BAB II

LANDASAN TEORI

2.1. Tinjauan Literatur

Dalam rangka mendukung penelitian ini dibutuhkan landasan teori dan tinjauan pustaka dari penelitian terdahulu atau penelitian yang sudah dilakukan sebelumnya yang berkaitan dengan penelitian ini. Adapun penelitian terdahulu yang menjadi tinjauan Pustaka penulis dapat dilihat pada tabel 2.1 berikut ini:

Nomor Literatur	Peneliti	Tahun	Judul
Literatur 1	Javier Gonzales-Dominiquez, Ignacio Lopez-Moreno, Hasim Sak, Joaquin Gonzalez-Rodriguez, Pedro J.Moreno	2014	Automatic language identification using long short-term memory recurrent *neural network*s
Literatur 2	Joseph Chee Chang, Chu-Cheng Lin	2014	Recurrent-Neural-Network for Language Detection on Twitter Code-Switching Corpus
Literatur 3	Amir Hamzah	2010	Deteksi Bahasa Untuk Teks Berbahasa Indonesia
Literatur 4	Pray Christanto	2022	Deteksi Bahasa Lampung Pada Mesin Penerjemah Menggunakan Bahasa Pemrograman C#
Literatur 5	Joseph Frans Sijabat	2022	Deteksi Bahasa Lampung Pada Mesin Penerjemah Menggunakan Bahasa Pemrograman *Python*

Tabel 2. 1 *Tinjauan literatur*

2.1.1. Tinjauan Literatur 1

(Gonzalez-Dominguez et al., 2014) melakukan penelitian dengan menggunakan *RNN* (*Reccurrent Neural network*) dan LSTM (*Long Short Term Memory*) dalam pembuatan sistem pendeteksi bahasa otomatis / *Automatic Language Identification* sebagai pendekatan baru dalam pendeteksian bahasa otomatis, dimana penelitian ini termotivasi dari kesuksesan sistem pendeteksi bahasa otomatis dengan menggunakan DNN (*Deep Neural network*) yang sebelumnya sudah ada. Penelitian ini juga membandingkan hasil dari percobaan dengan sistem *i-vector*. Dalam penelitian ini dibuktikan bahwa dengan menggunakan *RNN* dan LSTM mendapatkan hasil yang cukup baik, dimana, dengan menggunakan *RNN* dan LSTM didapatkan 28% peningkatan performa dibandingkan sistem pendeteksi bahasa otomatis dengan menggunakan metode *i-vector*

2.1.2. Tinjauan Literatur 2

(Chang & Lin, 2014) melakukan penelitian untuk mendeteksi bahasa yang digunakan pada media sosial twitter dimana penggunanya sering melakukan *Code-Switching* atau menggunakan beberapa bahasa dalam satu percakapan, dimana ini akan menyebabkan kesulitan tersendiri dalam pendeteksian bahasa dikarenakan kesulitannya dalam pembuatan sistem. Oleh karena itu literatur ini membuat sistem pendeteksi bahasa untuk *Code-Switching* menggunakan *RNN*(*Reccurent Neural Network*) dan menghasilkan sistem yang lebih efisien dibandingkan sistem sebelumnya yang menggunakan SVM (*Support Vector Machine*) model dimana hasil yang didapatkan dengan menggunakan metode

SVM adalah 90 - 95% akurasi sedangkan hasil dari penggunaan *RNN* adalah 92 - 96% akurasi.

2.1.3. Tinjauan Literatur 3

Penelitian oleh (Hamzah, 2010) bertujuan untuk merancang *sebuah search engine* yang mampu untuk mencari dokumen berbahasa Indonesia, Penelitian tersebut difokuskan pada model deteksi *N-Gram*, yaitu *unigram*, *bigram*, dan *trigram*. Penelitiannya tersebut bertujuan untuk mengkaji metode-metode pendeteksian dokumen berbahas Indonesia, manfaat dari penelitian ini sendiri adalah untuk mencari metode paling efisien untuk mendeteksi sebuah dokumen berbahasa Indonesia atau bukan. Selain itu juga metode yang akan di teliti ini nantinya akan sangat bermanfaat ditengah banyaknya koleksi dokumen yang bertebaran di internet khusunya untuk bidang Pendidikan Sistem Temu Kembali Informasi (STKI) dan dalam perkembangan mesin pencarian khusus (*Special Search Engine*).

2.1.4. Tinjauan Literatur 4

Penelitian oleh (Sakti, 2022), bertujuan untuk mengidentifikasi bahasa lampung yang akan digunakan sebagai tahapan *pre-processing* dalam pembuatan kamus bahasa Lampung dan Indonesia. Penelitian ini menggunakan bahasa pemrograman c# dan juga menggunakan *unity* dengan menggunakan metode *word checker* dan n-gram, penelitian tersebut mendapatkan hasil *training* dimana didapatkan :

- *accuration* sebesar 96%
- *precision* sebesar 98%

- *recall* sebesar 94%

sedangkan untuk hasil akhir dari penelitian tersebut merupakan *sebuah game engine* yang dibuat menggunakan *unity* berfungsi untuk mendeteksi bahasa Lampung dan Indonesia.

