BAB II
TINJAUAN PUSTAKA DAN DASAR TEORI

2.1. Tinjauan Pustaka

Sentiment analysis merupakan salah satu tugas utama dalam *Natural Language Processing* (NLP). NLP adalah bidang studi yang berfokus pada pengembangan algoritma dan sistem yang memungkinkan komputer untuk memahami dan memproses bahasa manusia. NLP juga dikenal sebagai linguistik komputasi, ucapan komputasi, dan pemrosesan bahasa. Tujuan utama NLP adalah mengatasi tantangan dalam pemahaman bahasa alami manusia dengan semua aturan gramatikal dan semantiknya, serta mengubah bahasa menjadi representasi informasi yang dapat diproses oleh komputer (Pustejovsky & Stubbs, 2012).

Sentiment analysis adalah proses untuk mengidentifikasi dan mengevaluasi sentimen, opini, atau sikap yang terkandung dalam teks, seperti ulasan produk, media sosial, atau komentar. Tujuan utamanya adalah untuk memahami perasaan atau pandangan yang terkait dengan suatu topik atau entitas tertentu. Ada dua teknik yang digunakan dalam *sentiment analysis*, yaitu *machine learning* dan *lexicon-based* (Medhat dkk., 2014; Tul dkk., 2017). Dalam penelitian ini pendekatan yang digunakan adalah *machine learning* khususnya *deep learning (DL)*. Tujuannya adalah untuk mendekatkan *machine learning* dengan *artificial intelligence (AI)*.

Metode *machine learning* klasik telah berkembang menjadi *deep learning*. Perkembangan ini juga mempengaruhi perkembangan teknik *sentiment analysis* dengan menggunakan metode *deep learning*. Metode *machine learning* klasik dapat digunakan untuk menganalisis sentimen teks. Penelitian yang dilakukan oleh (Tripathy dkk., 2016) membandingkan metode *machine learning* klasik dan *deep learning* dalam menganalisis sentimen ulasan *e-marketing* dalam format teks Urdu Romawi. Metode DL yang dibandingkan adalah *Long Short Term Memory (LSTM)*, sedangkan metode *machine learning* klasik yang digunakan adalah *Naïve Bayes (NB), Random Forest (RF),* dan *Support Vector Machine (SVM)*. Hasil penelitian menunjukkan bahwa LSTM memiliki akurasi terbaik, yaitu sebesar 95%, dibandingkan dengan SVM (92%), RF (88%), dan NB (77%).

LSTM adalah metode analisis sentimen yang efisien karena tidak memerlukan pemilihan fitur dan merupakan pengembangan dari RNN untuk mengatasi masalah gradien yang memudar (Ghulam dkk., 2019). Jaringan LSTM efektif dalam analisis sentimen karena kemampuannya untuk menangkap dan mempertahankan ketergantungan jangka panjang (Y. Huang dkk., 2023; Q. Liu dkk., 2023; Singh & Sharma, 2023). LSTM dirancang untuk mengatasi masalah difusi gradien serta memperbaiki keterbatasan dalam penyimpanan memori selektif. LSTM juga menambahkan mekanisme perhatian yang membantu dalam mempertahankan konteks kata-kata dalam teks (Hochreiter dan Schmidhuber, 1997). Keunggulan utama LSTM adalah adanya sel memori dengan gerbang yang dapat menyimpan, menghapus, dan memperbarui data secara selektif, membuatnya lebih efektif dalam analisis sentimen tekstual dibandingkan dengan RNN tradisional (Xu dkk., 2016).

Tantangan yang dihadapi LSTM adalah kesulitan menangkap korelasi semantik kontekstual dan mengelola dominasi gradien jangka pendek, sehingga membatasi pembelajaran informasi jarak jauh (Y. Huang dkk., 2023; Zhang dkk., 2023). Tantangan lainnya meliputi kesulitan dalam penilaian kesalahan sebelum dan setelah implementasi (Liu, Guo, dan Mahmud, 2021), sifat *black box* dari model *deep learning* yang membuat interpretasi hasil sulit karena fitur berdimensi tinggi dan bobot yang tidak terduga (Kentour dan Lu, 2021), serta tantangan dalam menciptakan set pelatihan berkualitas tinggi dengan pelabelan yang akurat akibat kompleksitas dan subjektivitas anotasi sentimen (Wu dkk., 2022). Untuk meningkatkan model analisis sentimen berbasis LSTM, penting untuk fokus pada peningkatan interpretasi, manajemen gradien, deteksi kesalahan, dan strategi pelabelan.

