BAB II
TINJAUAN PUSTAKA

Pada bab ini disajikan tinjauan pustaka yang mencakup referensi penelitian sebelumnya serta tinjauan teoritis yang menjadi acuan bagi peneliti untuk memperkaya teori dalam melakukan perancangan aplikasi serta memperkuat landasan penelitian.

2.1 Pencapaian Terdahulu

Pada penelitian ini, referensi penelitian sebelumnya digunakan sebagai bahan rujukan dan landasan untuk penulis. Oleh karena itu, referensi yang diacu mencakup topik-topik mengenai penerapan metode komputasi yang sejenis.

Tabel 2. 1 Hasil Pencapaian Terdahulu

No.	Nama Peneliti	Publikasi	Judul	Hasil
1.	Junita Amalia, Juanda Pakpahan, Melani Pakpahan, Yeni Panjaitan	JATISI (Jurnal Teknik Informatika dan Sistem Informasi), 2022. *Natural Language Processing* (NLP)	Model Klasifikasi Berita Palsu Menggunakan Bidirectional LSTM Dan Word2Vec Sebagai Vektorisasi	Penelitian ini membuat model untuk mendeteksi berita palsu yang tersebar di media sosial dan platform sms di era digital. Model dilatih untuk mendeteksi berita palsu dari segi judul dan konten menggunakan metode Bidirectional LSTM dan *embedding* Word2Vec. Hasil pengujian menunjukkan bahwa akurasi tertinggi diperoleh untuk judul dan isi berita, yaitu 79,18% dan 92,80%. Penggunaan metode BiLSTM dapat secara efektif mendeteksi berita palsu dari segi judul, khususnya konten berita.
2.	Widi Afandi, Satria Nur Saputro, Andini Mulia Kusumaningrum, Hikari Ardiansyah, Muhammad Hilmi Kafabi, Sudianto	Jurnal Informatika: Jurnal pengembangan IT (JPIT), 2022. *Natural Language Processing* (NLP)	Klasifikasi Judul Berita Clickbait menggunakan RNN-LSTM	Penelitian ini membuat model untuk mengklasifikasi berita *clickbait* pada media elektronik. Model dilatih untuk mendeteksi judul berita yang bersifat sensasional atau tidak sesuai dengan isi berita. Model klasifikasi dilatih menggunakan arsitektur RNN-LSTM. Berdasarkan hasil pengujian, diperoleh akurasi sebesar 77%, sehingga dapat digunakan

				untuk mengklasifikasi judul berita *clickbait* dan *non-clickbait*.
3.	Dian Sukma Hani, Chanifah Indah Ratnasari	Jurnal Media Informatika Budidarma, 2023. *Natural Language Processing* (NLP)	Klasifikasi Masalah Pada Komunitas Marah-Marah di Twitter Menggunakan Long Short-Term Memory	Penelitian ini membuat model untuk mengklasifikasi tweet dalam sebuah komunitas media sosial twitter yaitu komunitas marah-marah. Model dilatih untuk mengklasifikasi jenis permasalahan yang dibahas dalam tweet tersebut. Pengklasifikasian dilakukan menggunakan algoritma LSTM dengan enam kategori yaitu studi, percintaan, keluarga, karier/pekerjaan, personal, dan umpatan. Berdasarkan hasil pengujian, model memperoleh akurasi sebesar 91,94%, sehingga model LSTM tersebut efektif dalam mengklasifikasikan *tweet* komunitas marah-marah.
4.	Muhammad Navi Nugraha, Muhammad Arrofiq	Journal of Internet and Software Engineering (JISE), 2024. *Natural Language Processing* (NLP)	Purwarupa Sistem Klasifikasi Legalitas Investasi Berbasis Algoritma Bidirectional Long Short Term Memory	Penelitian ini membuat aplikasi sederhana berbasis website yang diintegrasikan dengan model *machine learning* untuk mengklasifikasi pesan iklan investasi. Model dilatih menggunakan dataset hasil *scraping* dari grup media sosial Telegram. Pengklasifikasian dilakukan menggunakan algoritma Bidirectional LSTM dengan dua kategori yaitu legal dan ilegal. Berdasarkan hasil pengujian, didapatkan akurasi terbesar model yaitu 96%, sehingga model tersebut dapat digunakan untuk mengklasifikasikan pesan investasi legal dan ilegal.
5.	Steven Dharmawan, Viny Christanti Mawardi, Novario Jaya Perdana	Jurnal Ilmu Komputer dan Sistem Informasi (JIKSI), 2023. *Natural Language Processing* (NLP)	Klasifikasi Ujaran Kebencian Menggunakan Metode FeedForward Neural Network (IndoBERT)	Penelitian ini mengembangkan aplikasi sederhana berbasis website yang diintegrasikan dengan model *machine learning* untuk melakukan mengklasifikasikan sebuah komentar. Model dilatih menggunakan *dataset* hasil *scraping* dari twitter dan

