatan sistem/*engine*. Dalam pembuatan sistem ini, perangkat keras yang akan digunakan yaitu laptop dengan spesifikasi sebagai berikut :

1. Laptop Processor AMD A9-9420 @ 3.00 GHz
2. RAM 4 GB DDR4
3. HDD 1 TB

3.2.2 Software

Adapun kebutuhan perangkat lunak (*Software*) yang digunakan pada penelitian ini adalah sebagai berikut:

1. Sistem Operasi Windows 10 Pro.
2. IDLE *Python* 3.10 untuk penulisan program yang akan digunakan oleh penulis
3. Google Collab

3.3 Tahapan Penelitian

Tahapan penelitian adalah sebuah langkah yang dilakukan peneliti dalam melaksanakan penelitian. Berikut dibawah ini merupakan tahapan – tahapan penelitian yang dilakukan oleh penulis dapat dilihat pada gambar 3.1 :

Gambar 3.1 Bagan Tahapan Penelitian

3.4 Pengembangan Sistem

Pada penelitian ini sistem akan dibuat melalui 2 tahapan yaitu PreProcessing dan Processing. Dengan kedua tahapan tersebut, korpus bahasa lampung dialek "O" dan terjemahannya yang akan digunakan dapat diolah serta diproses kedalam model yang akan dibuat, dilatih maupun diuji.

3.4.1 Preprocessing

Tahapan ini sangatlah penting terutama pada proses pendeteksian kalimat, untuk meningkatkan kualitas kalimat yang akan dideteksi. Kalimat akan melalui proses *filtering* yaitu dimana kalimat dipisahkan dengan tanda baca serta *white space* yang terdapat pada masing-masing kalimat pada setiap korpus. Setelah itu kalimat yang telah melewati proses tersebut akan dibuat kedalam bentuk vektor. Pembuatan kedalam bentuk vektor ini dilakukan dengan membuat kalimat tersebut menjadi data yang berupa teks atau dapat disebut juga tokenisasi. Misalkan dengan suatu kalimat pada korpus bahasa lampung yang akan dibuat menjadi *array* pada saat *pelatihan data* sebagai berikut,

Kalimat korpus : "nikeu lapah adek masjid"

Dibuat menjadi : ['nikeu'],['lapah'],['adek'],['masjid']

Dengan banyaknya kata yang akan dilatih pada *model* akan dibentuk ke dalam vektor kata, atau menjadi kumpulan *array* dengan kalimat-kalimat yang berada pada korpus. Dengan adanya vektor tersebut dapat dijelaskan sebagai berikut:

Kumpulan kalimat : {['nikeu],['lapah'],['adek'],['masjid']},
{['saya'],['pergi'],['bekerja']},
{['kami'],['sedang'],['belajar']},
{['nikeu'],['jama'],['sapa']}

Dari kumpulan kalimat sebelumnya, dapat dibagi menjadi dua kategori yaitu bahasa Indonesia dan bahasa Lampung untuk dirubah kedalam bentuk matriks. Dengan keterangan bahwa D1 dan D2 merupakan kalimat bahasa Indonesia.

D1 : Kalimat 1 : Saya pergi bekerja

D2 : Kalimat 2 : kami sedang belajar

kemudian dirubah menjadi matriks seperti ini :

Tabel 3.1 Vektor Kata Bahasa Indonesia

Saya	pergi	bekerja	kami	sedang	belajar
1	1	1	0	0	0
0	0	0	1	1	1

Sedangkan untuk D3 dan D4 merupakan kalimat bahasa Lampung.

D3 : Kalimat 3 : nikeu lapah adek masjid

D4 : Kalimat 4 : nikeu jama sapa

kemudian dirubah menjadi matriks seperti pada tabel 3.2 Vektor Kata Bahasa Lampung

Tabel 3.2 Vektor Kata Bahasa Lampung

nikeu	lapah	adek	masjid	nikeu	Jama	Sapa
1	1	1	1	0	0	0
0	0	0	0	1	1	1

3.4.2 Processing

Langkah selanjutnya adalah ekstraksi fitur, agar mesin mudah untuk memahami data seperti manusia. Kalimat yang telah diubah menjadi vektor kata tersebut, diubah kedalam bentuk numerik untuk dilakukan ekstraksi fitur

menggunakan sistem pembobotan kata menggunakan *Term Frequency – Inverse Document Frequency* (TF-IDF).

Misalkan kita ingin menghitung bobot kata “nikeu” dari kalimat pertama, karena kata “nikeu” muncul satu kali pada dokumen pertama maka perhitungan bobot kata “nikeu” menjadi berikut :

$$W_{nikeu,d_1} = 1 \text{ x } (\log_{10}\left(\frac{4}{2}\right))$$

$$= 1 \text{ x } 0.301$$

$$= 0.301$$

Tabel 3.3 Perhitungan TF-IDF #1

Term(t)	**TFD1**	**TFD2**	**TFD3**	**TFD4**	**idf**
saya	1/3	0	0	0	log(4/1) = 0.602
pergi	1/3	0	0	0	log(4/1) = 0.602
bekerja	1/3	0	0	0	log(4/1) = 0.602
kami	0	1/3	0	0	log(4/1) = 0.602
sedang	0	1/3	0	0	log(4/1) = 0.602
belajar	0	1/3	0	0	log(4/1) = 0.602
nikeu	0	0	1/4	1/3	log(4/2) = 0.301
lapah	0	0	1/4	0	log(4/1) = 0.602
adek	0	0	1/4	0	log(4/1) = 0.602
masjid	0	0	1/4	0	log(4/1) = 0.602
jama	0	0	0	1/3	log(4/1) = 0.602
sapa	0	0	0	1/3	log(4/1) = 0.602

Dimana TFD-n merupakan banyak kata dalam dokumen/kalimat-n, maka matriks nilai TF-IDF yaitu perkalian TF dengan IDF dapat dilihat pada tabel 3.4:

Tabel 3.4 Perhitungan TF-IDF #2

Term(t)	**Tf.idf**			
	D1	**D2**	**D3**	**D4**
saya	0.206	0	0	0

Pergi	0.206	0	0	0
bekerja	0.206	0	0	0
Kami	0	0.206	0	0
sedang	0	0.206	0	0
belajar	0	0.206	0	0
Nikeu	0	0	0.075	0.103
Lapah	0	0	0.206	0
Adek	0	0	0.206	0
masjid	0	0	0.206	0
Jama	0	0	0	0.206
sapa	0	0	0	0.206

Setelah ekstraksi fitur didapatkan, maka dibentuk matriks dari fitur-fitur tersebut berukuran d x n dimana d merepresentasikan dokumen dan n merepresentasikan kata/*term*. Isi dari matriks TF-IDF inilah yang akan menjadi masukan untuk membangun *classifier Backpropagation* Neural network.

Prosedur pelatihan Backpropagation direpresentasikan secara sistematis sebagai berikut :

Kondisi 1 : Inisialisasi bobot

Peramban maju

Kondisi 2 :Tiap unit masukan (X_i , $i = 1,\ldots, n$) menerima sinyal X_i dan menghantarkan sinyal ini ke semua unit lapisan di atasnya (unit tersembunyi),

Kondisi 3 :Setiap unit tersembunyi (Z_i , $i = 1,\ldots, p$) jumlahkan bobot sinyal masukannya,

$$\boldsymbol{z_{inj} = v_0 + \sum_{i=1}^{n} x_i v_{ij}} \qquad \textbf{(3.1)}$$

V_{ij} = bias pada unit tersembunyi *j* menggunakan fungsi aktivasi *sigmoid biner* yang ditunjukkan pada Persamaan (3.2) untuk menghitung sinyal keluarannya dan mengirimkan sinyal ini keseluruh unit pada lapisan diatasnya (unit keluaran) ,

$$\boldsymbol{Z_j = f\,(z_{inj}) = \frac{1}{1+e^{-z_{inj}}}} \qquad \textbf{(3.2)}$$

Tiap unit keluaran (Y_k, *k = 1,..., m*) jumlahkan bobot sinyal masukannya,

$$\boldsymbol{y_{ink} = w_{0k} + \sum_{j=1}^{n} z_j w_{jk}} \qquad \textbf{(3.3)}$$

W_{jk} = bias pada unit keluaran *k* menggunakan fungsi aktivasi *sigmoid biner* yang ditunjukkan pada Persamaan (3.4) untuk menghitung sinyal keluarannya

$$\boldsymbol{Y_k = f\,(y_{ink}) = \frac{1}{1+e^{-y_{ink}}}} \qquad \textbf{(3.4)}$$