Topic modelling adalah pendekatan *text mining* yang cukup handal untuk menemukan hubungan antara data tekstual tersembunyi dan teks dalam korpus (Jelodar dkk., 2019). Pendapat lain menyebutkan bahwa *topic modelling* adalah pembelajaran tanpa pengawasan karena data yang digunakan tidak memiliki pengidentifikasi (Putra dan Kusumawardani, 2017). Beberapa teknik *topic modeling* adalah *Latent Dirichlet Allocation (LDA), Non-negative Matrix Factorization*

(NMF), Probabilistic Latent Semantic Analysis (PLSA), Hierarchical Dirichlet Process (HDP) dan *Correlated Topic Model (CTM)*. LDA memodelkan dokumen sebagai kombinasi topik berdasarkan probabilitas, sementara NMF, PLSA, HDP, dan CTM memiliki pendekatan masing-masing untuk topik modeling. LDA dipilih karena kemampuannya dalam menginterpretasikan topik, menangani data besar, dan mengelola variasi topik secara efektif.

2.2. Dasar Teori

Penelitian terdahulu definisi *text pre-processing* tersaji di Tabel 2.1 berikut :

Tabel 2.1. Penelitian Terdahulu Definisi *Text Pre-Processing*

No.	Peneliti	Judul	Definisi
1.	Kadhim (2018)	*An Evaluation of Preprocessing Techniques for Text Classification*	*Text pre-processing* merupakan langkah krusial dalam klasifikasi teks yang melibatkan teknik seperti *tokenizing, stopword removal*, serta *stemming* dan *lemmatization* untuk memperbaiki data sebelum analisis. Proses ini membersihkan dan mempersiapkan data mentah, sehingga meningkatkan akurasi dan kinerja model klasifikasi.
2.	Hickman dkk. (2022)	*Text Preprocessing for Text Mining in Organizational Research: Review and Recommendations*	*Text pre-processing* adalah langkah awal penting dalam analisis teks dan penambangan teks di penelitian organisasi. Proses ini mencakup teknik-teknik seperti *cleansing, normalization, tokenizing, stopword removal*, serta *stemming* dan *lemmatization*. Tujuannya adalah untuk meningkatkan kualitas data teks agar lebih mudah dianalisis.
3.	Chai (2023)	*Comparison of text preprocessing methods*	*Text pre-processing* adalah langkah penting dalam persiapan data untuk pemodelan dan aplikasi pemrosesan bahasa alami. Melibatkan berbagai teknik seperti *tokenizing*, *normalization*, *stemming* dan *lemmatization*, tujuannya adalah membersihkan dan mempersiapkan data untuk penggunaan efektif dalam model komputer. Metode ini disesuaikan dengan karakteristik data dan kebutuhan aplikasi untuk hasil analisis yang optimal.

Dari beberapa definisi tersebut dapat disimpulkan bahwa *text pre-processing* memainkan peran krusial dalam berbagai konteks seperti klasifikasi teks, analisis teks, dan pemodelan NLP. Tujuannya adalah meningkatkan kualitas data, akurasi model, dan efektivitas aplikasi NLP. Proses *text pre-processing* mencakup *Case folding*, *Tokenizing, Normalization, Negation handling, Stopwords removal, stem-ming, padding*

Case folding adalah proses standarisasi huruf dalam teks sehingga semuanya menjadi huruf kecil atau huruf besar. Dalam penelitian ini, dilakukan case folding untuk mengubah seluruh data menjadi huruf kecil. Umumnya, huruf kapital digunakan di awal kalimat, seperti pada contoh "Udara di tempat ini sangat sejuk". Namun, dengan case folding, kalimat tersebut diubah menjadi "udara di tempat ini sangat sejuk" dengan huruf kecil di awal kalimat (Hidayatullah & Ma'arif, 2017). Tujuannya untuk memastikan konsistensi dalam penggunaan huruf kecil di seluruh teks. Dengan menggunakan case folding, variasi kata dengan huruf kapital dapat *dihindari*. Hal ini membuat data teks lebih terstandarisasi dan memudahkan analisis, mengurangi kebingungan atau kesalahan akibat variasi huruf kapitalisasi.