2.1.5. Tinjauan Literatur 5

Penelitian oleh (Sijabat, 2022.), bertujuan untuk mengidentifikasi bahasa lampung yang akan digunakan sebagai tahapan *pre-processing* dalam pembuatan kamus bahasa Lampung dan Indonesia. Penelitian ini menggunakan bahasa pemrograman *python* dengan menggunakan metode *N-Gram* dan *Naïve Bayes Classifier*, penelitian tersebut mendapatkan hasil *training* dimana didapatkan :

- *accucary* sebesar 98%
- *precision* sebesar 99%
- *recall* sebesar 96%

2.2. Artificial Intelegence

Artificial Intelligence atau kecerdasan buatan merupakan salah satu cabang ilmu yang berhubungan dengan pemanfaatan mesin untuk memecahkan permasalahan yang rumit dengan cara yang lebih manusiawi (Winiarti & Nugraha, 2014). Mesin dapat bertindak dalam memecahkan suatu masalah dengan menggunakan pendekatan-pendekatan yang manusiawi yang dibekali oleh pengetahuan serta kemampuan penalaran yang baik. Penelitian AI *bias*anya bersangkutan dengan otomatisasi tugas-tugas yang membutuhkan kecerdasan dan proses manusiawi dalam penyelesaiannya, seperti pengenalan objek, pengenalan bahasa, otomatisasi penjadwalan, otomatisasi pemesanan dan banyak lainnya,

selain pada cabang ilmu komputer AI juga berhubungan erat dengan cabang ilmu lainnya terutama Matematika dan Psikologi.

2.3. Machine Learning

Menurut (Ginting & Rohmadi, 2015) *Machine learning* merupakan bagian dari kecerdasan buatan (*Artificial Intelligence*). *Machine learning* adalah salah satu disiplin ilmu dari *computer science* yang mempelajari bagaimana membuat komputer atau mesin agar memiliki suatu kecerdasan. Agar mempunyai suatu kecerdasan, komputer atau mesin harus dapat belajar. Atau dengan kata lain *machine learning* adalah bidang keilmuan yang mempelajari tentang bagaimana pembelajaran komputer atau mesin agar menjadi cerdas. *Machine learning* berhubungan erat dengan *computational statistics* yang berfokus pada suatu prediksi atau pembuatan keputusan berdasarkan penggunaan komputer. Beberapa implementasi dari *machine learning* yaitu seperti *text analysis, image processing, fincance, search and recommendation engine, speech understanding.*

2.4. Deep Learning

Menurut (Zulfa & Winarko, 2017) *Deep Learning* merupakan area baru dalam penelitian *Machine Learning*, yang memanfaatkan jaringan syaraf tiruan untuk implementasi permasalahan dengan *dataset* yang besar. *Deep Learning* adalah tentang belajar beberapa tingkat representasi dan abstraksi yang membantu untuk memahami data seperti gambar, suara, dan teks.

2.5. *Neural network*

Neural network atau *bias*a disebut dengan *Artificial Neural network* adalah bagian dari pendekatan *deep learning*, yang dimana adalah sebuah *framework* yang disatukan oleh metode metode *deep learning*

Neural network terdiri dari *layer-layer*, dimana *layer* ini bisa berjumlah mulai dari tiga hingga ratusan, dimana *neural network* yang hanya terdiri dari beberapa *layer* saja disebut *shallow neural network,*sedangkan *neural network* yang terdiri dari banyak *layer* disebut dengan *deep neural network,* dalam pendekatan deep learning *deep neural network* inilah yang digunakan. Oleh karena inilah *Deep Learning* dapat melakukan tugas-tugas kompleks seperti deteksi objek, *machine translation* dan sebagainya.

Arsitektur *neural network* merupakan elemen-elemen yang memangun sebuah *neural network*. Pada umumnya arstitektur dari *neural network* terdiri dari:

- *Layers*
- *Nodes*
- *Edges/Weights*
- *Biass*
- *Activation functions*

2.5.1. *Layers*

Seperti yang telah dijelaskan bahwa sebuah *neural network* terdiri dari beberapa atau banyak *layer* / Lapisan, meskipun jumlah *layer* ini berbeda pada setiap mode, namun ada 3 jenis *layer* yang digunakan. Setiap *layer* terdiri dari *node-node* yang membentuk sebuah *layer*.

Layer yang digunakan dalam *neural network* adalah:

- *Input layer*

  Layer ini merupakan *layer* yang terdiri dari data *input* yang memasuki *neural network*.