		youtube. Klasifikasi dilakukan menggunakan metode FeedForward Neural Network dan IndoBERT dengan dua kategori yaitu normal dan ujaran kebencian. Berdasarkan hasil pengujian, diperoleh nilai akurasi terbaik sebesar 89,52% dengan catatan model cenderung mengklasifikasi komentar ke dalam kategori ujaran kebencian jika terdapat kata kasar.

2.2 Tinjauan Teoritis

Tinjauan teoritis berfungsi sebagai fondasi dalam mendukung teori yang relevan dengan topik pada penelitian ini. Penjelasan rinci mengenai setiap tinjauan teoritis yang diaplikasikan pada penelitian ini, diuraikan sebagai berikut.

2.2.1 *Artificial Intelligence*

Artificial Intelligence (AI) merupakan salah satu bidang dalam ilmu komputer yang didedikasikan untuk pengembangan mesin yang mampu meniru tugas-tugas yang biasanya memerlukan kemampuan kognitif manusia. Penciptaan sistem yang memanfaatkan AI bertujuan untuk menciptakan mesin yang dapat belajar, bernalar, dan melakukan tindakan dengan cara yang mirip dengan manusia. Dalam domain AI yang luas, terdapat berbagai macam teknik dan metodologi yang dapat digunakan, termasuk pemrosesan bahasa alami. Pemrosesan bahasa alami memungkinkan mesin untuk memahami seluk-beluk kontekstual bahasa dan memahami makna secara spesifik kata-kata dalam kalimat.

2.2.2 *Deep Learning*

Deep learning adalah algoritma jaringan saraf tiruan yang memanfaatkan data sebagai masukkan dan memprosesnya dengan memanfaatkan beberapa lapisan tersembunyi (Zakiyamani, Cahyani, Riana, & Hardianti, 2022). *Deep learning* memanfaatkan pendekatan pembelajaran yang terinspirasi oleh struktur otak manusia, sehingga memungkinkan komputer untuk belajar dari data yang sangat besar dan kompleks. Pendekatan pembelajaran tersebut terdiri dari berbagai

macam algoritma yang dirancang agar dapat mudah beradaptasi dan dapat diterapkan dalam berbagai skenario pembelajaran. Algoritma ini juga diterapkan pada tugas-tugas seperti pengenalan wajah, gambar, suara, klasifikasi teks, dan lain sebagainya.