Tokenizing adalah langkah krusial dalam mengolah teks yang melibatkan pemecahan dokumen teks menjadi unit-unit yang disebut token. Tujuan utama dari *tokenizing* adalah membagi teks ke dalam bagian-bagian yang lebih kecil seperti kata, kalimat, atau simbol yang memiliki makna sendiri. Dalam proses *tokenizing*, karakter-karakter yang dianggap tidak relevan atau ilegal, seperti tanda baca dan simbol tertentu, perlu dihapus atau diabaikan. Ini termasuk elemen-elemen seperti *hashtag* (#), *username* (@), *URL* (http://....), serta simbol seperti persen (%), *ampersand* (&), dan tanda panah (>). Selain itu, tanda kurung buka dan tutup ((,),{,}) serta tanda kurung siku ([,]) juga dihapus. Penghapusan karakter-karakter ini penting untuk memastikan bahwa token yang dihasilkan bersih dan fokus pada informasi yang relevan, sehingga meningkatkan kualitas analisis teks yang akan dilakukan. Dengan menghapus karakter-karakter tidak penting, *tokenizing* mempersiapkan teks untuk analisis lanjutan seperti penghitungan frekuensi kata atau identifikasi entitas. Proses ini meningkatkan pemahaman terhadap struktur teks, memungkinkan pemrosesan lebih efisien, dan membantu membersihkan teks dari elemen yang tidak relevan, meningkatkan kualitas analisis teks (Symeonidis dkk., 2018).

Normalization adalah teknik penting dalam *text pre-processing* yang bertujuan untuk meningkatkan kinerja model machine learning dengan menstandarisasi nilai variabel ke skala yang seragam. Metode umum (*min-max scaling, z-score normalization*, dan *L2 normalization*) dapat diterapkan

menggunakan *library NumPy* dan *scikit-learn* di *Python* (Raschka & Mirjalili, 2019). Penting juga untuk menangani *missing values* dengan hati-hati, seperti menghapus data yang hilang atau menggantinya dengan nilai rata-rata (Pyle, 1999). Meskipun kedua sumber yang disebutkan membahas *normalization*. (Raschka & Mirjalili, 2019) lebih fokus pada konteks *machine learning*, sementara (Pyle, 1999) lebih menekankan pada *data mining*.

Negation handling adalah proses yang bertujuan untuk mengidentifikasi dan mengelola kata-kata negatif dalam teks, sehingga dapat meningkatkan kualitas analisis yang dilakukan oleh program pemrosesan bahasa alami (NLP). Beberapa teknik yang sering digunakan dalam langkah ini meliputi penandaan negasi, pemrosesan ketergantungan, dan metode berbasis aturan. Setelah kata-kata negatif berhasil terdeteksi, berbagai teknik seperti pembalikan makna (*flip*), pergeseran makna (*shifting*), atau intensifikasi dapat diterapkan untuk memodifikasi arti kata tersebut, sehingga analisis menjadi lebih akurat dan efektif (Jurafsky & Martin, 2000).

Stopwords removal adalah proses yang bertujuan untuk menghapus kata-kata umum yang dianggap tidak memiliki relevansi signifikan dalam konteks analisis teks. Kata-kata ini, seperti "ja," "la," "in," "to," dan sejenisnya, sering kali tidak memberikan informasi substansial dan dapat mengganggu pemahaman makna yang lebih dalam dari teks. Dengan *stopwords removal*, ukuran dataset menjadi lebih kecil, yang tidak hanya mengurangi beban komputasi tetapi juga mempercepat proses analisis. Dalam proses ini algoritma lebih fokus pada kata-kata kunci yang memiliki bobot informasi lebih tinggi, sehingga dapat menghasilkan wawasan yang lebih berarti dalam analisis data. *Stopwords removal* sangat bermanfaat dalam berbagai aplikasi, termasuk analisis teks, pemrosesan bahasa alami (NLP), dan indeksasi dokumen, di mana efisiensi dan relevansi informasi adalah kunci untuk menghasilkan hasil yang akurat dan bermanfaat. Dengan demikian, *stopwords removal* menjadi langkah penting dalam mempersiapkan data untuk analisis yang lebih mendalam dan tepat (Symeonidis dkk., 2018).

