BAB II
TINJAUAN PUSTAKA

Penelitian ini dibantu dengan tinjauan pustaka yang digunakan sebagai referensi dan landasan teori penelitian terhadap masalah yang diangkat. Referensi didapatkan dengan membaca jurnal internasional dan nasional, artikel pada halaman web, dan buku-buku dengan rentang lima tahun sebelum penelitian ini dilakukan yang berkaitan dengan penelitian ini.

2.1 Pencapaian Terdahulu

Penelitian kali ini menggunakan referensi penelitian terdahulu yang digunakan sebagai perbandingan, acuan dan studi literatur. Penelitian mengenai analisis sentimen Sinovac pada Twitter terhadap masyarakat Indonesia dengan menggunakan algoritma Long Short Term Memory dan tujuh kelas sentimen data belum pernah dilakukan sebelumnya, maka dari itu referensi topik yang digunakan mengenai penelitian sentimen analisis vaksinasi dengan algoritma Long Short Term Memory dan algoritma *machine learning* yang relevan. Referensi penelitian terdahulu dijabarkan pada **Tabel 2.1**.

Tabel 2.1 Hasil Penelitian Terdahulu

No	Nama Penulis dan Tahun	Judul	Hasil
1	Baita, A., Pristyanto, Y., & Cahyono, N (2021)	Analisis Sentimen Mengenai Vaksin Sinovac Menggunakan Algoritma Support Vector Machine (SVM) dan K-Nearest Neighbor (KNN)	Penelitian ini menghasilkan 1030 sentimen positif, 388 sentimen negatif dan 687 sentimen netral. Serta menghasilkan nilai akurasi 70% untuk algoritma SVM dan 56% untuk algoritma KNN
2	Lestari, S., & Saepudin, S (2021)	Analisis Sentimen Vaksin Sinovac Pada Twitter Menggunakan Algoritma Naïve Bayes	Penelitian ini menghasilkan 86% sentimen positif, 14% sentimen negatif. Serta menghasilkan nilai akurasi 92,96%

3	Alam, N A., Khan ,S Md., Dhruba A R., Khan, M M., Al-Amri, J F., Masud, Mehedi., Rawashdeh, M (2021)	*Deep Learning-Based Sentimen Analysis of Covid-19 Vaccination Responses from Twitter Data*	Penelitian ini menghasilkan 33,96% sentimen positif, 17,55% sentimen negatif dan 48,49% sentimen netral. Serta menghasilkan nilai akurasi LSTM sebesar 90,95%
4	Azizah, L M., Ajipratama, D B., Putri, N A., Damarjati, C (2022)	Analisis Sentimen Masyarakat terhadap Kebijakan Vaksinasi Covid-19 di Indonesia pada Twitter Menggunakan Algoritma LSTM	Penelitian ini menghasilkan 79% sentimen positif, 8% sentimen netral dan 8% sentimen negatif. Serta menghasilkan nilai akurasi sebesar 71%
5	Romli., Kamarula, F., Rochmawati (2022)	Perbandingan CNN dan Bi-Lstm pada Analisis Sentimen dan Emosi Masyarakat Indonesia di Media Sosial Twitter Selama Pandemik Covid-19 yang Menggunakan Metode *Word2vec*	Penelitian ini menghasilkan 61% sentimen positif, 82% sentimen negatif dan 36% sentimen netral. Serta menghasilkan nilai akurasi 84,36% untuk algoritma Bi-LSTM dan 68,58% untuk algoritma CNN
6	CNN (2020)	Epidemiolog Soal Masyarakat Ragu Vaksin: Banyak yang Bingung	Hasil survei menyebutkan 69% kontra terhadap dan 31% pro terhadap vaksin Sinovac

2.2 Tinjauan Teoritis

Teori merupakan alur penalaran dari seperangkat konsep, definisi dan proposisi yang disusun secara sistematis. Dengan demikian tinjauan teori digunakan untuk mendefinisikan dan menjadi pedoman untuk melakukan penelitian.