Perambatan Mundur

Kondisi 4 : Tiap unit keluaran (Y_k, *k = 1,..., m*) menerima pola target yang saling berhubungan pada masukan pola pelatihan, hitung kesalahan informasinya,

$$\boldsymbol{\delta_k = (t_k - y_k)\, f^1\,(y_{ink}) = (t_k - y_k)\, y_k\,(1 - y_k)} \qquad \textbf{(3.5)}$$

hitung koreksi bobotnya (digunakan untuk memperbaharui W_{jk} nantinya),

$$\boldsymbol{\Delta w_{jk} = \alpha \delta_k z_j} \qquad \textbf{(3.6)}$$

hitung koreksi biasnya (digunakan untuk memperbaharui W_{jk} nantinya), dan kirimkan δ_k ke unit-unit pada lapisan dibawahnya,

Kondisi 5 : Setiap unit lapisan tersembunyi (Z_j, $j = 1,\ldots, p$) jumlahkan hasil perubahan masukannya (dari unit-unit lapisan diatasnya),

$$\Delta_{in\,j} = \sum_{k=1}^{n} \delta_k w_{jk} \quad \textbf{(3.7)}$$

kalikan dengan turunan fungsi aktivasinya untuk menghitung informasi kesalahannya,

$$\delta_j = \delta_{inj} f^1(z_{inj}) = \delta_{inj}\, Z_j\,(1 - Z_j) \quad \textbf{(3.8)}$$

hitung koreksi bobotnya (digunakan untuk memperbaharui V_{ij} nanti),

Kondisi 6 : Tiap unit keluaran (Y_k, $k = 1,\ldots, m$) update bias dan bobotnya ($j = 0,\ldots, p$) :

$$W_{jk} \leftarrow W_{jk} + \Delta W_{jk} \quad \textbf{(3.9)}$$

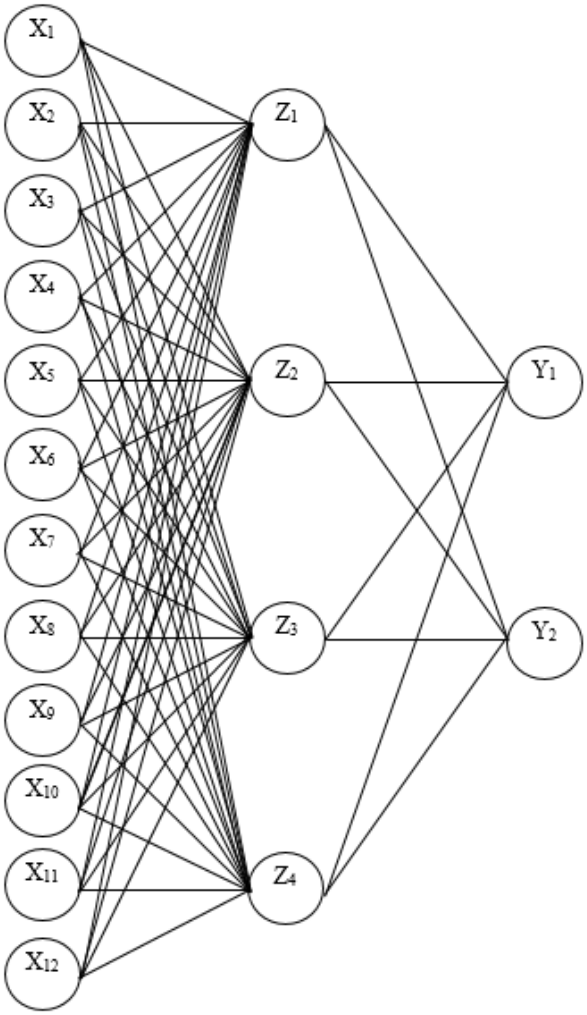
Tiap unit lapisan tersembunyi (Z_j, $j = 1,\ldots, p$) update bias dan bobotnya (i = $1,\ldots,n$) :

Kondisi 7 : Test kondisi berhenti.

Kondisi 8 : Jika kondisi henti tidak tercapai, maka ulangi kondisi 2 – 7.

Kondisi 9 : Untuk setiap pasangan data pelatihan, lakukan langkah 2 – 8.

Contoh perhitungan sederhana algoritma backpropagation sesuai dengan perhitungan sebelumnya pada TFIDF dengan dua target output, arsitektur jaringan yang digunakan ditunjukkan pada gambar 3.2



Gambar 3.2 Struktur jaringan yang digunakan

Pada model arsitektur yang ditunjukkan, X_1 hingga X_{12} adalah atribut yang akan digunakan untuk mengklasifikasikan kalimat baru. Z1 hingga Z4 adalah neuron yang terdapat pada layer tersembunyi. Y1 dan Y2 adalah keluaran yang akan dibandingkan untuk menentukan nilai true ataupun false (apakah nilai tersebut masuk kedalam kategori bahasa Lampung ataupun bahasa Indonesia.

Tabel 3.5 menunjukkan bagaimana ilustrasi Vij yang menggambarkan bobot dari input layer ke layer tersembunyi (*hidden layer*) dengan menggunakan 4 *neuron* pada layer tersembunyi.

Tabel 3.5 Bobot Vij #1

	Z_1	Z_2	Z_3	Z_4
X_1	$V_{1.1}$	$V_{1.2}$	$V_{1.3}$	$V_{1.4}$
X_2	$V_{2.1}$	$V_{2.2}$	$V_{2.3}$	$V_{2.4}$
X_3	$V_{3.1}$	$V_{3.2}$	$V_{3.3}$	$V_{3.4}$
X_4	$V_{4.4}$	$V_{4.2}$	$V_{4.3}$	$V_{4.4}$
X_5	$V_{5.1}$	$V_{5.2}$	$V_{5.3}$	$V_{5.4}$
X_6	$V_{6.1}$	$V_{6.2}$	$V_{6.3}$	$V_{6.4}$
X_7	$V_{7.1}$	$V_{7.2}$	$V_{7.3}$	$V_{7.4}$
X_8	$V_{8.1}$	$V_{8.2}$	$V_{8.3}$	$V_{8.4}$
X_9	$V_{9.1}$	$V_{9.2}$	$V_{9.3}$	$V_{9.4}$
X_{10}	$V_{10.1}$	$V_{10.2}$	$V_{10.3}$	$V_{10.4}$
X_{11}	$V_{11.1}$	$V_{11.2}$	$V_{11.3}$	$V_{11.4}$
X_{12}	$V_{12.1}$	$V_{12.2}$	$V_{12.3}$	$V_{12.4}$

Tabel 3.6 menunjukkan ilustrasi Wjk tentang bagaimana bobot layer tersembunyi yang terhubung ke label output atau hasil klasifikasi dari layer tersembunyi (*hidden layer*) dengan menggunakan 4 *neuron*.

Tabel 3.6 Bobot Wjk #1

	Y_1	Y_2
Z_1	$W_{1.1}$	$W_{1.2}$
Z_2	$W_{2.1}$	$W_{2.2}$
Z_3	$W_{3.1}$	$W_{3.2}$
Z_4	$W_{4.1}$	$W_{4.2}$

Pertama, proses yang harus dilakukan dalam algoritma Backpropagation adalah menetapkan semua bobot, yaitu Vij yang dijelaskan dalam Tabel 3.7 dan Wjk yang dijelaskan dalam Tabel 3.8 dengan menggunakan bilangan acak kecil yang berada dalam rentang [-1, 1].

Tabel 3.7 Bobot Vij #2

	Z_1	Z_2	Z_3	Z_4
1	-0,3	0,3	0,3	0,2
X_1	0,2	0,3	-0,1	0,1
X_2	0,3	0,1	-0,1	-0,3
X_3	0,2	0,3	-0,1	0,1

X_4	0,3	0,1	-0,1	-0,3
X_5	0,1	0,3	0,3	0,2
X_6	0,2	0,3	-0,1	0,1
X_7	0,3	0,1	-0,1	-0,3
X_8	0,1	0,3	0,3	0,2
X_9	0,2	0,3	-0,1	0,1
X_{10}	0,1	0,3	0,3	0,2
X_{11}	0,2	0,3	-0,1	0,1
X_{12}	0,3	0,1	-0,1	-0,3

Tabel 3.8 Bobot W_{jk} #2

	Y_1	Y_2
1	-0,1	0,1
Z_1	0,5	0,3
Z_2	0,6	0,3
Z_3	-0,4	0,1
Z_4	0,5	0,3

Proses berikutnya setelah menentukan bobot yaitu menghitung output pada *hidden layer* (Zj) menggunakan Persamaan (3.1).