- *Hidden layer*

  Layer ini merupakan *layer* yang melakukan komputasi, *layer* ini terletak di antara *input layer* dan *output layer*, mengambil nilai dari *input layer*, melakukan kalkulasi dan menghasilkan *output* yang dibutuhkan oleh *output layer* .

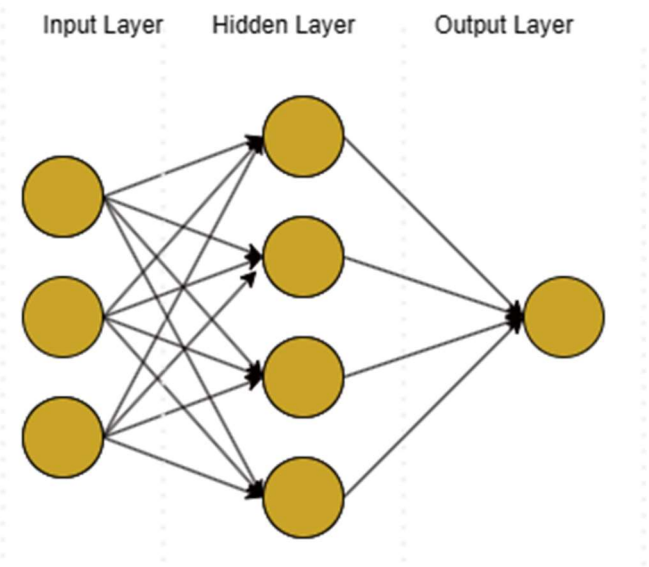
  Layer ini terdiri dari satu atau beberapa *nodes* yang disebut dengan activation *nodes*, setiap *nodes* merupakan fungsi matematis untuk menghasilkan *output*. *Layer* ini dapat berjumlah lebih dari satu, dan pada deep *neural network* dapat berjumlah ratusan *layer* tergantung dengan tugas yang dilakukan.

  Jika terdapat lebih dari satu *hidden layer* maka setiap *output* dari *hidden layer* akan dimasukan kepada *hidden layer* berikutnya sebagai *input*, dan *output* dari setiap *nodes* didalam *hidden layer* akan dikirimkan kepada setiap *nodes* pada *layer* berikutnya.

- *Output layer*

  Layer ini merupakan *layer* yang berada pada posisi paling akhir dalam sebuah *neural network*, terdiri dari *nodes* yang menghasilkan *output* dari semua perhitunhan matematis yang dilakukan pada *layer-layer* sebelumnya.

Struktur dasar dari *neural network* dapat dilihat pada gambar 2.1 berikut:



Gambar 2. 1 *Struktur Neural network*

2.5.2. *Nodes*

Nodes merupakan bagian dari sebuah *layer* dimana pada setiap *node* terdiri dari

- Activation

  Merupakan status dari *node* apakah *node* sudah aktif atau belum

- Threshold Value (opsional)

  Jika ada maka threshold value akan menentukan apakah sebuah neuron sudah aktif atau belum, nilai dari threshold value pada awalnya adalah 0 dan 1.

- *Activation function*

  Pada bagian ini terjadi perhitungan *output* dari sebuah *nodes* berdasarkan *input* yang diberikan, terdapat beberapa jenis *activation function* seperti:

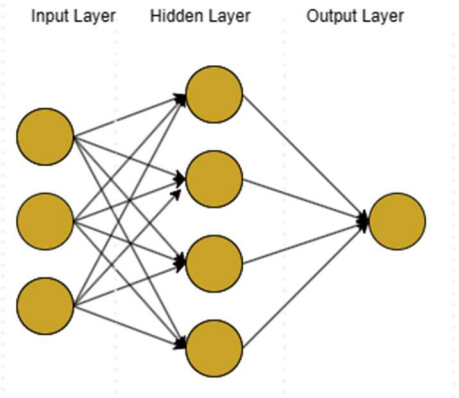
  - *Sigmoid*
  - *ReLu*
  - *Tanh*

- *Output Function*

  Fungsi ini menghasilkan *output* untuk *node* berdasarkan *activation function*.

2.5.3. Edges

Di dalam sebuah *neural network* terdapat panah panah yang merepresentasikan koneksi dari setiap *nodes* dari sebuah *layer*, koneksi ini disebut juga dengan edges. Setiap edge mengarah kepada sebuah *nodes* memiliki bobotnya tersendiri, yang dapat berdampak pada satu *nodes* pada *nodes* yang lain, bobot ini dapat bernilai integer positif ataupun negatif.



Gambar 2. 2 *Struktur neural network*

2.5.4. *Bias*es

Bias merupakan sebuah *node*, dan pada *neural network* setiap *layer* memiliki *bias nodes* kecuali pada *output layer*, *bias nodes* memiliki nilai, yang disebut juga dengan *bias*. Nilai ini dimasukkan kedalam proses menghitung jumlah bobot, juga memiliki peranan dalam menentukan *output* dari sebuah *node*.