2.2.1 Sinovac

Sinovac adalah vaksin yang digunakan untuk mengobati infeksi SARS-CoV-2 dan COVID-19. Sinovac juga dikenal dengan nama CoronaVac, dan telah mendapatkan persetujuan penggunaan darurat dari Badan Pengawas Obat dan

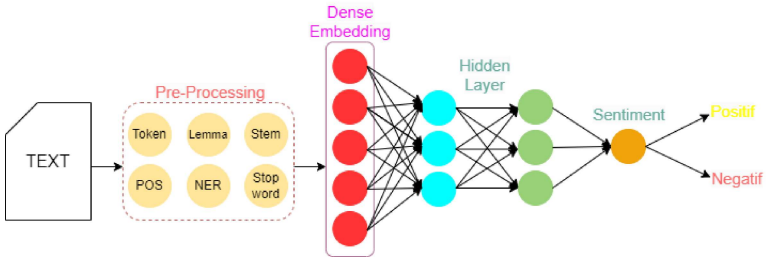
Makanan (BPOM) RI. CoronaVac adalah vaksin yang mengandung virus SARS-CoV-2 yang sudah tidak aktif. Vaksin Sinovac akan membantu sistem kekebalan tubuh untuk mengidentifikasi virus yang sudah tidak aktif dan memproduksi *antibody* untuk melawannya, sehingga dapat mencegah infeksi COVID-19. Vaksin Sinovac dibuat dan dikembangkan oleh Sinovac *Biotech* Ltd. Vaksin ini telah dilakukan uji klinis di Brazil, Turki dan Indonesia (Lestari & Saepudin, 2021).

2.2.2 Artificial Intelligence

Artificial Intelligence adalah kepintaran yang ditambahkan kedalam suatu sistem yang dapat disusun pada konteks ilmiah. Andreas Kaplan dan Michael Haenlein mendeskripsikan *Artificial Intelligence* sebagai kemampuan sistem untuk menerangkan data eksternal dengan benar, untuk belajar dari data tersebut, dan menggunakan pembelajaran tersebut guna mencapai tujuan dan tugas tertentu melalui adaptasi yang fleksibel (Siahaan et al., 2020).

2.2.3 Deep Learning

Deep learning merupakan metode pembelajaran yang menggunakan *Artificial Neural Network*. *Artificial Neural Network* dibuat semirip mungkin dengan otak manusia, dimana seluruh *neuron* terkoneksi satu dengan yang lain sehingga membentuk jaringan *neuron* yang sangat rumit. *Deep learning* juga memanfaatkan *multiple non-linear transformation*. *Deep learning* menggunakan beberapa *layers* yaitu *input layers* dan *output layers*. Hal tersebut digunakan untuk memproses non-linear dengan beberapa tahapan, yang hasilnya digunakan pada klasifikasi pola dan *feature learning*. *Deep learning* dapat dideskripsikan sebagai sebuah kelas dalam algoritma *machine learning* yang menggunakan beberapa lapisan pemrosesan non-linear yang disusun secara *cascade* untuk *feature extraction* dan *transformation* (Dang, Garcia & Prieta., 2020). Klasifikasi *deep learning* dapat dilihat pada **Gambar 2.1** berikut.



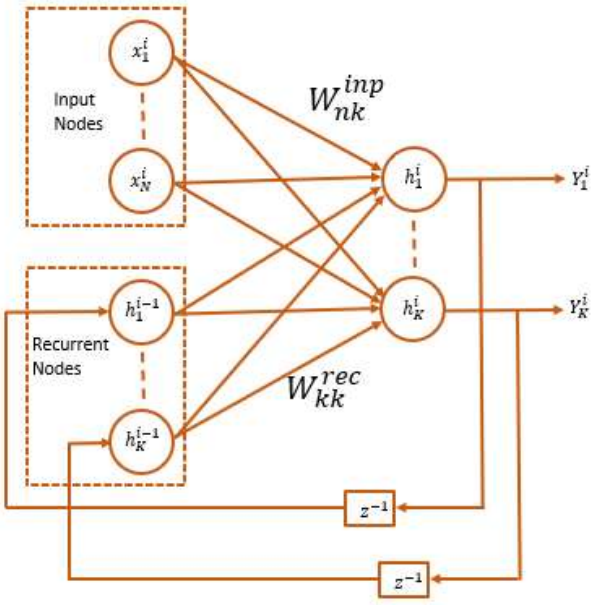
Gambar 2.1 Klasifikasi *Deep Learning* (Dang, Garcia & Prieta., 2020)

2.2.4 Analisis Sentimen

Analisis sentimen adalah suatu kegiatan yang menganalisis perasaan, emosi dan pendapat seseorang yang diungkapkan secara tertulis. Analisis sentimen juga merupakan ilmu yang digunakan untuk menganalisis pendapat seseorang, emosi seseorang, evaluasi seseorang, sikap seseorang dan ke dalam bahasa tertulis. Pada umumnya analisis sentimen melakukan suatu proses pengelompokkan teks yang ada di dalam dokumen, kalimat atau suatu pendapat sehingga menghasilkan suatu nilai yang mengartikan apakah dokumen, kalimat dan pendapat tersebut bernilai positif, negatif atau netral (Aldean, Paradise & Nugraha., 2022).