Z_net 1 = -0,3 + 0,2(0) + 0,3(0) + 0,2(0) + 0,3(0) + 0,1(0) + 0,2(0) +

0,3(0,075) + 0,1(0,206) + 0,2(0,206) + 0,1(0,206) +0,2(0) + 0,3(0)

= -0,20

Z_net 2 = 0,3 + 0,3(0) + 0,1(0) + 0,3(0) + 0,1(0) + 0,3(0) + 0,3(0) +

0,1(0,075) + 0,3(0,206) + 0,3(0,206) + 0,3(0,206) + 0,3(0) + 0,1(0)

= 0,43

Z_net 3 = 0,3 + -0,1(0) + -0,1(0) + -0,1(0) + -0,1(0) + 0,3(0) + -0,1(0) +

-0,1(0,075) + 0,3(0,206) + -0,1(0,206) + 0,3(0,206) + -0,1(0) + -0,1(0)

= 0,40

Z_net 4 = 0,2 + 0,1(0) + -0,3(0) + 0,1(0) + -0,3(0) + 0,2(0) + 0,1(0) +

-0,3(0,075) + 0,2(0,206) + 0,1(0,206) + 0,2(0,206) + 0,1(0) + -0,3(0)

= 0,28

$Z_1 = 1 / (1+ e^{0,20}) = 0,45$

$Z_2 = 1 / (1+ e^{-0,43}) = 0,61$

$Z_3 = 1 / (1+ e^{-0,40}) = 0,60$

$Z_4 = 1 / (1+ e^{-0,28}) = 0,57$

Proses berikutnya sesudah menghitung output pada *hidden layer* (Zj) adalah menghitung unit (Y_k) menggunakan persamaan (3.3)

Y_net 1 = -0,1 + 0,45(0,5) + 0,61(0,6) + 0,60(-0,4) + 0,57(0,5) = 0,54

Y_net 2 = 0,1 + 0,45(0,3) + 0,61(0,3) + 0,60(0,1) + 0,57(0,3) = 0,64

$Y_1 = 1 / (1+ e^{-0,54}) = 0,63$

$Y_2 = 1 / (1+ e^{-0,64}) = 0,66$

Setelah didapatkan Y_k, dihitung faktor δ pada *layer output* Y_k menggunakan Persamaan (3.5)

$\delta\_ Y_{k1} = (0 – 0,63)(0,63)(1 – 0,63) = -0,233$

$\delta\_ Y_{k2} = (0 – 0,66)(0,66)(1 – 0,66) = -0,224$

Setelah didapatkan faktor δ pada *layer output* Y_k, dihitung Suku perubahan bobot W_{jk} (dengan $\alpha = 0,2$) menggunakan Persamaan (3.6)

$\Delta W_{01} = 0,2 (-0,233) (1) = -0,047$

$\Delta W_{11} = 0,2 (-0,233) (0,45) = -0,021$

$\Delta W_{21} = 0,2 (-0,233) (0,61) = -0,029$

$\Delta W_{31} = 0,2 (-0,233) (0,60) = -0,028$

$\Delta W_{41} = 0,2 (-0,233) (0,57) = -0,027$

ΔW_{02} = 0,2 (-0,224) (1) = -0,045

ΔW_{12} = 0,2 ((-0,224) (0,45) = -0,020

ΔW_{22} = 0,2 ((-0,224) (0,61) = -0,028

ΔW_{32} = 0,2 ((-0,224) (0,60) = -0,027

ΔW_{42} = 0,2 ((-0,224) (0,57) = -0,026

Setelah didapatkan Suku perubahan bobot W_{jk}, dilanjutkan Menghitung jumlah kesalahan pada *hidden layer* (δ) menggunakan Persamaan (3.7)

δ_net 1 = (-0,233 * 0,5) + (-0,224 * 0,3) = -0,1837

δ_net 2 = (-0,233 * -0,3) + (-0,224 * 0,3) = 0,0027

δ_net 3 = (-0,233 * -0,4) + (-0,224 * 0,1) = 0,0708

δ_net 4 = (-0,233 * 0,5) + (-0,224 * 0,3) = -0,1837

Setelah didapatkan jumlah kesalahan pada *hidden layer* (δ), Selanjutnya menghitung faktor kesalahan δ pada *hidden layer* menggunakan Persamaan (3.8).

δ_1 = -0,1837 (0,45) (1-0.45) = -0,04546

δ_2 = 0,0027 (0,61) (1-0,61) = 0,00064

δ_3 = 0,0708 (0,60) (1-0,60) = 0,016992

δ_4 = -0,1837 (0,57) (1-0,57) = -0,04502

Suku perubahan bobot V_{ij} (dengan $\alpha = 0,2$) ditunjukkan pada Tabel 3.9

Tabel 3.9 Suku perubahan bobot V_{ij}

	Z_1	Z_2	Z_3	Z_4
1	(0,2) (-0,04546) (1) = 0,00909 ≈ 0	(0,2) (-0,00064) (1) = 0,00013 ≈ 0	(0,2) (0,016992) (1) = 0,00340 ≈ 0	(0,2) (-0,04502) (1) = 0,00900 ≈ 0
X_1	(0,2) (-0,04546) (0,206) = 0,00187 ≈ 0	(0,2) (-0,00064) (0) = 0	(0,2) (0,016992) (0) = 0	(0,2) (-0,04502) (0) = 0
X_2	(0,2) (-0,04546) (0,206) = 0,00187 ≈ 0	(0,2) (-0,00064) (0) = 0	(0,2) (0,016992) (0) = 0	(0,2) (-0,04502) (0) = 0
X_3	(0,2) (-0,04546) (0,206) = 0,00187 ≈ 0	(0,2) (-0,00064) (0) = 0	(0,2) (0,016992) (0) = 0	(0,2) (-0,04502) (0) = 0
X_4	(0,2) (-0,04546) (0) = 0	(0,2) (-0,00064) (0,206() = 0,00003 ≈ 0	(0,2) (0,016992) (0) = 0	(0,2) (-0,04502) (0) = 0
X_5	(0,2) (-0,04546) (0) = 0	(0,2) (-0,00064) (0,206() = 0,00003 ≈ 0	(0,2) (0,016992) (0) = 0	(0,2) (-0,04502) (0) = 0
X_6	(0,2) (-0,04546) (0) = 0	(0,2) (-0,00064) (0,206() = 0,00003 ≈ 0	(0,2) (0,016992) (0) = 0	(0,2) (-0,04502) (0) = 0
X_7	(0,2) (-0,04546) (0) = 0	(0,2) (-0,00064) (0) = 0	(0,2) (0,016992) (0,075) = 0,00026 ≈ 0	(0,2) (-0,04502) (0,103) = 0,00093 ≈ 0
X_8	(0,2) (-0,04546) (0) = 0	(0,2) (-0,00064) (0) = 0	(0,2) (0,016992) (0,206) = 0,00070 ≈ 0	(0,2) (-0,04502) (0) = 0
X_9	(0,2) (-0,04546) (0) = 0	(0,2) (-0,00064) (0) = 0	(0,2) (0,016992) (0,206) = 0,00070 ≈ 0	(0,2) (-0,04502) (0) = 0
X_{10}	(0,2) (-0,04546) (0) = 0	(0,2) (-0,00064) (0) = 0	(0,2) (0,016992) (0,206) = 0,00070 ≈ 0	(0,2) (-0,04502) (0) = 0
X_{11}	(0,2) (-0,04546) (0) = 0	(0,2) (-0,00064) (0) = 0	(0,2) (0,016992) (0) = 0	(0,2) (-0,04502) (0,206) = 0,00185 ≈ 0
X_{12}	(0,2) (-0,04546) (0) = 0	(0,2) (-0,00064) (0) = 0	(0,2) (0,016992) (0) = 0	(0,2) (-0,04502) (0,206) = 0,00185 ≈ 0

Proses terakhir adalah menghitung semua perubahan bobot yang terjadi pada V_{ij} yang ditunjukkan pada Tabel 3.10 dan perubahan bobot yang terjadi pada W_{jk} yang ditunjukkan pada Tabel 3.11.

Tabel 3.10 Perubahan bobot V_{ij}

	Z_1	Z_2	Z_3	Z_4
1	-0,3 + 0 = -0.3	0,3 + 0 = 0,3	0,3 + 0 = 0,3	0,2 + 0 = 0,2
X_1	0,2 + 0 = 0,2	0,3 + 0 = 0,3	-0,1 + 0 = -0,1	0,1 + 0 = 0,1
X_2	0,3 + 0 = 0,3	0,1 + 0 = 0,1	-0,1 + 0 = -0,1	-0,3 + 0 = -0,3
X_3	0,2 + 0 = 0,2	0,3 + 0 = 0,3	-0,1 + 0 = -0,1	0,1 + 0 =0,1
X_4	0,3 + 0 = 0,3	0,1 + 0 = 0,1	-0,1 + 0 = -0,1	-0,3 + 0 = -0,3
X_5	0,1 + 0 = 0,1	0,3 + 0 = 0,3	0,3 + 0 = 0,3	0,2 + 0 = 0,2
X_6	0,2 + 0 = 0,2	0,3 + 0 = 0,3	-0,1 + 0 = 0,1	0,1 + 0 = 0,1
X_7	0,3 + 0 = 0,3	0,1 + 0 = 0,1	-0,1 + 0 = -0,1	-0,3 + 0 = -0,3
X_8	0,1 + 0 = 0,1	0,3 + 0 = 0,3	0,3 + 0 = 0,3	0,2 + 0 = 0,2
X_9	0,2 + 0 = 0,2	0,3 + 0 = 0,3	-0,1 + 0 = -0,1	0,1 + 0 = 0,1
X_{10}	0,1 + 0 = 0,1	0,3 + 0 = 0,3	0,3 + 0 = 0,3	0,2 + 0 = 0,2
X_{11}	0,2 + 0 = 0,2	0,3 + 0 = 0,3	-0,1 + 0 = -0,1	0,1 + 0 = 0,1
X_{12}	0,3 + 0 = 0,3	0,1 + 0 = 0,1	-0,1 + 0 = -0,1	-0,3 + 0 = -0,3