Tabel 2.2. Penelitian Terdahulu Definisi *Stemming*

No.	Peneliti	Judul	Definisi
1.	Hidayatullah & Ma'arif, (2017)	*Pre-processing Tasks in Indonesian Twitter Messages*	*Stemming* adalah proses menyederhanakan kata ke bentuk dasarnya dengan menghapus imbuhan. Sastrawi, *library* populer untuk bahasa Indonesia, menggunakan algoritma Nazief-Adrian yang telah diperbaiki dengan *Confix Stripping (CS)* dan ditingkatkan dengan *Enhanced Confix Stripping (ECS)* serta dimodifikasi dengan *Modified ECS*.
2.	(Adriani dkk., 2007)	*Stemming Indonesian : A confix-stripping approach*	*Stemming* adalah proses penyederhanaan kata dengan menghapus akhirannya, bermanfaat dalam berbagai aplikasi seperti pencarian informasi, terjemahan, dan pengelompokan kata. Sastrawi, sebuah metode penting untuk bahasa Indonesia, menggunakan algoritma CS yang dianggap inovatif dan akurat dalam proses *stemming*. Metode ini membantu dalam menemukan informasi dalam teks secara efisien, terutama dalam pencarian informasi melalui *query ad hoc*.

Dari berbagai definisi, dapat disimpulkan bahwa *stemming* adalah proses menyederhanakan kata ke bentuk dasarnya dengan menghapus imbuhan, yang berguna untuk pencarian informasi, terjemahan, dan pengelompokan kata. Sastrawi menggunakan algoritma Nazief-Adrian yang dimodifikasi untuk meningkatkan akurasi dan efisiensi. Pendekatan ini mengklasifikasikan imbuhan dalam Bahasa Indonesia menjadi tiga kategori: *inflection suffixes*, *derivation suffixes* (DS), dan *derivation prefixes* (DP). Proses *stemming* dalam Sastrawi juga mengikuti beberapa aturan, seperti larangan kombinasi imbuhan tertentu, penerapan imbuhan yang sama lebih dari sekali, dan ketentuan bahwa kata dengan satu atau dua huruf tidak di-*stemming*. Aturan ini membantu Sastrawi memetakan kata dengan lebih tepat ke bentuk dasarnya.

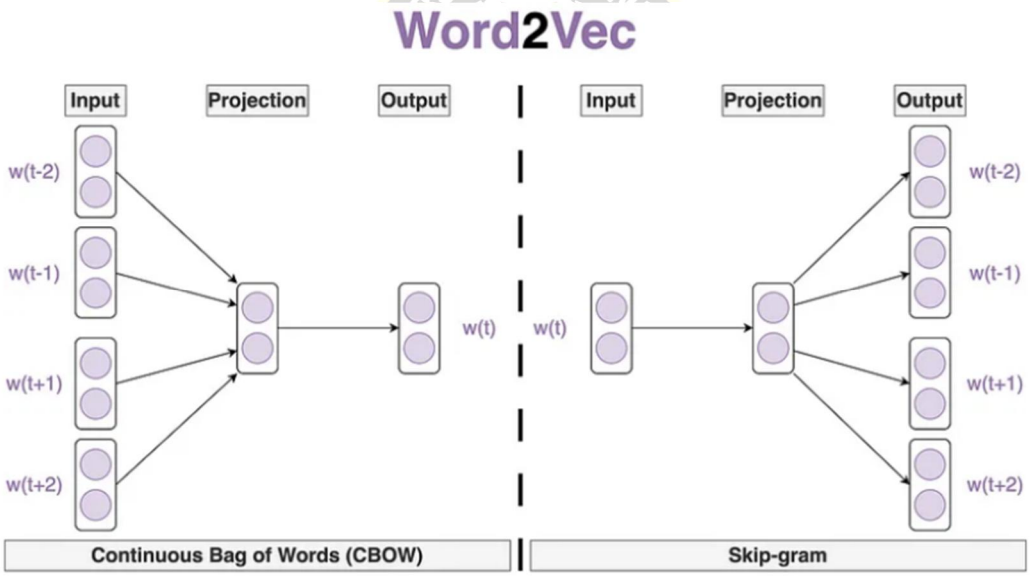
Dalam *neural network*, keseragaman panjang input sangat penting, tetapi variasi panjang teks dapat menjadi tantangan. Teknik *padding* digunakan untuk menambahkan token nol pada teks yang lebih pendek, sehingga semua teks memiliki panjang yang seragam. Ini meningkatkan efisiensi pemrosesan dengan menyamakan dimensi input, memungkinkan model beroperasi lebih optimal dan meningkatkan kinerja secara keseluruhan (Giménez dkk., 2020).