2.2.5 Long Short Term Memory

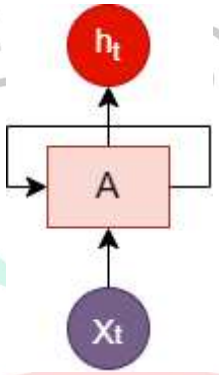
Long Short Term Memory merupakan salah satu tipe dari *Neural Network,* yang termasuk ke dalam bagian dari *Recurrent Neural Network* (RNN). RNN menggunakan *n* umpan balik dari *output layers* dan kembali ke *input layers*. Setiap koneksi umpan balik tersebut dapat digunakan sebagai *time-delay gate* yang tertera pada **Gambar 2.2**.



Gambar 2.2 *Time-delay Gate* (Kamal Wisyaldin et al., 2020)

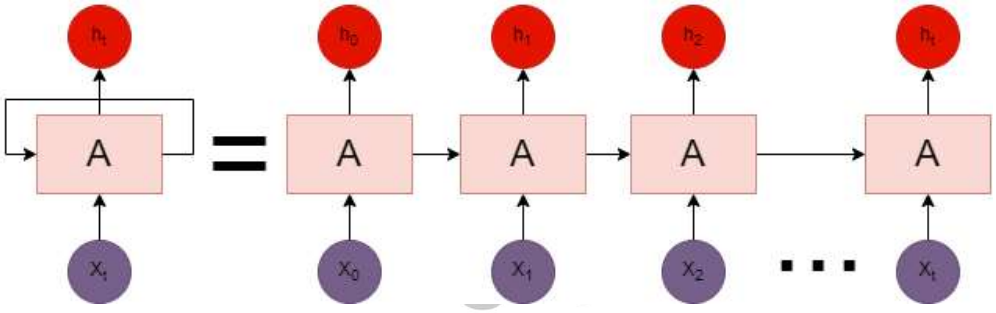
Arsitektur RNN mewakili secara eksplisit pengaruh nilai *output* masa lalu pada perhitungan *output* saat ini, menjadikannya ideal untuk memodelkan struktur autokorelasi dari data *time series* (Kamal Wisyaldin et al., 2020). Metode yang

digunakan oleh RNN untuk mendapatkan informasi masa lalu adalah dengan melakukan perulangan dalam arsitektur, yang secara otomatis mengembalikan informasi dari masa lalu. Pada **Gambar 2.3** adalah gambaran *output* dari RNN A. RNN yang dimaksud menerima input X_t dan menghasilkan *output* h_t. Serta perulangan tersebut memungkinkan informasi ditransfer dari satu tahap ke tahap berikutnya.



Gambar 2.3 Perulangan RNN

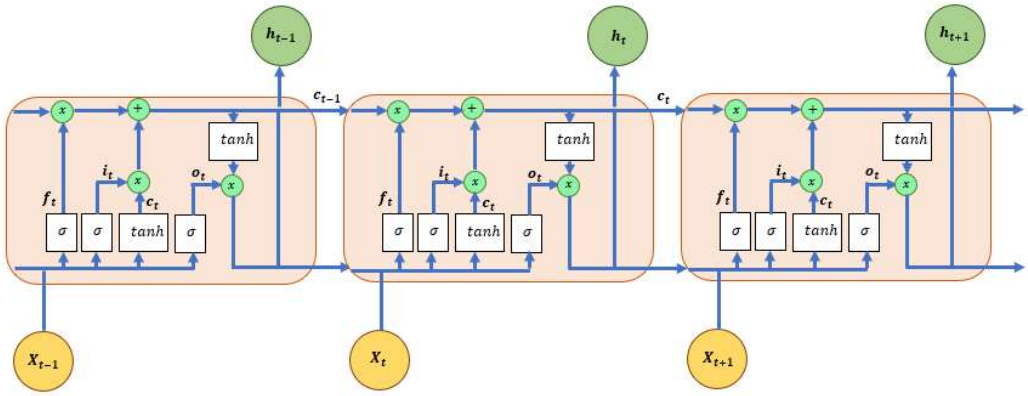
RNN pada **Gambar 2.3** yang dimaksud menerima input X_t dan menghasilkan output h_t. Serta perulangan memungkinkan informasi ditransfer dari satu tahap ke tahap berikutnya. Untuk mempermudah pembelajaran, pada **Gambar 2.4** merupakan contoh perulangan RNN bila dibuka, bisa digambarkan seperti banyak *Neural Network* yang berhubungan satu sama lain.