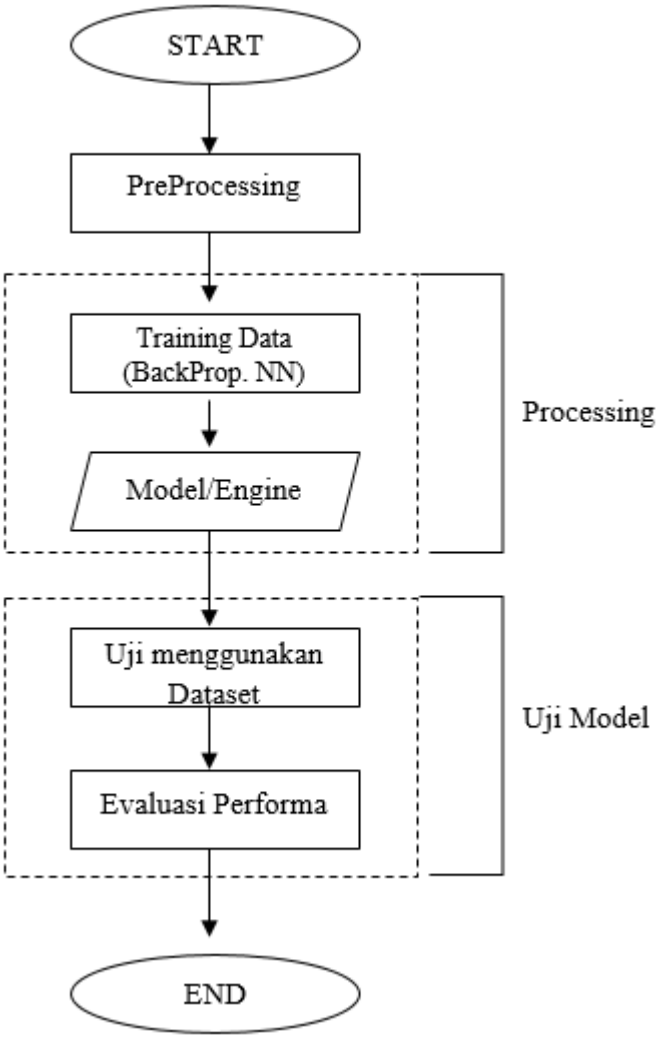
Tabel 3.11 Perubahan bobot W_{jk}

	Y_1	Y_2
1	-0,1 + -0,046 = -0,146	0,1 + -0,045 = 0,055
Z_1	0,5 + -0,039 = 0,461	0,3 + -0,039 = 0,261
Z_2	0,6 + -0,045 = 0,555	0,3 + -0,045 = 0,255
Z_3	-0,4 + -0,007 = -0,407	0,1 + -0,006 = 0,094
Z_4	0,5 + -0,017 = 0,483	0,3 + 0,016 = 0,316

Proses tersebut dilakukan berulang hingga pada data terakhir, baru didapatkan satu *epoch*.

3.5 Diagram Alir Sistem

Aliran proses deteksi kata pada program dengan suatu masukan diperlihatkan pada gambar. Pada proses "*Preprocessing*" ditunjukan bahwa akan adanya data yang dilatih pada model berupa korpus yaitu korpus bahasa Lampung dan bahasa Indonesia. Dengan korpus ini akan terbentuk model atau model yang akan diuji dengan data set sehingga menghasilkan model detection language dengan menilai melalui performa yang akan ditampilkan pada evaluasinya. Dapat dilihat pada gambar 3.3 Diagram Alir Sistem.



Gambar 3.3 Diagram alir sistem

BAB IV
IMPLEMENTASI

Model atau *engine* pendeteksi bahasa Indonesia dan Lampung, yang menggunakan *Python* dan library pendukungnya untuk menghasilkan model yang dibuat dalam beberapa langkah, merupakan hasil dari implementasi pengembangan penelitian ini. Dengan memberikan label terlebih dahulu terhadap korpus yang akan digunakan sebagai *dataset*. *Dataset* tersebut kemudian digunakan sebagai data pelatihan untuk model baru yang akan dibuat. Selanjutnya model diuji dengan *Performance Evaluation Measure* (PEM) untuk menentukan seberapa akurat model tersebut dapat mendeteksi proses setelahnya.

5.1 Pelabelan *Dataset*

Sebelum dapat digunakan untuk pembangunan model setiap korpus akan mengalami tahapan preprocessing. Setiap korpus akan diberi label yang sesuai dengan jenisnya, dengan menggunakan 2 kategori yaitu Lampung dan Indonesia.

Korpus pertama dilabeli dengan bahasa Lampung, sedangkan korpus kedua dilbeli bahasa Indonesia. Pelabelan data dapat dilihat pada gambar 4.1

```
lampung_df = pd.read_csv('gdrive/My
Drive/Scriptsweet/bhn_lpg.csv', sep=';',
encoding='unicode_escape')
lampung_df.columns = ["words"]
lampung_df['label'] = "lampung"

indonesia_df = pd.read_csv('gdrive/My
Drive/Scriptsweet/bhn_ind.csv', sep=';',
encoding='unicode_escape')
indonesia_df.columns = ["words"]
indonesia_df['label'] = "indonesia"
```

Gambar 4.1 Pelabelan *dataset*

Korpus dipanggil untuk dibaca dari drive menggunakan fungsi **pd.read_csv** kemudian disimpan dalam dataframe **'lampung_df'** dan **'indonesia_df'**. Setelah itu ditambahkan kolom baru dengan nama "label" menggunakan fungsi **lampung_df['label']** dengan nilai **"lampung"** untuk yang berbahasa Lampung, dan **"indonesia"** untuk yang berbahasa Indonesia.

Kedua korpus tersebut digabungkan menggunakan fungsi **concat** pada dataframe dan ditransformasikan menjadi kalimat-kalimat sederhana menggunakan fungsi **case_folding(text)** yang didalamnya terdapat proses filtering meliputi **strip()** untuk menghapus spasi pada teks, **re.sub(…)** untuk menghapus karakter khusus selain alphabet dan numerik, **str.maketrans(…)** untuk menghapus tanda baca dari teks, serta **lower()** untuk mengubah semua huruf menjadi huruf kecil. Kemudian disimpan kembali dalam bentuk csv yang nantinya digunakan dalm pembuatan model. Proses tersebut dapat dilihat pada gambar 4.2 proses pelabelan, filtering dan penggabungan data sedangkan pada gambar 4.3 merupakan hasil akhir pelabelan, filtering dan penggabungan data

```
def case_folding(text):
  text = text.strip()
  text = re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z\t])|(\w+:\/\/\S+)","
",text)
  text = text.translate(str.maketrans("","",string.punctuation))
  text = text.lower()
  return text

dataframe_df = pd.concat([lampung_df, indonesia_df])
dataframe_df["words"] = dataframe_df["words"].apply(lambda x:
case_folding(x))
dataframe_df = dataframe_df.sample(frac=1).reset_index(drop=True)
dataframe_df

dataframe_df.to_csv('gdrive/My Drive/Scriptsweet/bhn_gabung.csv',
sep=';', encoding='unicode_escape',index=False)
```

Gambar 4.2 Pelabelan, filtering dan penggabungan data

	words	label
0	lamun nikeu abangkeu , ulahnyo nikeu jadei ula...	lampung
1	naga ngejawab lamun yo munih mak pandai , yo n...	lampung
2	ranau mak agow mulang , yo ghabai sayan di nuw...	lampung
3	yo nutuk adekkedow gaweh naga lapah	lampung
4	nikeu agow kututuk , matei bareng matei , ughi...	lampung
...	...	...
9071	Karena orang ke ladang	indonesia
9072	Kalau malam ya	indonesia
9073	Orang ada di rumah	indonesia
9074	Orang ada ladang semua	indonesia
9075	Alhamdulillah selesai semua tugas yang diberik...	indonesia

18152 rows × 2 columns

Gambar 4.3 Hasil pelabelan, filtering dan penggabungan data

5.2 Pembuatan Model

Pada titik ini, proses pembuatan model menggunakan *package/library* bernama *Scikit-Learn* yang tersedia secara gratis dan bebas digunakan untuk kebutuhan komputasi dan *Machine Learning*. *Sklearn* sudah mendukung banyak fungsi seperti *N-gram*, *Backpropagation NN* dan *TFIDF Vectorrizer*, sehingga penulis hanya perlu menggunakan fungsi bawaannya. Dalam perhitungan algoritma backpropagation, implementasinya seringkali tidak terlihat secara langsung karena prosesnya menggunakan model yang telah dibangun sebelumnya.

```
mix_df = pd.read_csv('gdrive/My Drive/Scriptsweet/bhn_gabung.csv',
sep=';', encoding='unicode_escape')
mix_df.head()

#pembuatan model
mix_df['label'].unique()

lb = LabelBinarizer()
X = mix_df['words']
y = lb.fit_transform(mix_df['label']).flatten()

X_train, X_test, y_train, y_test = train_test_split(X ,y,
random_state=42, test_size=0.2)
```

Gambar 4.4 Pembagian data latih dan data uji

Pada gambar 4.4 menunjukan bahwa data yang dipanggil kembali dari drive menggunakan **`pd.read_csv`** yand disimpan dalam dataframe **`mix_df`.** Kemudian, digunakan **`LabelBinarizer`** untuk mengubah label kategorikal menjadi representasi biner. Variabel **`y`** akan berisi target yang telah diubah.