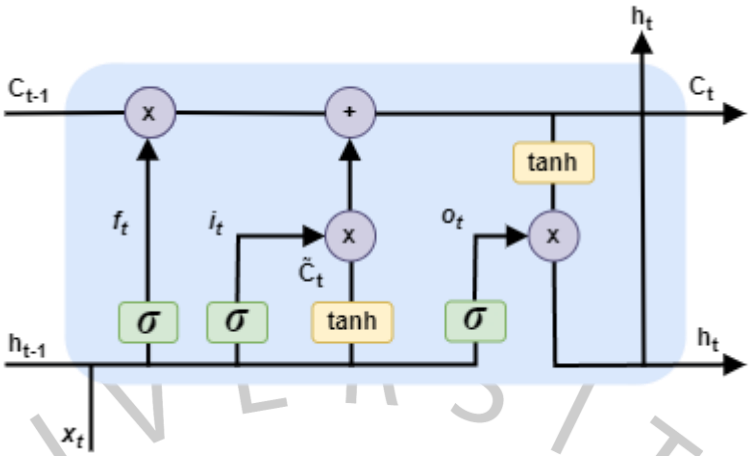
2.2.3 *Natural Language Processing*

Natural Language Processing (NLP) adalah bidang studi yang berfokus pada penelitian dan pengembangan metode komputer untuk memahami dan memproses bahasa manusia, baik dalam bentuk teks atau ucapan (Jesse & Belinkov, 2019). Tahapan dalam NLP melibatkan langkah pra-pemrosesan untuk mengubah data teks atau ucapan menjadi informasi yang dapat dipahami oleh komputer. Langkah-langkah umum tersebut mencakup memecah teks menjadi kata, membersihkan data dari *noise*, menghilangkan kata *stopword*, dan mengubahnya kata ke bentuk dasar. Setelah tahap pra-pemrosesan, data tersebut siap untuk dianalisis lebih lanjut menggunakan berbagai teknik NLP, seperti analisis sentimen, klasifikasi teks, dan *text-to-speech*.

2.2.4 *Long Short-Term Memory* (LSTM)

Menurut (Pratama, Wijaya, Santosa, & Tamba, 2023), Long Short-Term Memory (LSTM) merupakan jenis khusus dari jaringan saraf tiruan (*neural network*) yang dirancang untuk mengatasi masalah pemodelan dan prediksi pada data yang memiliki urutan atau pola tertentu. LSTM mampu mengingat informasi lampau yang muncul sebelumnya, sehingga memungkinkan prediksi yang lebih akurat. Kemampuan mengingat informasi dalam jangka panjang, membuat LSTM digunakan di berbagai tugas yang membutuhkan pemahaman konteks, seperti analisis sentimen dan klasifikasi teks.

Arsitektur LSTM memungkinkan adanya proses penambahan atau penghapusan kumpulan informasi pada memori *cell state* atau kondisi sel dengan struktur yang disebut dengan *gates* atau gerbang. Gerbang berfungsi sebagai komponen yang menjaga informasi dan mengontrol aliran data di dalam jaringan. Gerbang yang terdiri dari tiga tiga komponen inti yaitu *input gate*, *forget gate*, dan *output gate* terdapat pada setiap unit. Adapun penjelasan mengenai keterkaitan setiap komponen pada gerbang dan kondisi sel, dijabarkan sebagai berikut:



Gambar 2. 1 Arsitektur LSTM

a) *Input Gate*

Gerbang pertama dalam proses LSTM adalah *input gate* (i_t) yang berperan untuk menerima informasi lampau dari sel pada waktu sebelumnya dan informasi terbaru yang berasal dari masukkan pada keadaan saat ini atau waktu t. Informasi tersebut selanjutnya digabung, lalu diproses oleh lapisan *sigmoid* dan *tanh*. Lapisan *sigmoid*, yang juga dikenal dengan sebutan "*input gate layer*" berperan untuk memilah informasi mana yang akan diperbarui dengan mengubah nilai menjadi 0 hingga 1. Selanjutnya, lapisan $tanh$ akan membuat vektor untuk menampung kandidat nilai baru atau $\tilde{C}$ ke dalam kondisi sel yang hasilnya berupa nilai -1 hingga 1. Langkah selanjutnya yaitu menggabungkan hasil dari masing-masing lapisan untuk memperbarui kondisi sel. Adapun persamaan *input gate*, sebagai berikut :

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

$$\widetilde{C_t} = tanh(W_c.[h_{t-1}, x_t] + b_c) \quad (2)$$

Dimana pada i_t atau *input gate*, W_i adalah bobot dari *input gate* pada waktu saat ini, h_{t-1} adalah luaran dari *timestamp* sebelumnya, x_t adalah masukkan dari *timestamp* saat ini, b_i adalah nilai bias pada *input gate*, σ adalah lapisan *sigmoid*, $\widetilde{C_t}$ adalah kandidat nilai baru dari *timestamp* saat ini, tanh adalah lapisan tanh, W_C adalah berat untuk kandidat $\widetilde{C_t}$, dan b_c adalah nilai bias untuk kandidat $\widetilde{C_t}$.