Gambar 2.4 Perulangan RNN yang Dibuka

Long Short Term Memory dibangun dan dirancang untuk mengatasi masalah gradien menghilang dari RNN ketika berhadapan dengan *vanishing* dan *exploding gradient* tersebut. Arsitektur LSTM terdiri dari lapisan *input*, lapisan *output*, dan lapisan tersembunyi yang disajikan pada **Gambar 2.5**. Lapisan

tersembunyi terdiri dari sel memori. Satu sel memori memiliki tiga *gate* yaitu *input gate, forget gate* dan *output gate* (Wiranda & Sadikin., 2019).



Gambar 2.5 Arsitektur Long Short Term Memory (Wiranda & Sadikin., 2019)

Input gate (i_t) berperan mengambil *output* sebelumnya dan *input* baru serta melewatkan mereka melalui lapisan *sigmoid*. *Gate* ini mengembalikan nilai 0 atau 1. Rumus dari (i_t) adalah :

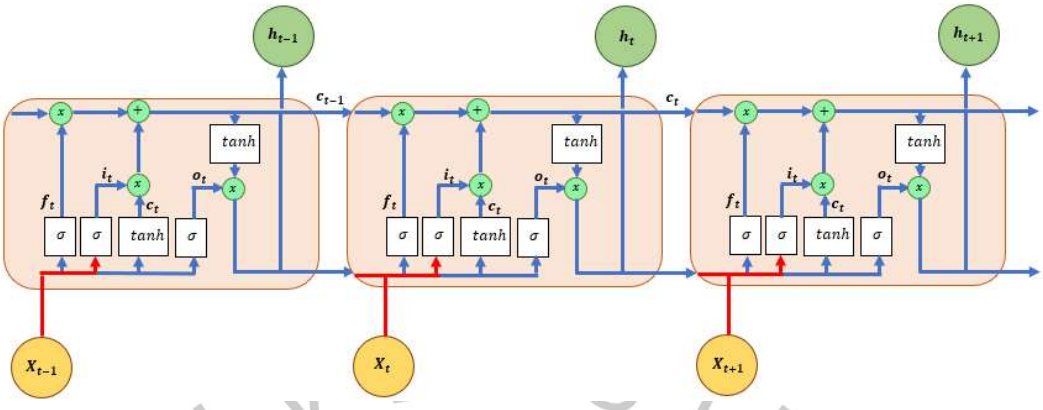
$$i_t = \sigma(W_i S_{t-1} + W_i X_t) \quad \text{.....} \quad (2.1)$$

dengan W_i adalah bobot dari *Input Gate,* S_{t-1} adalah *state* sebelumnya atau *state* pada waktu $t-1$. X_t adalah *input* pada waktu t dan σ adalah fungsi aktivitas *sigmoid.* Nilai gerbang *input* dikalikan dengan *output* dari lapisan kandidat ($\tilde{C}$). dengan rumus

$$\tilde{C} = tanh(W_c S_{t-1} + W_c X_t) \quad \text{.....} \quad (2.2)$$

$$c_t = (i_t * \tilde{C}_t + f_t * c_{t-1}) \quad \text{.....} \quad (2.3)$$

dimana $\tilde{C}$ adalah *intermediate cell state*, W_c adalah bobot dari *cell state,* S_{t-1} adalah *state* sebelumnya atau *state* pada waktu t–1 dan X_t adalah *input* pada waktu t. Pada **Gambar 2.6** *Input Gate* ditandai dengan garis berwarna merah.