Selanjutnya, dilakukan pemisahan data menjadi data latih dan data uji menggunakan fungsi **`train_test_split`** dari pustaka **`sklearn.model_selection`**. Data latih akan terdiri dari 80% (14.523 korpus) data awal (**X_train** dan **y_train**), sedangkan data uji akan terdiri dari 20% (3.631 korpus) data awal (**`X_test`** dan **`y_test`**). Angka **42** digunakan sebagai biji acak (**`random seed`**) untuk memastikan reproduktibilitas hasil yang sama setiap kali kode dijalankan.

```
#perhitungan N-Gram & Backprop NN
count_vectorizer = TfidfVectorizer(min_df=5, ngram_range=
(1,3)).fit(X_train)
X_train_vectorized = count_vectorizer.transform(X_train)

model = MLPClassifier(hidden_layer_sizes=(200,100,),
max_iter=1000).fit(X_train_vectorized, y_train)
```

Gambar 4.5 Representasi fitur dan pembuatan model

Setelah pembagian data, seperti pada gambar 4.5 dilakukan penghitungan representasi fitur menggunakan N-Gram. Digunakan **`TfidfVectorizer`** dengan parameter **`min_df=5`** untuk mengabaikan kata-kata yang muncul kurang dari 5 kali, dan **`ngram_range=(1,3)`** untuk menghasilkan N-Gram dari 1 hingga 3 kata. **`X_train`** diubah menjadi representasi vektor menggunakan metode transform pada objek **`count_vectorizer`**, dan hasilnya disimpan dalam variabel **`X_train_vectorized`**. Selanjutnya, pembuatan model jaringan saraf tiruan (neural network) menggunakan **`MLPClassifier`**. Model ini memiliki dua lapisan tersembunyi dengan ukuran **`200`** dan **`100`** unit masing-masing, dan akan

dilatih dengan maksimum **1000** iterasi. Model ini di-fit ke **X_train_vectorized** dan **y_train**.

Adapun *N-gram* dari setiap korpus dapat dilihat pada gambar 4.5

```
def find_ngrams(input_list, n):
    return list(zip(*[input_list[i:] for i in range(n)]))

df['unigram'] = df['teks'].map(lambda x: find_ngrams(x.split(" "), 1))
df['bigram'] = df['teks'].map(lambda x: find_ngrams(x.split(" "), 2))
df['trigram'] = df['teks'].map(lambda x: find_ngrams(x.split(" "), 3))
df.head()
```

	teks	unigram	bigram	trigram
0	ulahnyo nyak ghegeh sijo katteu sijo ulah nyak...	[(ulahnyo,), (nyak,), (ghegeh,), (sijo,), (kat...	[(ulahnyo, nyak), (nyak, ghegeh), (ghegeh, sij...	[(ulahnyo, nyak, ghegeh), (nyak, ghegeh, sijo)...
1	lamun nikeu abangkeu ulahnyo nikeu jadei ulai ...	[(lamun,), (nikeu,), (abangkeu,), (ulahnyo,), ...	[(lamun, nikeu), (nikeu, abangkeu), (abangkeu,...	[(lamun, nikeu, abangkeu), (nikeu, abangkeu, u...
2	naga ngejawab lamun yo munih mak pandai yo nga...	[(naga,), (ngejawab,), (lamun,), (yo,), (munih...	[(naga, ngejawab), (ngejawab, lamun), (lamun, ...	[(naga, ngejawab, lamun), (ngejawab, lamun, yo...
3	ranau mak agow mulang yo ghabai sayan di nuwo ...	[(ranau,), (mak,), (agow,), (mulang,), (yo,), ...	[(ranau, mak), (mak, agow), (agow, mulang), (m...	[(ranau, mak, agow), (mak, agow, mulang), (ago...
4	yo nutuk adekkedow gaweh naga lapah	[(yo,), (nutuk,), (adekkedow,), (gaweh,), (nag...	[(yo, nutuk), (nutuk, adekkedow), (adekkedow, ...	[(yo, nutuk, adekkedow), (nutuk, adekkedow, ga...

Gambar 4.6 N-gram yang telah dibuat dari setiap korpus

Model kembali diperiksa kesesuaiannya dengan hasil pemrosesan secara manual setelah dilakukan beberapa proses pelatihan. Model akan memprediksi kalimat yang diberikan, dimana kalimat tersebut berasal dari korpus yang ada dalam Gambar 4.7 dilakukan pengecekan model yang dapat dilihat dibawah ini.

```
kata = [
    "nyak mengan di nuwo minak",
    "kamu pergi kemana saja",
    "dang ghegeh ino lamun ngaman dijo"
]
kata = count_vectorizer.transform(kata)
test = model.predict(kata)
print(lb.inverse_transform(test))
```

```
['lampung' 'indonesia' 'lampung']
```

Gambar 4.7 Pengecekan model

Dengan menggunakan *joblib* yang merupakan salah satu fungsi di *python* sebagai perantara pembuatan model, hasil dari model tersebut akan disimpan.

Kemudian akan digunakan kembali dalam pembuatan UI pada model. Yang dapat dilihat pada Gambar 4.8 Penyimpanan Model

```
saved_count_vectorizer = joblib.dump(count_vectorizer, 'gdrive/My
Drive/Scriptsweet/count_vectorizer.joblib')
saved_model = joblib.dump(model, 'gdrive/My Drive/Scriptsweet/model.joblib')
```

Gambar 4.8 Penyimpanan model

5.3 *Pengujian Performance Evaluation Measure* (PEM)

Kemudian model yang telah diperoleh akan diuji menggunakan PEM untuk melihat apakah model bekerja seperti yang diinginkan penulis. Pengujian *Performance Evaluation Measure* menghasilkan *confusion matrix* yang berisi nilai akurasi, presisi, *recall* dan *f1-Score* yang digunakan sebagai 4 nilai evaluasi performa dari suatu sistem. Hasil pengujian PEM dapat terlihat pada gambar 4.9

```
akurasi :  0.9754888460479206
presisi :  0.9808533916849015
recall  :  0.9707634001082837

Confusion Matrix
 [[1749   35]
 [  54 1793]]

Classification Report
              precision    recall  f1-score   support

           0       0.97      0.98      0.98      1784
           1       0.98      0.97      0.98      1847

    accuracy                           0.98      3631
   macro avg       0.98      0.98      0.98      3631
weighted avg       0.98      0.98      0.98      3631
```

Gambar 4.9 Pengujian PEM

Dari hasil pengujian pada gambar 4.9 tersebut diketahui bahwa *Confusion matrix* menunjukkan jumlah prediksi yang benar dan salah untuk setiap kelas. Dalam kasus ini angka 0 merupakan kelas untuk kategori bahasa Lampung dan angka 1 untuk kelas dalam kategori bahasa Indonesia. Sehingga terdapat 1784

prediksi untuk bahasa Lampung dengan rincian 1749 prediksi yang benar dan 35 prediksi yang salah. Sedangkan untuk bahasa Indonesia terdapat 1847 prediksi dengan rincian 54 prediksi yang salah dan 1793 prediksi yang benar. Pada bagian *Classification report* memberikan informasi yang lebih detail tentang presisi, *recall*, dan *f1-score* untuk setiap kelas. Pada angka 0 (Lampung), presisi adalah 0.97, *recall* adalah 0.98, dan *f1-score* adalah 0.98. Kemudian pada kelas 1 (Indonesia), presisi adalah 0.98, *recall* adalah 0.97, dan *f1-score* adalah 0.98.