b) *Forget Gate*

Gerbang kedua dalam proses LSTM adalah *forget gate* (f_t), atau gerbang lupa. Gerbang ini berfungsi untuk menentukan bobot yang diberikan pada informasi masa lalu dalam kondisi sel. Bobot ini yang akan menentukan informasi mana yang akan dipertahankan dan dihapus, sehingga LSTM dapat fokus pada informasi yang relevan untuk prediksi saat ini atau waktu t. *Forget gate* terdiri dari lapisan *sigmoid* yang berperan untuk menyaring informasi dengan memberikan luaran berupa 0 dan 1. Pada prosesnya, *forget gate* memerlukan informasi berupa *hidden state* yang berasal dari *cell* pada waktu sebelumnya dan informasi saat ini. Setelah itu menerapkan fungsi aktivasi *sigmoid*. Jika $f_t = 0$ maka informasi pada keadaan sel sebelumnya akan dibuang, sementara jika $f_t = 1$ maka informasi akan disimpan. Berikut adalah persamaan dari *forget gate*.

$$f_t = \sigma(W_f . [h_{t-1}, x_t] + b_f) \quad (3)$$

Dimana pada f_t atau *forget gate*, W_f adalah bobot dari *forget gate* pada waktu saat ini, h_{t-1} adalah luaran berupa vektor dari *timestep* sebelumnya, x_t adalah masukkan dari *timestep* saat ini, dan b_f adalah nilai bias pada *forget gate*.

c) *Cell State*

Komponen *cell state* pada jaringan LSTM merupakan komponen vital yang berfungsi sebagai penyimpanan informasi jangka panjang. Komponen ini berperan untuk memperbarui kondisi sel pada waktu sebelumnya atau C_{t-1} menjadi kondisi sel baru pada waktu saat ini yang ditandai dengan C_t. Luaran yang dihasilkan oleh jaringan LSTM sebelumnya atau h_{t-1} akan dikalikan dengan f_t. Tujuannya adalah agar *cell state* hanya mengingat informasi yang dianggap penting dan melupakan informasi lainnya. Setelah itu, hasilnya akan dijumlahkan dengan hasil perkalian antara i_t dengan $\widetilde{C}_t$. Adapun persamaan untuk proses pembaruan kondisi sel sebagai berikut.

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C}_t \quad (4)$$

Dimana pada C_t atau *cell state*, f_t adalah nilai *forget gate* pada waktu saat ini, C_{t-1} adalah kondisi sel pada *timestamp* sebelumnya atau $t-1$, i_t adalah nilai *input gate* pada *timestamp* saat ini atau t, dan $\widetilde{C_t}$ adalah kandidat *cell state* baru.

d) *Output Gate*

Pada gerbang *output gate* (o_t), terjadi proses penentuan luaran berupa informasi yang akan dibagikan. Langkah pertama, informasi akan diproses oleh lapisan *sigmoid* untuk menentukan informasi apa yang akan kita jadikan sebagai luaran. Selanjutnya informasi tersebut diproses oleh lapisan *tanh* yang akan mengubah informasi tersebut menjadi rentang antara -1 dan 1. Hasil tersebut akan menentukan seberapa penting informasi pada waktu saat ini untuk diteruskan ke sel LSTM berikutnya. Berikut adalah persamaan *output gate*.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \quad (5)$$