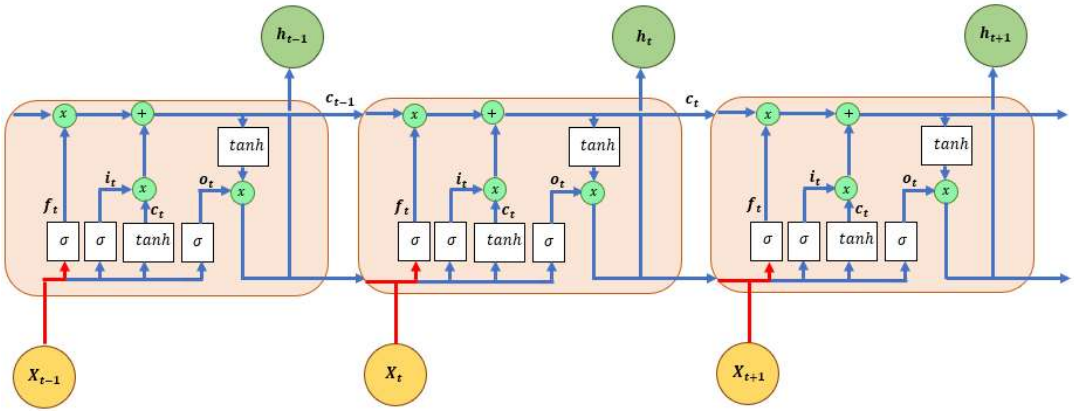


Gambar 2.6 *Input Gate* LSTM (Wiranda & Sadikin., 2019)

Gate selanjutnya pada sel memori LSTM adalah *forget gate* (f_t). Dimana *forget gate* adalah lapisan *sigmoid* yang mengambil *output* pada waktu t – 1 dan *input* pada waktu t dan menggabungkannya serta menerapkan fungsi aktivitas *sigmoid*. Karena *sigmoid*, *output* dari *gate* ini adalah 0 atau 1. Jika f_t = 0 maka keadaan (*state*) sebelumnya akan dilupakan, sementara jika f_t = 1 *state* sebelumnya tidak berubah. Rumus dari f_t adalah

$$f_t = \sigma(W_f S_{t-1} + W_f X_t) \quad \text{.....} \quad (2.4)$$

dengan W_f adalah bobot dari *forget gate*, S_{t-1} = *state* sebelumnya atau *state* pada waktu t -1, X_t adalah *input* pada waktu t dan σ adalah fungsi aktivitas *sigmoid*. Pada **Gambar 2.7** *Forget Gate* ditandai dengan garis berwarna merah.



Gambar 2.7 *Forget Gate* LSTM (Wiranda & Sadikin., 2019)

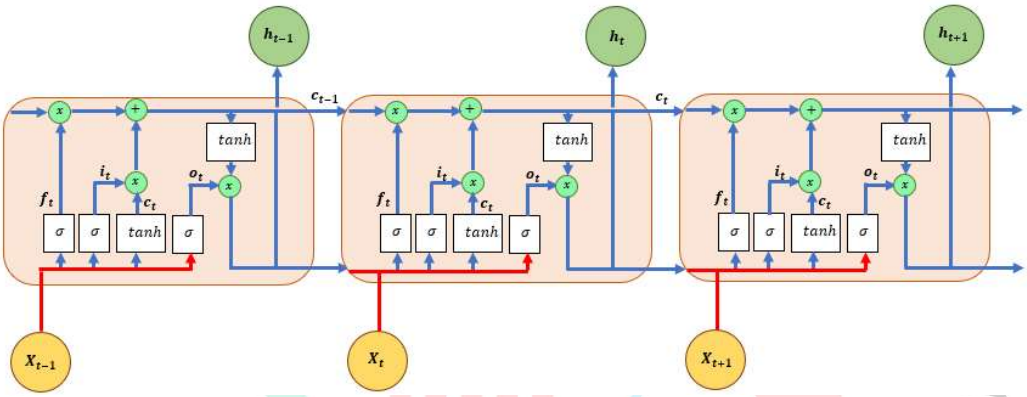
Gate terakhir pada sel memori LSTM adalah *output gate* (O_t). *Output gate* mengontrol seberapa banyak *state* yang lewat ke *output* dan bekerja dengan cara

yang sama dengan *gate* yang lainnya, dan terakhir menghasilkan *cell state* yang baru (h_t). Rumus dari *output gate* dan *cell state*