Tabel 2.3. Penelitian Terdahulu Definisi *Word Embedding*

No.	Peneliti	Judul	Definisi
1.	Mikolov dkk. (2013)	*Efficient Estimation of Word Representations in Vector Space*	*Word Embedding* adalah metode dalam NLP dan machine learning yang merepresentasikan kata atau frasa sebagai vektor dalam ruang kontinu. Teknik ini membuat kata-kata dengan makna serupa memiliki vektor yang berdekatan, memungkinkan komputer untuk lebih memahami dan menganalisis teks, serta meningkatkan kualitas NLP.
2.	Pennington dkk., (2014)	*GloVe : Global Vectors for Word Representation*	*Word Embedding* merepresentasikan kata sebagai vektor berdimensi tinggi untuk menangkap hubungan antar kata, mempermudah pemahaman komputer pada makna kata dalam NLP dan *machine learning*.
3.	Lample dkk., (2017)	*Unsupervised Machine Translation Using Monolingual Corpora*	*Word Embedding* adalah metode mengonversi kata ke vektor numerik dalam ruang kontinu, membantu model *machine learning* memahami bahasa, menangkap hubungan antar kata, dan meningkatkan kinerja pada tugas NLP dan *machine learning*.
4.	Devlin dkk. (2018)	*BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*	*Word Embedding* adalah teknik yang mengubah kata menjadi vektor numerik, sehingga kata dengan makna serupa memiliki vektor yang sama. Teknik ini memungkinkan komputer memahami dan memanipulasi makna kata, serta mengenali relasi semantis antar kata.
5.	Cer dkk. (2018)	*Universal Sentence Encoder*	*Word Embedding* adalah metode NLP yang mengubah kata menjadi vektor padat untuk menangkap makna dan konteks. Teknik ini memungkinkan evaluasi kesamaan semantis antar kata, sehingga kita bisa memahami, memanipulasi, dan mengidentifikasi hubungan makna kata dalam teks

Dari beberapa definisi dalam Tabel 2.3 dapat disimpulkan bahwa *word embedding* merupakan teknik dalam NLP dan *machine learning* yang merepresentasikan kata sebagai vektor dalam ruang kontinu untuk menangkap makna dan konteks. Teknik ini membuat kata dengan makna serupa memiliki vektor yang berdekatan, memudahkan komputer memahami, menganalisis, dan memanipulasi makna kata. *Word Embedding* memungkinkan evaluasi kesamaan semantis dan pengenalan hubungan antar kata, meningkatkan akurasi dan relevansi dalam NLP dan ML.

Model *Word2Vec* adalah metode word embedding dalam NLP yang mengubah kata menjadi vektor numerik untuk menangkap makna sintaksis dan semantis. Dengan membaca teks besar, *Word2Vec* mengelompokkan kata dengan makna serupa dalam vektor yang sama, memperbaiki pemahaman bahasa alami, dan meningkatkan akurasi dalam aplikasi NLP (Al-Amin dkk., 2017). *Word2Vec* menggunakan *neural network* untuk belajar representasi vektor kata dalam konteks yang lebih luas, menangkap makna semantik. Model ini memproses data teks besar dengan cepat, menghasilkan vektor yang berguna untuk aplikasi NLP seperti analisis sentimen dan klasifikasi teks. Terdapat dua arsitektur: skip-gram, yang memprediksi kata-kata sekitar, dan Continuous Bag of Words (CBOW), yang memprediksi kata saat ini berdasarkan konteksnya. Kedua model ini meningkatkan pemahaman hubungan semantis antar kata. Arsitektur model *Word2vec skip-gram* dan CBOW ditunjukkan pada Gambar 2.1 (Nawangsari dkk., 2019).