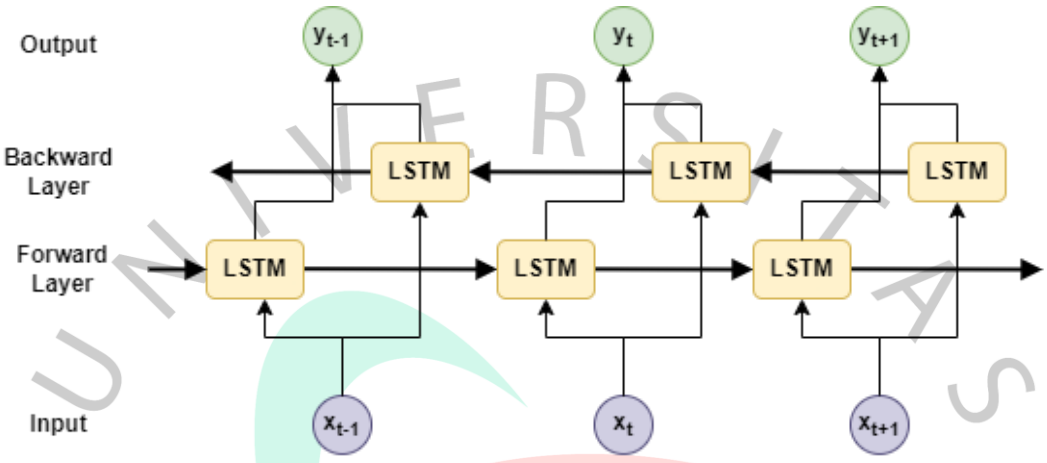
$$h_t = o_t * \tanh(C_t) \quad (6)$$

Dimana o_t atau *output gate*, σ adalah fungsi aktivasi lapisan *sigmoid*, W_o adalah bobot pada *output gate*, h_{t-1} adalah keadaan tersembunyi pada *timestamp* sebelumnya atau $t-1$, x_t adalah masukkan dari stempel waktu saat ini atau t, b_o adalah nilai bias dari *output gate*. Sementara h_t atau keadaan tersembunyi pada stempel waktu saat ini, *tanh* adalah fungsi aktivasi *hyperbolic tangen*, dan C_t adalah kondisi sel yang baru pada stempel waktu saat ini atau t.

2.2.5 Bidirectional Long Short-Term Memory (BiLSTM)

BiLSTM (*Bidirectional Long Short-Term Memory*) dikembangkan dari arsitektur *Long Short-Term Memory* (LSTM) yang dirancang untuk meningkatkan kemampuan dalam memahami konteks kata dalam sebuah kalimat. Berbeda dengan LSTM yang hanya memproses informasi secara *backward*, BiLSTM memproses informasi secara dua arah, yaitu *forward* dan *backward*. Pemrosesan secara dua arah ini memungkinkan model untuk menangkap konteks dari keadaan sebelumnya dan yang akan datang dalam urutan, sehingga memberikan

pemahaman yang lebih menyeluruh terhadap teks (Airlangga, 2024). Dengan kemampuannya, BiLSTM telah banyak digunakan dalam berbagai aplikasi pemrosesan bahasa alami seperti klasifikasi teks, ringkasan teks, terjemahan mesin, dan lainnya.



Gambar 2. 2 Arsitektur Bidirectional LSTM

Gambar 2.2 merupakan ilustrasi arsitektur BiLSTM yang di dalamnya terdapat tiga unit LSTM dengan *timestep* yang terdiri dari waktu lampau atau $(t-1)$, saat ini (t), dan waktu berikutnya $(t+1)$. Lapisan *forward* pada BILSTM memproses informasi dari awal kalimat ke akhir dengan masukkan dari *timestep* sebelumnya dan *timestep* saat ini. Lapisan *backward* memproses informasi dari akhir kalimat ke awal dengan masukkan yang diperoleh dari *timestep* saat ini dan *timestep* berikutnya. Pada dasarnya luaran yang dihasilkan BiLSTM adalah penjumlahan antara nilai probabilitas *output* pada *timestep* sebelumnya atau $t-1$ dengan nilai probabilitas *output* pada *timestep* setelahnya atau $t+1$. Adapun persamaan BiLSTM adalah sebagai berikut:

$$y_t = \overrightarrow{h_t} + \overleftarrow{h_t} \tag{7}$$

Dimana y_t adalah *output* BiLSTM, $\overrightarrow{h_t}$ adalah *output* yang dihasilkan dari lapisan *forward*, dan $\overleftarrow{h_t}$ adalah *output* yang dihasilkan dari lapisan *backward*.

2.2.6 *Text Classification*

Text classification merupakan salah satu teknik yang digunakan dalam penerapan pemrosesan bahasa alami. Menurut (Miranda, Gabriella, Wahyudi, &

Chai, 2023), *text classification* adalah proses menandai teks dengan label kategori tertentu dari sekumpulan data yang telah ditetapkan sebelumnya. Teknik ini memungkinkan pengelompokan teks ke dalam kategori-kategori yang relevan, sehingga informasi yang memiliki makna tertentu dapat diekstraksi dari kumpulan teks yang besar.