$$O_t = \sigma(W_o S_{t-1} + W_o X_t \quad \text{..... (2.5)}$$

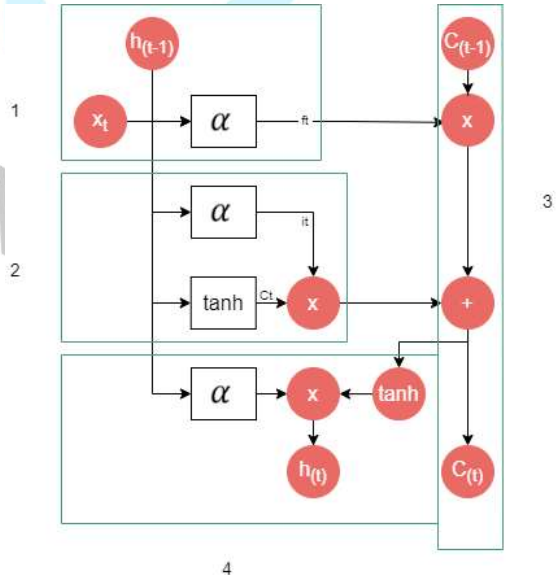
$$h_t = O_t * tanh(C_t) \quad \text{..... (2.6)}$$

dengan W_o adalah bobot dari *output gate*, S_{t-1} adalah *state* sebelumnya atau *state* pada waktu t – 1, X_t adalah *input* pada waktu t dan σ adalah fungsi aktivitas *sigmoid*. Pada **Gambar 2.8** *Output Gate* ditandai dengan garis berwarna merah.



Gambar 2.8 *Output Gate* LSTM (Wiranda & Sadikin., 2019)

Struktur jaringan pada Long Short Term Memory dapat dilihat pada **Gambar 2.9** berikut.



Gambar 2.9 Struktur Jaringan Long Short Term Memory

Berikut penjelasan dari struktur jaringan algoritma *Long Short Term Memory* pada **Gambar 2.9** diatas.

1. Pada proses nomor 1, vektor yang digunakan sebagai *input* (x_t) akan digabungkan dengan vektor dari *output* sebelumnya atau ($h_{(t-1)}$). Kemudian dengan menggunakan aktivasi *sigmoid*, *output* dari *forget gate* diubah menjadi angka yang berkisar antara nol sampai satu.
2. Langkah nomor 2 terdapat proses yang menghitung nilai baru dari input saat ini (x_t). Sebelumnya, *input* vektor dan *output* vektor digabungkan, dan pada grafik ada dua aktivasi, yang pertama dengan *sigmoid* dan yang kedua dengan tanh, setelah itu hasil aktivasi kedua dikalikan.
3. Langkah nomor 3 ada proses yang membandingkan status sel sebelumnya dengan *forget gate*, dan hasilnya akan dibandingkan dengan *cell state* sebelumnya.
4. Proses nomor 4 adalah prosedur untuk menghitung nilai gerbang x_t dan $h_{(t-1)}$ dan diaktifkan oleh fungsi sigmoid. Kemudian, untuk mengembalikan output sebagai nilai $h_{(t)}$, dikalikan dengan memori yang telah diaktifkan dengan tanh. Nilai ini diinputkan ke memori LSTM selanjutnya, dan memori yang tidak aktif dengan tanh diinputkan ke memori LSTM selanjutnya.

2.2.6 NLTK

Natual Language Toolkit (NLTK) adalah sebuah *library* pemrosesan dan analisis teks Python yang diciptakan oleh Steven Bird dan Edward Loper. *Library* ini digunakan untuk menganalisa teks dan menangani data teks menggunakan teknik-teknik seperti klasifikasi, tokenisasi dan lainnya. Tokenisasi, atau pemisahan kata dalam kalimat dan kalimat dalam paragraf adalah salah satu penggunaan NLTK. Misalnya kalimat “serombongan opa jalan pagi pake masker sambil goyang tangan dan ngobrolin efek vaksin sinovac di badan mereka dengan suara keras”, akan diubah menjadi ['serombongan', 'opa', 'jalan', 'pagi', 'pake', 'masker', 'sambil', 'goyang', 'tangan', 'dan', 'ngobrolin', 'efek', 'vaksin', 'sinovac', 'di', 'badan', 'mereka', 'dengan', 'suara', 'keras'] (Setyawati et al., 2021).