5.4 Pembuatan UI (*User Interface*)

Langkah selanjutnya dalam penelitian ini adalah pembuatan UI secara sederhana dengan menyediakan sebuah *fill box* untuk memasukkan kalimat yang akan dideteksi dan tombol deteksi, yang mana hasil dari pemrosesan model akan ditampilkan dengan label Bahasa Indonesia ataupun Bahasa Lampung untuk *Engine* pendeteksi bahasa. Berikut *source code* pada gambar 4.10 dan 4.11

```
cv = joblib.load('gdrive/My Drive/Scriptsweet/count_vectorizer.joblib')
model = joblib.load('gdrive/My Drive/Scriptsweet/model.joblib')

def preprocess_text(text):
    text = text.translate(str.maketrans("", "", string.punctuation))
    text = re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z\t])|(\w+:\/\/\S+)"," ",text)
    text = re.sub(r'[^ -~]', '', text)
    text = text.lower()
    return text
```

Gambar 4.10 SC interface preprocessing

Pada *source code* diatas model akan dipanggil kembali dari *drive* yang terhubung menggunakan fungsi **`joblib.load`**. Setelah model didapatkan, akan dilakukan preprocessing pada kalimat yang diinputkan dari *fill box* untuk menghilangkan *whitespace*, tanda baca dan merubah huruf menjadi huruf kecil.

```
def language_cls(text,n=3):
    text = cv.transform([preprocess_text(text)])
    result = 'indonesia' if model.predict(text)[0] == 0 else 'lampung'
    return result

intfc = gr.Interface(
    fn=language_cls,
    inputs=gr.Textbox(label="Input here", lines=2, placeholder="Input your
text"),
    outputs= gr.Label(label='Bahasa')
)

intfc.launch(debug=True)
```

Gambar 4.11 SC interface model *engine* deteksi

Source code diatas akan menampilkan *fill box* untuk diinputkan kalimat menggunakan **`gr.Textbox`** yang akan di deteksi dengan mendefinisikan terlebih dahulu fungsi **`language_cls`** berisi argument **`text`** untuk teks yang akan diklasifikasikan. Di dalam fungsi **`language_cls`**, teks tersebut diubah kedalam bentuk vektor menggunakan **`cv.transform`** setelah melalui preprocessing. Setelah mendapatkan representasi vektor dari **`text`**, dilakukan prediksi menggunakan model yang telah dilatih sebelumnya (**`model`**). Jika hasil prediksi (**`model.predict(text)`**) adalah 0, maka teks tersebut diklasifikasikan sebagai bahasa Indonesia. Jika bukan 0, maka teks tersebut diklasifikasikan sebagai bahasa Lampung. Hasil klasifikasi disimpan dalam variabel **`result`**. Kemudian tombol deteksi untuk memulai *engine* mendeteksi kalimat yang telah diinputkan sebelumnya. Kemudian hasil kalimat yang telah dideteksi akan ditampilkan dengan label bahasa Indonesia ataupun bahasa Lampung menggunakan **`gr.Label`**.

Dalam penelitian ini bukan hanya mesin pendeteksi bahasa yang dihasilkan, tetapi juga menghasilkan mesin *n-gram*, yang merupakan sebuah *engine* untuk memenggal kalimat menjadi beberapa potong kata. Sama seperti sebelumnya mesin *n-gram* ini dibuatkan UI-nya juga secara sederhana dengan menyediakan *fill box* menggunakan **`gr.inputs.Textbox`** untuk memasukkan kalimat yang

akan dipenggal menjadi potongan kata, kemudian jumlah potongan kata yang diinginkan menggunakan **`gr.inputs.Number`**. Sehingga hasil kalimat yang telah dipenggal akan ditampilkan dalam bentuk tabel dengan kolom sesuai jumlah potongan yang kita inginkan.

Pada bagian ini sama seperti sebelumnya yaitu dilakukan preprocessing terlebih dahulu kepada kalimat yang diinputkan dari *fill box* untuk menghilangkan whitespace, tanda baca dan merubah huruf menjadi huruf kecil.

```
def ngrams(text,n):
    text = preprocess_text(text).split(" ")
    new_n=math.floor(n)
    range(new_n)
    # print(output)
    return list(zip(*[text[i:] for i in range(new_n)]))
```

```
intfc = gr.Interface(
    fn=ngrams,
    inputs=[gr.inputs.Textbox(label="Masukan
kalimat"),gr.inputs.Number(label="jumlah ngrams")],
    outputs= gr.Dataframe()
)

intfc.launch(debug=True)
```

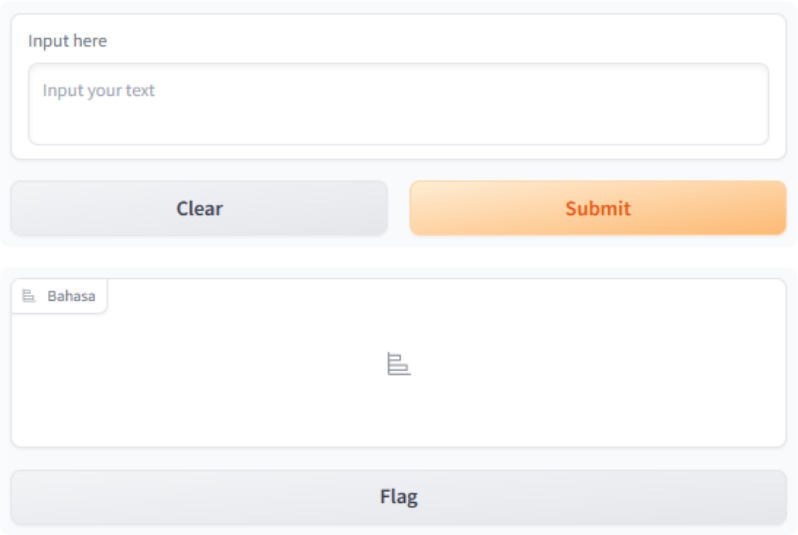
Gambar 4.12 SC interface *engine n-gram* model

Setelah dilakukan preprocessing, didefinisikan fungsi **`n-grams`** kalimat akan dipenggal menggunakan **`split`** dengan pemisah spasi sesuai jumlah potongan yang diinginkan. Selanjutnya, dilakukan pembentukan n-gram dengan menggabungkan kata-kata dalam teks menggunakan fungsi **`zip`** dan **`range`**. **`text[i:] for i in range(new_n)`** menghasilkan iterasi yang mengambil kata-kata dari indeks ke-i hingga akhir teks. Kemudian, fungsi **`list`** digunakan untuk mengubah hasil n-gram menjadi daftar.

5.5 Hasil *Engine*

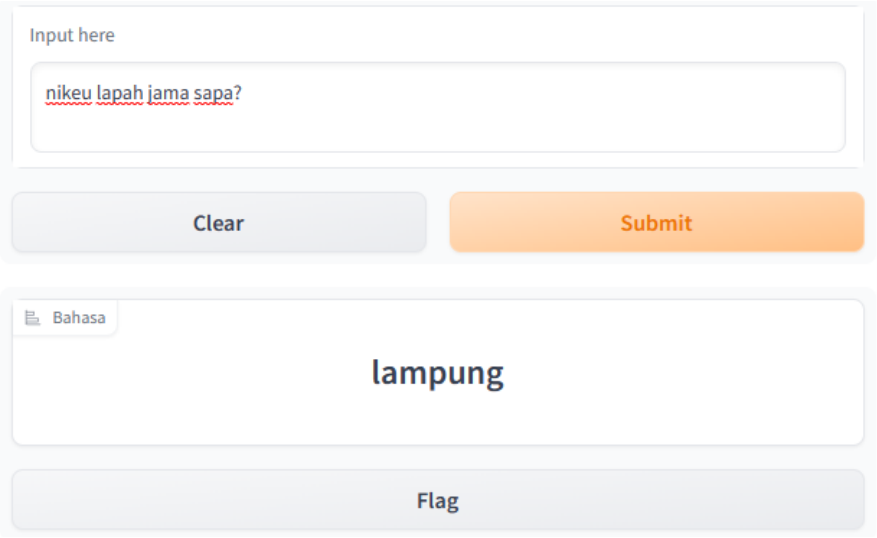
Hasil yang di dapat adalah sebuah *Engine* deteksi bahasa sederhana dan sebuah *Engine* n-gram yang menampilkan hasil dari penelitian yang sedang di teliti. Kemudian untuk testing *Engine* ini, berikut adalah hasil tampilan dari *Engine* deteksi yang sudah selesai.