Dalam penerapannya, teknik ini dapat digunakan untuk mem*filter* email *spam*, mengklasifikasi sentimen ulasan produk menjadi positif atau negatif, dan membuat *bot* yang dapat menjawab pertanyaan sesuai dengan *keyword* tertentu. Adapun *tools* yang digunakan dalam merancang sistem pada penelitian ini, yaitu:

a) Python

Python merupakan bahasa pemrograman canggih yang dikenal karena kemudahan mempelajarinya. Diciptakan oleh Guido van Rossum pada akhir tahun 1980-an, Python berhasil menjadi pilihan utama pada pengembangan perangkat lunak, termasuk pengembangan web, kecerdasan buatan, dan otomasi sistem. Python dirancang dengan tujuan khusus agar kode dan sintaks dapat dibaca secara sederhana, membuatnya ideal untuk pemula maupun programmer berpengalaman.

b) Google Colaboratory

Google Colaboratory merupakan platform pengembangan berbasis *cloud* yang diciptakan Google. Platform *cloud* ini dirancang khusus untuk memenuhi kebutuhan pemrograman, analisis data, dan pembelajaran mesin. Keunggulan utama platform ini adalah dapat menjalankan kode Python secara interaktif. Pengguna dapat memanfaatkan platform ini untuk menjalankan kode Python pada *environment* Jupyter Notebook yang dapat diakses secara lokal melalui browser.

c) NLTK

Natural Language Toolkit (NLTK) adalah salah satu pustaka yang dirancang untuk memfasilitasi pengguna dalam mengolah data berupa teks. NLTK menyediakan berbagai macam fungsi yang memfasilitasi pengguna untuk melakukan beragam tugas pemrosesan bahasa alami. Fungsi-fungsi tersebut mencakup tokenisasi kalimat, mengubah kata menjadi bentuk

dasarnya, dan mengubah teks menjadi format yang dapat memudahkan pemrosesan. NLTK dapat menjadi pilihan yang tepat bagi para peneliti, pengembang, dan praktisi yang ingin menjelajahi dunia pemrosesan bahasa alami.

d) Keras

Keras adalah salah satu *Application Programming Interface* (API) yang dibuat untuk menyederhanakan proses pengembangan model *deep learning* menggunakan Python. *Keras* berperan sebagai jembatan antara pengguna dan *library deep learning* sehingga menjadi lebih mudah dan sederhana. *Library* yang disediakan oleh *Keras* antara lain seperti Tensorflow, Theano, dan CNTK.

e) *SKLearn*

Scikit-Learn atau yang biasa disingkat *SKLearn* adalah *library* pada bahasa pemrograman Python yang dimanfaatkan untuk pengembangan *machine learning*. *Library* ini menyediakan berbagai fitur untuk analisis dan pemodelan data, termasuk algoritma untuk klasifikasi, regresi, *clustering*, dan algoritma serupa. Fitur *SKLearn* yang digunakan oleh peneliti yaitu fungsi untuk memisahkan data latih dan data uji pada dataset.

2.2.7 Black Box

Metode pengujian *black box* merupakan pengujian kualitas perangkat lunak yang difokuskan pada aspek fungsionalitas dari perangkat lunak tersebut (Wijaya & Astuti, 2021). Pengujian black box dilakukan dengan maksud untuk menemukan fungsi yang belum berjalan sesuai dengan tujuan, kesalahan pada *interface*, masalah dalam struktur data, kendala kinerja, kesalahan pada proses awal hingga akhir. Pada dasarnya, pengujian ini memastikan bahwa semua fitur pada perangkat lunak bekerja sesuai dengan tujuan dan ketentuan awal.

2.2.8 White Box

Metode pengujian menggunakan *white-box testing* berfokus pada analisis struktur internal kode program. Berbeda dengan pengujian *black box*, metode ini berfokus pada pengujian jalur eksekusi, percabangan kondisional, dan logika dalam kode untuk mengidentifikasi kesalahan atau kegagalan yang dapat terjadi. Penguji yang menerapkan white box sebagai metode pengujian perangkat lunak

harus mengetahui dan memahami kode sumber perangkat lunak tersebut secara mendalam (Sie, Musdar, & Bahri, 2022).