2.2.7 Keras

Keras adalah sebuah *package* yang bertindak sebagai pembungkus TensorFlow agar lebih ramah pengguna. Namun, Keras tidak hanya untuk TensorFlow tetapi dapat digunakan untuk yang lainnya seperti Theano dan CNTK (Amandri Achyar et al., 2022).

2.2.8 TextBlob

TextBlob adalah *library* dari Python 2 dan Python 3 yang digunakan untuk memproses data teks. *Library* TextBlob menyediakan API untuk pemrosesan *Natural Language Processing* (NLP) seperti ekstraksi frasa kata benda, analisis sentimen, klasifikasi, penerjemahan dan sebagainya. Pembelajaran bahasa alami diproses menggunakan hasil objek TextBlob dan pustaka TextBlob hanya dapat mengenali Bahasa Inggris (Parlika et al., 2020). Library TextBlob menggunakan skor yang telah ditetapkan, skor tersebut dapat dilihat pada tabel berikut.

Tabel 2.2 Skor Sentimen (Diwakar, 2021)

Skor	Label
> 0,7 - <= 1	*Excited*
> 0,3 - <= 0,7	*Happy*
> 0 - <= 0,3	*Tender*
== 0	*Neutral*
> 0,3 - < 0	*Sad*
> -0,7 - <= -0,3	*Scared*
> -1 - <= -0,7	*Angry*

2.2.9 Sastrawi

Sastrawi adalah *library* Bahasa Indonesia yang digunakan untuk menguraikan kata dan frasa menjadi istilah-istilah dasar. Algoritma dalam *library* ini merupakan modifikasi dari algoritma Nazief dan Adriani. Sebuah kata dibedah dan dihilangkan kata tambahannya untuk menyisakan kata dasarnya saja. Sebagai contoh kata "menyenangkan" akan disederhanakan menjadi kata "senang". Penggunaan Sastrawi cukup popular untuk *stemming* kata dalam Bahasa Indonesia. *Stemming* adalah aspek penting dalam *pre-processing* data karena memastikan bahwa sebuah kata atau kalimat tetap memiliki arti yang sama meskipun disajikan secara berbeda (Setyawati et al., 2021).

2.2.10 Shapiro-Wilk

Pada tahun 1965, Samuel Stanford Shapiro dan Martin Wilk merancang teknik Shapiro-Wilk. Metode ini muncul sebagai proses statistik alternatif untuk menilai normalitas seluruh sampel. Statistik uji didapatkan dengan membagi kuadrat dari kombinasi linear yang relevan dari statistik sampel dengan estimasi variansi simetris normal. Pada awalnya metode ini terbatas pada ukuran sampel kurang dari 50. Prosedur metode ini memverifikasi bahwa hipotesis nol (H_0) dihasilkan dari distribusi normal dengan mean dan variansi yang independent (Sintia, 2022). Rumus dari Shapiro Wilk sebagai berikut.

$$T_3 = \frac{1}{D}\left[\sum_{i=1}^{k} a_i (X_{n-i+1} - X_i)\right]^2 i = 1, 2, 3, \dots, n \quad \text{.....(2.7)}$$

dengan n adalah jumlah data, a adalah Coefficient test Shapiro-Wilk. Rumus dari nilai D dapat dilihat pada rumus berikut.

$$D = \sum_{i=1}^{n} (X_i - \bar{X})^2; i = 1, 2, 3, \dots, n \quad \text{.....(2.8)}$$

dengan $\bar{X}$ adalah rata-rata data.

2.2.11 Skewness

Skewness adalah metrik untuk asimetri dalam kaitannya dengan intensitas rata-rata. Skewness juga dikenal sebagai momen orde ketiga yang dinormalisasi. Angka negativ mengindikasikan bahwa distribusi kecerahan condong ke kiri dari rata-rata, sedangkan nilai positif mengindikasikan bahwa distribusi kecerahan condong ke kanan dari rata-rata. Rumus skewness digambarkan dalam

$$Skewness = \sum_{i=1}^{L-1} \frac{(X_i - \bar{X})^3}{N.\sigma^3} \quad \text{..... (2.9)}$$

dimana X_i adalah nilai individu dalam dataset, $\bar{X}$ adalah rata-rata dari dataset, σ adalah simpangan baku (standar deviasi) dari dataset, dan N adalah jumlah total data dalam dataset.