Hasil yang di dapat adalah sebuah *Engine* deteksi bahasa sederhana dan sebuah *Engine* n-gram yang menampilkan hasil dari penelitian yang sedang di teliti. Kemudian untuk testing *Engine* ini, berikut adalah hasil tampilan dari *Engine* deteksi yang sudah selesai pada gambar 4.13 :

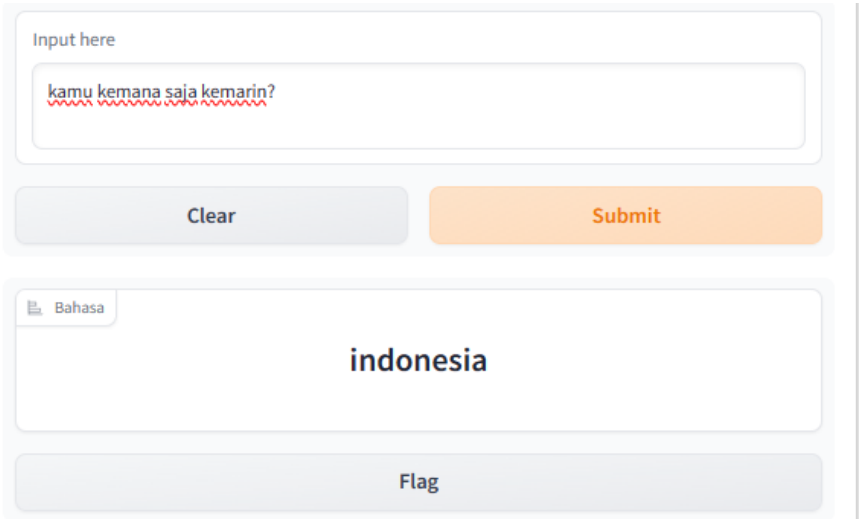


Gambar 4.13 tampilan hasil *engine* pada *Google Collab*

Hasil dari *train data* yang ada kemudian di cek apakah *pelatihan data*-nya berhasil atau tidak, dapat dilihat dari hasil Screenshot diatas bahwa kini model yang ada sudah mampu membedakan dan melabeli mana yang bahasa Indonesia dan mana yang bahasa Lampung.

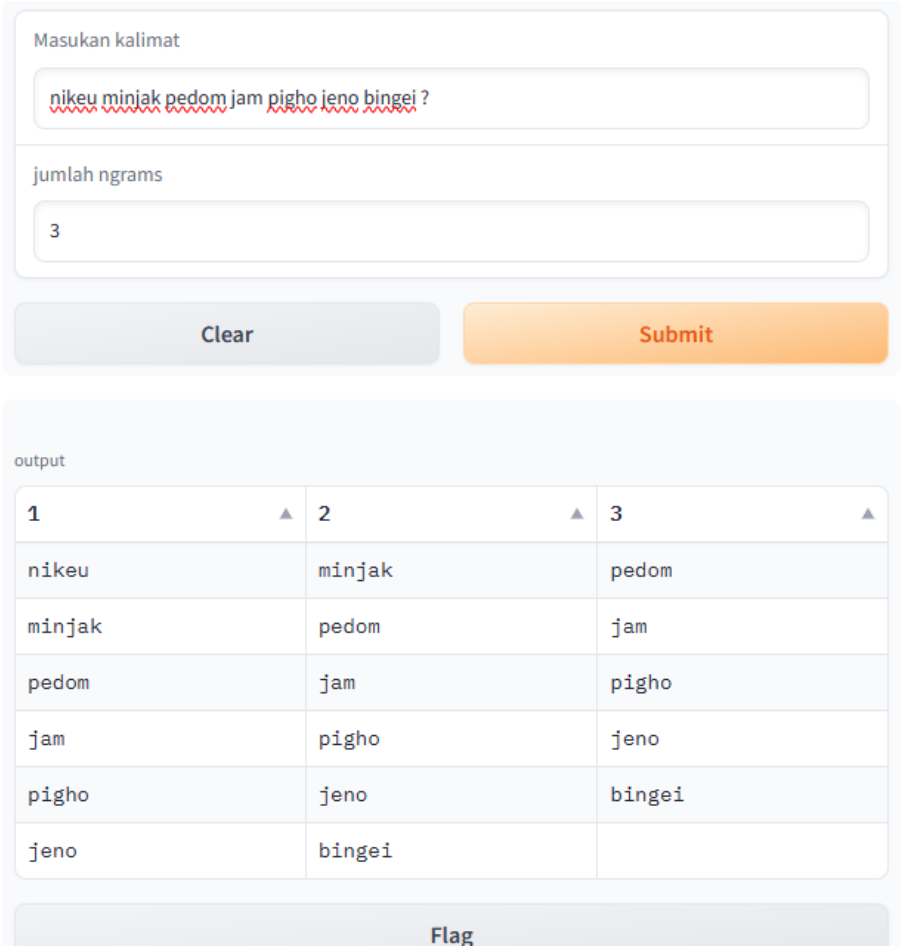


Gambar 4.14 tampilan hasil *engine* saat mendeteksi Bahasa Lampung



Gambar 4.15 tampilan hasil *engine* saat mendeteksi Bahasa Indonesia

Kemudian berikut adalah tampilan mesin *n-gram* yang telah selesai dengan pemenggalan kalimat menjadi beberapa kata :



Gambar 4.16 tampilan hasil *n-gram engine*

BAB V
KESIMPULAN DAN SARAN

Pada Bab ini disampaikan hasil kesimpulan dan saran berdasarkan penelitian yang telah dilakukan dan kemudian digunakan untuk meningkatkan sistem yang ada saat ini.

6.1 Kesimpulan

Berdasarkan hasil dari penelitian ini, algoritma Back-Propagation Neural network dapat melakukan pendeteksian dengan beberapa langkah sebagai berikut:

1. Inisialisasi bobot dan bias secara acak.
2. *Feedforward*: Input diberikan ke model neural network, dan output yang dihasilkan oleh model dihitung dengan mempropagasikan input ke lapisan-lapisan selanjutnya hingga mencapai output layer.
3. Menghitung *error*: *Error* dihitung sebagai selisih antara output yang dihasilkan oleh model dan nilai yang sebenarnya (label) dari input tersebut.
4. *Backpropagation*: *Error* dikembalikan ke lapisan-lapisan sebelumnya dalam model, dan bobot dan bias pada masing-masing lapisan diupdate untuk meminimalkan *error*.
5. Kondisi 2-4 diulangi sebanyak beberapa *iterasi* atau *epoch*, hingga model mencapai nilai yang diinginkan berdasarkan *dataset* yang telah dilabeli.

Untuk menentukan apakah *engine* dapat berfungsi sesuai dengan *dataset* dan model yang ada saat ini dengan menjalankan pengujian menggunakan PEM (*Performance Evaluation Measure*) :

a. Dengan perolehan nilai *persentase* 97% pada aspek *Accuracy* yang berdasarkan hasil *engine detection language*, ini telah memenuhi persyaratan sangat baik untuk kriteria tersebut.
b. Dengan perolehan nilai persentase 98% pada aspek *Precision* yang berdasarkan hasil *engine detection language*, ini telah memenuhi persyaratan sangat baik untuk kriteria tersebut.
c. Dengan perolehan nilai persentase 97% pada aspek *Recall* yang berdasarkan hasil *engine detection language*, ini telah memenuhi persyaratan sangat baik untuk kriteria tersebut.

6.2 Saran

Beberapa saran untuk pengembangan penelitian yang akan datang dalam kaitannya dengan penelitian ini :

1. Penelitian selanjutnya diharapkan dapat menggali potensi algoritma word embedding lainnya seperti *Word2Vec* atau *GloVe* dalam mendapatkan representasi teks yang lebih kaya dan kontekstual daripada metode yang digunakan pada penelitain ini yaitu TF-IDF.
2. Diharapkan dalam penelitian mendatang, dapat dieksplorasi penggunaan algoritma backpropagation dengan pendekatan *recurrent neural network* (RNN) atau *long short-term memory* (LSTM). Algoritma ini dapat mempelajari pola dan struktur kata (SPOK) dalam sebuah kalimat, sehingga membantu meningkatkan kemampuan deteksi bahasa pada kalimat dengan jumlah kata yang sama.
3. Pemanfaatan sumber daya kata atau korpus sudah cukup baik. Sehingga tujuan peneliti dalam pembuatan *engine* deteksi ini sesuai dengan yang

diinginkan oleh peneliti. Oleh karena itu, sangat mungkin untuk dapat menambahkan jumlah korpus ataupun bahasa lain untuk pengembangan di masa mendatang.

DAFTAR PUSTAKA

Abidin, Z., Wijaya, A., & Pasha, D. (2021). Aplikasi Stemming Kata Bahasa Lampung Dialek Api Menggunakan Pendekatan Brute-Force dan Pemograman C. *Jurnal Media Informatika Budidarma*, *5*(1), 1-8.

Aizawa, A. (2003). An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, *39*(1), 45-65.

Ardiyatno, U. (2021). *PENINGKATAN AKURASI MESIN PENERJEMAH BAHASA LAMPUNG BERBASIS KAMUS DENGAN TEKNIK STEMMING DAN PENAMBAHAN KOSA KATA* (Doctoral dissertation, Universitas Teknokrat Indonesia).

Astuti, E. D., 2009, *Pengantar Jaringan Saraf Tiruan*, Star Publishing, Wonosobo.

Berger, A et al (2000). Bridging the Lexical Chasm: Statistical Approaches to Answer Finding. In Proc. Int. Conf. Research and Development in Information Retrieval, 192-199

Budiman, A. A. (2018). Pendeteksi Bahasa Daerah Pada Twitter Dengan Machine Learning.

Cavnar, W. B., & Trenkle, J. M. (2001). N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval* (Vol. 161175).

Goswami, P., & Kamath, V. (2014). The DF-ICF Algorithm-Modified TF-IDF. *International Journal of Computer Applications*, *93*(13).