Bias merupakan aspek penting dalam neutal network dikarenakan *bias* membuat *activation function* untuk bergeser dan membuat model lebih merepresentasikan data dan membuat *output* yang akurat

2.5.5. *Activation function*s

Merupakan fungsi yang terdapat pada activation *nodes* yang berada pada *hidden layer* dari sebuah *neural network*. Activatio *function* menghitung jumlah bobot dari *input* yang diterimanya, menambahkan *bias*, lalu memasukan hasilnya nya kedalam fungsi aktifasi, *output* ini kemudian akan dikirimkan kepada *layer* atau *nodes* berikutnya. *Output* ini juga dikenal sebagai activation value. Pada *layer* berikutnya akan menerima beberapa activation value dari proses activation *node* sebelumnya dan mengkalkulasikan junlah bobotnya, dan memasukkan nilai jumlah bobot kedalam *activation function* nya, dan menghasilkan *output* dari *layer* tersebut. Inilah jalan data dari sebuah *neural network*. Juga *activation function* membantu merubah *input* yang merupakan data menjadi *output* yang merupakan sebuah prediksi.

Proses di dalam *activation function* jika data terus maju dari *input* ke *output* maka akan disebut dengan feed forward dikarenakan arah jalannya data hanya lurus kedepan. Secara garis besar perhitungan yang terjadi pada sebuah *neural network* adalah sebagai berikut

- *Input layer*

$$x = input\ weights$$

- *Hidden layer*

$$h = Activation\ Function(W_{xh}^{T} x_1 + b_h)$$

Dimana :

W_{x1}^{T} : *Weight* matrix dari *input layer* ke *hidden layer*

b_h : *Bias* term *hidden layer*

- *Output layer*

$$\hat{y} = activation\ function(\ W_{hy}^{T}h + b_y)$$

Dimana :

W_{hy}^{T} : *Weight* matrix dari *hidden layer* ke *output layer*

b_y : *Bias* term *output layer*

2.5.6. *Training* Pada *Neural network*

Proses *training* merupakan proses yang dilakukan untuk mendapatkan nilai dari *weight* dan *bias*, sehingga model *neural network* dapat dengan tepat melakukan prediksi dengan melakukan perhitungan terhadap, *input*, *weight*, dan *bias*.

Weights memiliki peranan yang sangat penting dalam neutal network, dimana saat melakukan perubahan nilai *weight* dari sebuah connection maka *output* dari sebuah *layer* akan berubah, sehingga memiliki nilai *weight* yang optimal sangat penting dalam membuat model yang akurat. Nilai dari *weight* tersebut dapat didapatkan dengan menggunakan dua algoritma yaitu, *Fungsi Loss* dan *Gradient Descent* dimana :

- Fungsi *Loss*

Fungsi *Loss* atau *bias*a dikenal dengan *Cost Function.* menghitung perbedaan antara probabilitas yang terprediksi dari sebuah kategori dan kategori itu sendiri. Sebagai contoh pada kasus klasifikasi dimana model dengan dua kategori, model akan memprediksi apakah sebuah rumah akan terjual atau tidak. Model hanya memiliki dua *output* yaitu, ya dan tidak, nilai 1 jika ya dan 0 jika tidak. Jika probabilitas dari *output* menghasilkan nilai mendekati 1 maka akan menjadi kategori ya, fungsi *Loss* akan mengukur perbedaan ini, sedangkan pada kasus regresi maka fungsi *Loss* akan mengukur error antara nilai asli dan nilai yang terprediksi. Sehingga fungsi *Loss* membantu model untuk meng evaluasi performanya. Jika nilai dari *Loss* sampai pada nilai minimum, maka nilai nilai dari koefisien model yang terakhir akan digunakan, sehingga jika *input* dimasukan maka perhitungan akan dilakukan dengan koefisien model yang optimal.

Ada banyak fungsi *Loss* yang dapat digunakan seperti mean squared error (Untuk Kasus Regresi) dan Log *Loss* (Untuk Kasus Klasifikasi) dengan rumus MSE (Mean Squared Error):

$$MSE = \frac{1}{n}\sum(y_i - f(x_i))^2$$

Dimana :

n = Total data point

yi = nilai asli

xi = *input*

f() = fungsi dari *input* untuk menghasilkan *output*

f(xi) = nilai terprediksi

sedangkan pada Log *Loss* secara matematis akan dirumuskan :

$$Log\ Loss = -\frac{1}{n}\sum y_i \left(\log\left(p(y_i)\right) + (1 - y_i)\right)\left(\log\left(1 - p(y_i)\right)\right)$$

Dimana :

N = Total Data Point

Yi = Label Asli

P = Probabilitas Terprediksi

- Gradient Descent Algorithm

Gradient merupakan nilai kemiringan atau kecondongan suatu garis yang membandingkan antara komponen y (ordinat) dengan komponen x (absis). Gradient disini merupakan nilai kemiringan suatu fungsi, yaitu tingkat perubahan parameter terhadap jumlah parameter lain. Secara matematis, gradient dapat digambarkan sebagai turunan parsial dari serangkaian parameter terhadap *input*nya. Semakin banyak gradient maka semakin curam lerengnya.

Gradient Descent juga digambarkan sebagai iterasi yang digunakan untuk menemukan nilai optimal dari parameter dengan menggunakan kalkulus untuk menemukan nilai minimum. Gradient Descent digunakan untuk pembaruan bobot dalam *Neural network* dengan cara meminimalkan *Loss function*. Dalam *Neural network*, algoritma gradient descent dan fungsi *Loss* bekerja sama untuk menemukan nilai yang akan diberikan ke connection sebagai *weight* dan *bias*. Nilai-nilai ini diperbarui dengan meminimalkan fungsi *Loss* menggunakan algoritma Gradient Descent, sebagaimana adanya kasus dalam model regresi linier. Selain itu, dengan kasus regresi linier, selalu hanya ada satu minimum, karena fungsi kerugian berbentuk mangkuk. Ini memudahkan algoritma Gradient Descent untuk menemukannya dan

memastikan titik terendah. Namun, dalam kasus jaringan saraf, tidak sesederhana itu Fungsi aktivasi yang digunakan oleh jaringan saraf berfungsi untuk memperkenalkan non-linearity terhadap situasi.

- Backpropagation

  Backpropagation dilakukan dengan menggunakan turunan parsial dari fungsi *Loss* . melibatkan penghitungan *Loss* setiap *node* di setiap *layer* dengan bergerak mundur di *Neural network*. Mengetahui *Loss* setiap *node* memungkinkan jaringan untuk memahami *weight* mana yang memiliki dampak negatif drastis pada *output* dan *Loss* . Dengan demikian, Gradient Descent mampu mengurangi *weight* dari connection yang memiliki tingkat kesalahan tinggi, sehingga mengurangi dampak yang ditimbulkan *node* tersebut pada *output*.

  Secara matematis maka :

$$f(x) = X(Y(Z(x)))$$

  Dimana :

  X,Y,Z = *Activation function*

2.6. *Reccurent Neural Network*

Reccurent Neural Network adalah salah satu jenis dari *Neural network* (Jaringan Neural) yang pada umumnya digunakan untuk mendeteksi pola pada data berurut (Schmidt, 2019). *Reccurent Neural Network* dibuat dengan tujuan agar *Neural network* memiliki memori dan dapat menyimpan data yang berguna dan menggunakan informasi tersebut untuk membantu menganalisa *input* saat ini (Bokka et al., 2021).

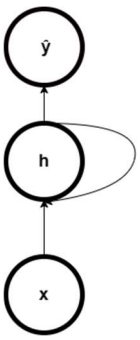
Reccurent Neural Network pada saat ini secara luas digunakan dalam beberapa aplikasi seperti:

- Speech Recognition : Seperti Alexa, Siri dan Google Voice Assistant menggunakan *Reccurent Neural Network* pada sistem Scpeech Recognition yang digunakan.

- Time Series Prediction : *Reccurent Neural Network* juga digunakan luas pada pengolahan dan prediksi data yang berbasiskan waktu, seperti prediksi cuaca, prediksi pasar saham, dan trafik website.

- Natural Language Processing : Aplikasi seperti Machine Translation, Chatbot, dan mesin penjawab pertanyaan menggunakan *Reccurent Neural Network* sebagai model nya.

Cara kerja *Reccurent Neural Network* sesuai dengan penamaan nya yaitu Reccurent atau berulang, pengulangan yang dilakukan adalah pengulangan pada setiap *input* dalam satu urutan *input*, jadi pada setiap *input* maka hasil dari perhitungan *hidden layer* akan disimpan dan pada saat waktu *input* yang berikutnya akan dimasukkan kembali sebagai penjumblah *input*, sehingga pengulangan terjadi pada setiap proses *input*. Arsitektur dari *RNN* dapat dijelaskan pada gambar berikut:

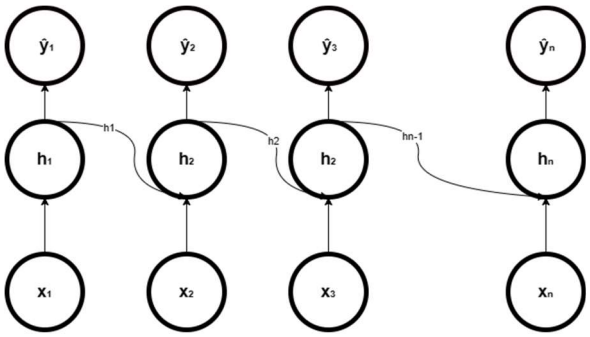


Gambar 2. 3 *Arsitektur RNN*

Dimana :