Evan, F. H. (2014). *Pembangunan Perangkat Lunak Peringkas Dokumen Dari Banyak Sumber Berbasis Web Menggunakan Sentence Scoring Dengan Metode Tf-Idf* (Doctoral dissertation, UAJY).

Haykin, S. (2009). *Neural networks and learning machines, 3/E*. Pearson Education India.

Hendrawati, T., & Yanti, C. P. (2021). Analysis of twitter users sentiment against the covid-19 outbreak using the backpropagation method with adam optimization. *J. Electr. Electron. Informatics*, *5*(1), 1.

Hirschberg, J., & Manning, C. D. (2015). Advances in natural language processing. *Science*, *349*(6245), 261-266

Hussein, O., Sfar, H., Mitrović, J., & Granitzer, M. (2020, December). NLP_Passau at SemEval-2020 Task 12: Multilingual neural network for offensive language detection in English, Danish and Turkish. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation* (pp. 2090-2097).

Jauhiainen, T., Lui, M., Zampieri, M., Baldwin, T., & Lindén, K. (2019). Automatic language identification in texts: A survey. *Journal of Artificial Intelligence Research*, *65*, 675-782.

Kencana, C. W., Setiawan, E. B., & Kurniawan, I. (2020). Hoax Detection System on Twitter using Feed-Forward and Back-Propagation Neural Networks Classification Method. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, *4*(4), 655-663.

Kusrini, E. T. L., & Taufiq, E. (2009). Algoritma data mining. *Yogyakarta: Andi Offset*.

Nasution, M. D. A., Hardinata, J. T., & Damanik, I. S. (2019, September). Jaringan Syaraf Tiruan Backpropagation untuk Klasifikasi Data Tilang Berdasarkan Jenis Pelanggaran. In *Prosiding Seminar Nasional Riset Information Science (SENARIS)* (Vol. 1, pp. 547-556).

O'Sullivan, Conor. (2020, Agustus 25). *Deep Neural Network Language Identification*. Toward Data Science. https://towardsdatascience.com/deep-neural-network-language-identification-ae1c158f6a7d

Prasetyo, H., Adiwijaya, A., & Astuti, W. (2019). Klasifikasi Multi-label Pada Hadis Bukhari Dalam Terjemahan Bahasa Indonesia Menggunakan

Mutual Information Dan Backpropagation Neural Network. *eProceedings of Engineering*, *6*(2).

Sakti, P. C. P. (2022). *DETEKSI BAHASA LAMPUNG PADA MESIN PENERJEMAH MENGGUNAKAN BAHASA PEMROGRAMAN C* (Doctoral dissertation, Universitas Teknokrat Indonesia).

Saraswati, E., Umaidah, Y., & Voutama, A. (2021). Penerapan Algoritma Artificial Neural Network untuk Klasifikasi Opini Publik Terhadap Covid-19. *Generation Journal*, *5*(2), 109-118.

Shukla, A., Tiwari, R., & Kala, R. (2010). *Real life applications of soft computing*. CRC press.

Sijabat, Joseph Frans (2022) *DETEKSI BAHASA LAMPUNG PADA MESIN PENERJEMAH DENGAN BAHASA PEMPROGRAMAN PYTHON*. Strata 1 thesis, Universitas Teknokrat Indonesia.

Wahyu, T. (2020). Digital Transformation : Google Colaboratory sebagai Aplikasi Pengolahan Data Open Source untuk Peneliti dan Pemula. https://dqlab.id/digital-transformation-mengenal-tools-opensournce

Wanto, A. “Penerapan Jaringan Saraf Tiruan Dalam Memprediksi Jumlah Kemiskinan Pada Kabupaten/Kota Di Provinsi Riau,” Kumpulan Jurnal Ilmu Komputer (KLIK), vol. 5, no. 1, pp. 61– 74, 2018.

LAMPIRAN

Lampiran 1. Pelabelan *dataset*

```
import re
import string
import pandas as pd

lampung_df = pd.read_csv('gdrive/My Drive/Scriptsweet/bhn_lpg.csv', sep=';', encoding='uni
                         code_escape')
lampung_df.columns = ["words"]
lampung_df['label'] = "lampung"

indonesia_df = pd.read_csv('gdrive/My Drive/Scriptsweet/
                           bhn_ind.csv', sep=';', encoding='unicode_escape')
indonesia_df.columns = ["words"]
indonesia_df['label'] = "indonesia"

def case_folding(text):
  text = text.strip()
  text = re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z\t])|(\w+:\/\/\S+)"," ",text)
  text = text.translate(str.maketrans("","",string.punctuation))
  text = text.lower()
  return text

dataframe_df = pd.concat([lampung_df, indonesia_df])
dataframe_df["words"] = dataframe_df["words"].apply(lambda x: case_folding(x))
dataframe_df = dataframe_df.sample(frac=1).reset_index
                           (drop=True)
dataframe_df

dataframe_df.to_csv('gdrive/My Drive/Scriptsweet/bhn_gabung.csv', sep=';', encoding='unico
de_escape',index=False)
```

Lampiran 2. Pembuatan model

```
import numpy as np
import pandas as pd
import joblib

from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neural_network import MLPClassifier

mix_df = pd.read_csv('gdrive/My Drive/Scriptsweet/
        bhn_gabung.csv', sep=';', encoding='unicode_escape')
mix_df.head()

#pembuatan model
mix_df['label'].unique()

lb = LabelBinarizer()
X = mix_df['words']
y = lb.fit_transform(mix_df['label']).flatten()

X_train, X_test, y_train, y_test = train_test_split(X ,y, random_state=42, test_size=0.2)

#perhitungan N-Gram & Backprop NN
count_vectorizer = TfidfVectorizer(min_df=5, ngram_range=(1,3)).fit(X_train)
X_train_vectorized = count_vectorizer.transform(X_train)

model = MLPClassifier(hidden_layer_sizes=(200,100,), max_iter=1000).fit(X_train_vectoriz
        ed, y_train)

saved_count_vectorizer = joblib.dump(count_vectorizer, 'gdrive/My Drive/Scriptsweet/count
_vectorizer.joblib')
saved_model = joblib.dump(model, 'gdrive/My Drive/Scriptsweet/model.joblib')
```

Lampiran 3. Pengujian PEM dengan *confusion matrix*

```
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix,
classification_report

word = count_vectorizer.transform(X_test)
predictions=model.predict(word)

print("akurasi : ",accuracy_score(y_test,predictions))
print("presisi : ",precision_score(y_test,predictions))
print("recall  : ",recall_score(y_test,predictions))

print("\nConfusion Matrix \n",confusion_matrix(y_test, predictions))
print("\nClassification Report \n",classification_report(y_test,predictions))
```

Lampiran 4. Pembuatan UI *engine* deteksi

```
import gradio as gr
import joblib
import re
import pandas as pd
import numpy as np
import string
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neural_network import MLPClassifier

# model initiation
model = joblib.load('gdrive/My Drive/Scriptsweet/model.joblib’)
cv = joblib.load('gdrive/My Drive/Scriptsweet/count_vectorizer.joblib')

def preprocess_text(text):
    # remove punctuations
    text = text.translate(str.maketrans("", "", string.punctuation))
    # remove user @ references and hashtags
    text = re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z\t])|(\w+:\/\/\S+)"," ",text)
    # remove useless characters
    text = re.sub(r'[^ -~]', '', text)
     # convert all text to lower case
    text = text.lower()
    return text

def language_cls(text,n=3):
    text = cv.transform([preprocess_text(text)])
    result = 'indonesia' if model.predict(text)[0] == 0 else  'lampung'
    # print(output)
    return result

intfc = gr.Interface(
    fn=language_cls,
    inputs=gr.Textbox(label="Input here", lines=2, placeholder="Input your text"),
    outputs= gr.Label(label='Bahasa')
)

intfc.launch(debug=True)
```

Lampiran 5. Pembuatan UI *engine* n-gram

```
import math
from typing import Text
import gradio as gr
from io import StringIO
import joblib
import re
import pandas as pd
import numpy as np
import string

# model build
def preprocess_text(text):

    # remove punctuations
    text = text.translate(str.maketrans("", "", string.punctuation))
    # remove user @ references and hashtags
    text = re.sub("(@[A-Za-z0-9]+)|([^0-9A-Za-z\t])|(\w+:\/\/\S+)"," ",text)
    # remove useless characters
    text = re.sub(r'[^ -~]', '', text)
     # convert all text to lower case
    text = text.lower()
    return text

def ngrams(text,n):
    text = preprocess_text(text).split(" ")
    new_n=math.floor(n)
    range(new_n)
    # print(output)
    return list(zip(*[text[i:] for i in range(new_n)]))

intfc = gr.Interface(
    fn=ngrams,
    inputs=[gr.inputs.Textbox(label="Masukan kalimat"),
                    gr.inputs.Number(label="jumlah ngrams")],
    outputs= gr.Dataframe()
)

intfc.launch(debug=True)
```