- x : *Input Layer*
- h : *Hidden Layer*
- ŷ : *Output Layer*

atau jika diperjelas maka akan menjadi seperti berikut :



Gambar 2. 4 *Arsitektur RNN diperjelas*

Dimana :

- x : *Input Layer*
- h : *Hidden Layer*
- ŷ : *Output Layer*

Seperti pada *neural network* pada umumnya pada untuk setiap connection akan dilakukan perhitungan terhadap *weight* dan *bias* dan pada *hidden layer* akan dilakukan perhitungan terhadap *activation function*, namun yang membedakan *RNN* dengan *neural network* sebelumnya adalah *RNN* mengambil *weight* matrix dari hasil perhitungan *hidden layer* dan menambahkan *weight* matrix tersebut dengan *weight* matrix dari *input*, jika dirumuskan perhitungan pada *hidden layer* maka akan didapat :

$$h = Activation\ Function(W_{xh}^{T} x_1 + W_{hh}^{T} h_{t-1} + b_h)$$

Dimana :

W_{x1}^{T} : *Weight* matrix dari *input layer* ke *hidden layer*

$W_{hh}^{T} h_{t-1}$: *Weight* matrix dari *hidden layer* sebelumnya

b_h : *Bias term hidden layer*

sedangkan pada perhitungan *output* akan sama dengan *neural network* yang dijelaskan pada subbab sebelumnya yaitu :

$$\hat{y} = activation\ function(\ W_{hy}^{T} h + b_y)$$

Dimana :

W_{hy}^{T} : *Weight* matrix dari *hidden layer* ke *output layer*

b_y : *Bias term output layer*

Dari notasi matematis di atas dapat disimpulkan bahwa setiap hasil dari *hidden layer* sebelumnya akan dimasukkan dan dijumlahkan ke dalam *hidden layer* saat ini, hasil dari *hidden layer* sebelumnya tersebut disebut juga dengan *state* dimana setiap perhitungan *hidden layer* maka *state* dari *input* sebelumnya mempengaruhi hasil dari perhitungan *output* saat ini, oleh karena pengulangan inilah mengapa *neural network* ini disebut juga dengan Recurrent *Neural network*.

2.6.1 Backpropagation Through Time (BPTT)

Backpropagation Through Time umumnya sama seperti backpropagation *bias*a pada *neural network*. Satu-satunya perbedaan utamanya adalah pada *RNN* backpropagation harus dilakukan pada setiap langkah waktu yang ada sepanjang jalur *RNN* yang berulang. Backpropagation mengacu pada dua hal, yang pertama merupakan sebuah metode matematika yang digunakan untuk menghitung turunan

dan penerapan chain rule. Yang kedua merupakan sebuah algoritma pelatihan untuk memperbarui bobot jaringan untuk meminimalkan kesalahan. Cara menghitung BPTT adalah dengan mencari nilai gradien mulai dari nilai error atau nilai *Loss* hingga ke parameter (bobot dan *bias*) yang ingin diubah

2.7. Natural Languang Processing

Pemrosesan bahasa alami (Natual Language Processing - NLP) merupakan salah satu bidang ilmu Kecerdasan Buatan (*Artificial Intelligence*) yang mempelajari komunikasi antara manusia dengan komputer melalui bahasa alami, baik lisan maupun tulisan, Bahasa sendiri dapat dibedakan ke dalam 2 kategori yaitu bahasa alami dan bahasa buatan, bahasa alami sendiri adalah bahasa yang sering di gunakan oleh manusia untuk berkomunikasi sehari-hari, sedangkan bahasa buatan adalah Bahasa khusus yang dibuat untuk kebutuhan tertentu (Arman, 2004).

Chomsky adalah orang yang pertama kali merepresentasikan bahasa sebagai rangkaian simbol. Chomsky berhasil memperlihatkan bahwa bahasa apapun dapat direpresentasikan dengan suatu cara yang universal. Pemikiran Chomsky yang merepresentasikan bahasa sebagai kumpulan simbol-simbol dan aturan yang mengatur susunan simbol-simbol tersebut telah membuka peluang untuk melakukan pemrosesan bahasa secara simbolik dengan teknologi komputer, sehingga melahirkan bidang ilmu Natural Language Processing (NLP) (Arman, 2004).

Pengolahan Bahasa Alami terdiri dari tiga bagian utama, yaitu:

- Parser adalah sebuah program yang menguraikan kalimat masukan bahasa alami menjadi bagian tata bahasa (kata benda, kata kerja, kata sifat, dll.).

- Sistem Representasi Pengetahuan Sebuah sistem yang menganalisis *output* dari sebuah parser untuk menentukan artinya.
- *Output* Translator merupakan terjemahan dari sistem pengetahuan dan melakukan langkah-langkah tertentu, dalam bentuk jawaban dalam bahasa alami atau keluaran khusus yang kompatibel dengan program komputer lain.

NLP sendiri memiliki beberapa kategori antara lain sebagai berikut:

- Natural Language Tranlator

  Merupakan sebuah translator/alat penerjemah dari satu bahasa alami ke bahasa alami lainnya, misalnya translator bahasa Inggris ke bahasa Indonesia, bahasa Indonesia ke bahasa Jepang dan sebagainya. Translator bahasa alami bukan hanya kamus yang menerjemahkan kata per kata, tetapi harus juga mentranslasikan sintaks dari bahasa asal ke bahasa tujuannya.

- Translator Bahasa Alami ke Bahasa Buatan

  Translator yang mengubah perintah-perintah dalam bahasa alami menjadi bahasa buatan yang dapat dieksekusi oleh mesin atau komputer. Sebagai contoh, translator yang memungkinkan kita memberikan perintah bahasa alami kepada komputer. Dengan sistem seperti ini, pengguna sistem dapat memberikan perintah dengan bahasa sehari-hari, misalnya, untuk menghapus semua file, pengguna cukup memberikan perintah “komputer, tolong hapus semua file!” Translator akan mentranslasikan perintah bahasa alami tersebut menjadi perintah bahasa formal yang dipahami oleh komputer

- Text Summarization

  Suatu sistem yang dapat meringkas hal-hal yang penting dari teks yang diberikan. Dalam dunia kecerdasan buatan pengolahan bahasa alami merupakan aplikasi terbesar setelah sistem pakar. Banyak para ahli *Artificial Intelligence* berpendapat bahwa bidang yang penting yang dapat dipecahkan oleh *Artificial Intelligence* adalah Pengolahan Bahasa Alami (Natural Language Processing).

2.8. Word Embedding

Pada subbab sebelumnya telah dijelaskan bahwa natural language processing merupakan sebuah bidang ilmu yang mempelajari tentang komunikasi antara manusia dan komputer dalam bentuk kata suara dan lain lain, namun dalam penerapannya pada *neural network* struktur dari *neural network* tidak memungkinkan sebuah kata / kalimat untuk menjadi *input*, dimana didalam perhitungan pada *neural network* dibutukan data dalam berntuk numerik, oleh karena itu dibutukan cara agar kata kata dapat menjadi bentuk numerik yang merepresentasikan data tersebut.

Dikarenakan masalah tersebut dibuat solusi yang disebut dengan one hot vector dimana setiap kata dirubah kedalam bentuk numerik dengan cara membuat vector berdasarkan posisi kata tersebut. Namun dengan menggunakan car aini setiap kata dan kata lainnya tidak memiliki hubungan sama sekali, dan juga tidak memiliki konteks. Maka dibuatlah cara yang disebut word embedding dimana vector setiap kata memiliki nilai kemiripan dengan vector kata lainnya.

2.9. Text Classification

Klasifikasi teks juga dikenal sebagai penandaan teks atau kategorisasi teks adalah proses mengkategorikan teks ke dalam kelompo-kelompok berdasarkan kelas nya. Dengan menggunakan Natural Language Processing (NLP), pengklasifikasi teks dapat secara otomatis menganalisis teks dan kemudian menetapkan sekumpulan kelas dan kategori yang telah ditentukan sebelumnya berdasarkan kontennya. Teks tidak terstruktur ada di mana-mana, seperti email, percakapan obrolan, situs web, dan media sosial, tetapi sulit untuk mengekstrak nilai dari data ini kecuali jika diatur dengan cara tertentu. Melakukannya dulunya merupakan proses yang sulit dan mahal karena memerlukan waktu dan sumber daya untuk menyortir data secara manual sulit dipertahankan. Pengklasifikasi teks dengan NLP telah terbukti menjadi alternatif yang bagus untuk menyusun data tekstual dengan cara yang cepat, hemat biaya, dan dapat disekalakan dengan cukup baik. Klasifikasi teks menjadi bagian bisnis yang semakin penting karena memungkinkan untuk dengan mudah memperoleh wawasan dari data dan mengotomatiskan proses bisnis.

Beberapa contoh dan kasus penggunaan paling umum untuk klasifikasi teks otomatis meliputi:

1. Analisis Sentimen: proses pemahaman jika teks tertentu bermakna secara positif atau negatif tentang subjek tertentu (contoh : penilaian suatu produk berdasarkan hasil survei).

2. Deteksi Topik: tugas untuk mengidentifikasi tema atau topik dari sebuah teks (contoh : mengetahui konteks dari sebuah teks yang di ambil dari internet).

3. Deteksi Bahasa: prosedur mendeteksi bahasa dari teks tertentu (contoh: pendeteksian bahasa yang digunakan pada Google Translate sebelum dilakukan penerjemahan).

2.10. Tensorflow

Tensorflow merupakan platform machine learning yang berbasiskan *python*, dan dibangun utamanya oleh Google. Tujuan utama dibuatnya tensorflow adalah sebagai alat agar peneliti dapat memanipulasi ekspresi matematis kedalam bentuk angka atau *tensor* (Chollet, 2021). Sebagai sebuah platform tensorflow mempunyai komponen ekosistem yang luas, dimana Sebagian dibangun oleh Google dan Sebagian dibangun oleh pihak ketiga, seperti :

- TF-agents digunakan untuk aplikasi reinforcement learning

- TFX digunakan untuk workflow management machine learning pada bidang industri

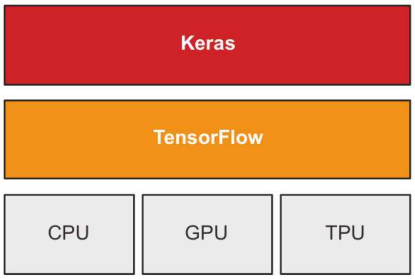
- Tensorflow Hub Repository penyimpanan model-model tensorflow yang sudah di *train* sebelumnya

Komponen-komponen ini mencakup penggunaan yang sangat luas mulai dari penelitian hingga aplikasi pada produksi skala besar. Tensorflow juga dapat dijalankan tidak hanya pada CPU, namun dapat berjalan pada GPU ataupun TPU.

2.11. Keras

Keras merupakan sebuah API (Application Programming Interface) untuk *python* yang dibuat atas TensorFlow, yang menyediakan cara mudah untuk mendefinisikan dan melatih model deep learning. Dikarenakan dibuat atas tensorflow, maka keras juga dapat berjalan menggunakan perangkat keras yang

beragam termasuk GPU dan TPU, juga dapat digunakan pada beragam jenis mesin seperti gambar 2.1 berikut:



Gambar 2. 5 *Urutan Keras pada TensorFlow*

2.12. *Python*

Python merupakan salah satu contoh bahasa pemrograman tingkat tinggi. Dimana *Python* didesain dengan sangat mudah agar dapat mudah di baca dan dipahami, karena sama seperti bahasa pemograman yang lainnya yaitu dengan menggunakan kata bahasa inggris. Bahasa tingkat tinggi juga mudah diubah portable untuk disesuaikan dengan mesin yang menjalankannya. Hal ini berbeda dengan bahasa mesin yang hanya dapat digunakan untuk mesin tersebut. Dengan berbagai kelebihan ini, maka banyak aplikasi ditulis menggunakan bahasa tingkat tinggi.

Python dibuat dan dikembangkan lebih lanjut oleh Guido Van Rossum, yaitu seorang programmer yang berasal dari Negara Belanda. Dibuat dan dikembangkan di kota Amsterdam, Belanda pada tahun 1990. Pada tahun 1995 *Python* dikembangkan lagi agar lebih kompatibel oleh Guido Van Rossum. Kemudian di awal tahun 2000 versi *Python* dikembangkan dan diperbaharui lagi sehingga kini Bahasa Pemrograman *Python* telah mencapai Versi 3. *Python* bisa dijalankan dan ditulis untuk membangun aplikasi di beragam sistem operasi seperti :

1. Linux/Unix

2. Microsoft Windows

3. Mac OS

4. Android

5. Java Virtual Machine

6. Symbian OS

7. Amiga

8. Palm

9. OS/2

2.13. Performance Evaluation Measure

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 satu bundel tahapan yang digunakan untuk mengukur performa suatu sistem. PEM dalam banyak kasus digunakan dalam *training* data, tujuannya untuk mengevaluasi model yang sudah dibuat. Ada banyak perhitungan untuk mendapatkan nilai PEM, *bias*anya diterapkan sebagai kombinasi atau juga secara parsial. Beberapa perhitungan dalam PEM antara lain :

• Precision.

Precision adalah tingkat ketepatan antara request pengguna dengan jawaban sistem;

• Accuration.

Accuration adalah penbandingan antara informasi yang dijawab oleh sistem dengan benar dengan keseluruhan informasi; dan

• Recall.

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

2.14. Metode Pengembangan Sistem

Metode Pengembangan Sistem adalah adalah metode-metode, prosedur-prosedur, konsep-konsep pekerjaan, aturan-aturan yang akan digunakan sebagai pedoman bagaimana dan apa yang harus dikerjakan selama pengembangan ini. Metode adalah suatu cara/teknik sistematis untuk mengerjakan sesuatu. Urut-urutan prosedur untuk penyelesaian masalah ini dikenal dengan istilah algoritma.