Gambar 2.1. Arsitektur Model *Word2Vec*

Tabel 2.4. Penelitian Terdahulu Definisi *Topic Modeling* LDA

No.	Peneliti	Judul	Definisi
1.	D. Blei dkk., (2001)	*Latent Dirichlet Allocation*	LDA mengidentifikasi topik tersembunyi dalam teks melalui probabilitas kata-kata, memungkinkan pemahaman pola, hubungan, dan topik utama seperti politik, ekonomi, olahraga, dan teknologi. Analisis probabilitas kata memfasilitasi pemahaman tren dan peristiwa dalam teks dengan sistematis.
2.	Rishel dkk., (2007)	*Determining the context of text using augmented latent semantic indexing*	LDA adalah metode statistik untuk mengelompokkan teks ke dalam topik-topik tersembunyi, sering digunakan untuk analisis teks seperti pengelompokan dokumen dan peringkasan teks.
3.	D. M. Blei, (2012)	*Probabilistic topic models*	LDA menggambarkan teks sebagai kombinasi topik tersembunyi dengan distribusi probabilitas kata-kata, memungkinkan identifikasi subjek, analisis tematik dokumen, dan prediksi kata-kata secara probabilistik dalam konteks tertentu.
4.	Ding dkk., (2019)	*Semantic Correlation Promoted Shape-Variant Context for Segmentation*	LDA adalah metode untuk menemukan topik-topik tersembunyi dalam kumpulan dokumen dengan menganalisis distribusi kata.
5.	Huang dkk., (2019)	*Dynamic Context Correspondence Network for Semantic Alignment*	LDA adalah metode statistik yang digunakan untuk pemodelan topik, di mana model ini mengidentifikasi topik-topik tersembunyi dalam data teks dengan menganalisis distribusi kata-kata di dalam dokumen.
6.	Lee dkk., (2021)	*Semi-Global Context Network for Semantic Correspondence*	LDA adalah metode untuk mengidentifikasi topik tersembunyi dalam dokumen teks dengan menganalisis distribusi kata, membantu memahami struktur topik dalam data teks besar.
7.	Sassi dkk., (2022)	*LEOnto+: A scalable ontology enrichment approach*	LDA adalah adalah teknik statistik yang digunakan untuk menemukan topik-topik tersembunyi dalam kumpulan dokumen teks dengan menganalisis distribusi kata di dokumen untuk mengidentifikasi pola dan struktur topik yang mendasari data teks.

Dari beberapa definisi pada tabel 2.4 dapat disimpulkan bahwa LDA adalah metode statistik untuk menemukan topik tersembunyi dalam teks dengan menganalisis distribusi kata. Ini membantu memahami pola dan hubungan dalam data teks, serta digunakan untuk pengelompokan dan peringkasan dokumen. LDA menggambarkan teks sebagai campuran topik tersembunyi dan probabilitas kata, memungkinkan identifikasi topik dan analisis tematik secara efisien.

Cara menghitung probabilitas kata (w) dalam topik (k) menggunakan model LDA (D. Blei dkk., 2001) dengan persamaan sebagai berikut:

$$P(w|k) = \frac{(n_{kw}+\beta)}{(\sum(n_k)+ V * \beta)} \quad (2.1)$$

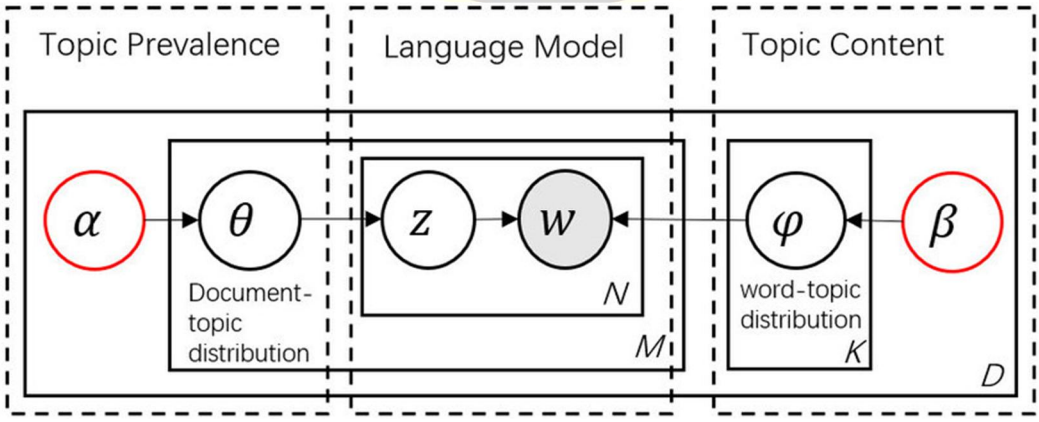
Di mana P(w|k) adalah probabilitas kata w muncul dalam topik k. n_kw menunjukkan frekuensi kemunculan kata w dalam topik k. β adalah *parameter multinomial* untuk setiap kata dalam setiap dokumen. n_k adalah total kata dalam topik k dan V adalah total kata dalam seluruh dokumen.

Cara menghitung distribusi topik dalam dokumen menggunakan model LDA dengan persamaan distribusi topik z dalam dokumen d (Blei dkk., 2003) adalah sebagai berikut:

$$P(z|d) = \frac{(n_{dz}+ \alpha)}{(\sum(n_d)+ K * \alpha)} \quad (2.2)$$

Di mana probabilitas bahwa topik z akan muncul dalam dokumen d diberikan oleh P(z | d). Jumlah n_dz menunjukkan seberapa sering topik z disebutkan dalam dokumen d, dan α adalah *parameter dirichlet* untuk setiap dokumen. Jumlah total kata dalam dokumen d secara keseluruhan adalah n_d. Di mana K menyatakan jumlah topik yang telah ditentukan.

Diagram blok dasar digunakan untuk menjelaskan metode LDA dalam melakukan pemodelan topik penelitian, seperti ditunjukkan pada Gambar 2.2.



Gambar 2.2. Diagram Blok Dasar LDA

Tabel 2.5. Penelitian Terdahulu Definisi *Coherence Score*

No.	Peneliti	Judul	Definisi
1.	Röder dkk. (2015)	*Exploring the space of topic coherence measures*	*Coherence score* adalah metrik yang mengevaluasi konsistensi semantik antar kata dalam satu topik.
2.	Lau & Baldwin (2016)	*The sensitivity of topic coherence evaluation to topic cardinality*	*Coherence score* digunakan untuk mengukur kualitas topik yang dihasilkan oleh model topik, dengan fokus pada konsistensi semantik antar kata dalam topik.
3.	Syed & Spruit (2017)	*Full-text or abstract? Examining topic coherence scores using latent dirichlet allocation*	*Coherence score* digunakan untuk mengevaluasi kualitas topik yang dihasilkan oleh model topik seperti LDA.

Dari beberapa definisi dalam Tabel 2.5 dapat disimpulkan bahwa *coherence score* adalah metrik yang mengevaluasi dan mengukur kualitas topik berdasarkan konsistensi semantik antar kata dalam topik, terutama digunakan dalam model topik seperti LDA. Perhitungan *coherence* model LDA $C(T)$ yaitu rata-rata kesepadanan dari semua topik dan dapat diungkapkan sebagai berikut:

$$C(T) = \frac{1}{K}\sum_{k=1}^{K} C(T_k) \quad \text{.....(2.3)}$$

Rumus tersebut digunakan untuk menghitung menghitung *coherence* model LDA $C(T)$. Ini merupakan rata-rata kesepadanan dari setiap topik T_k dalam model, di mana K adalah jumlah total topik dalam model. Perhitungan *coherence* topik $C(T_k)$ yaitu rata-rata kesepadanan dari semua pasangan kata dalam suatu topik T_k, sebegai berikut :

$$C(T_k) = \frac{1}{|W_k|(|W_k|-1)}\sum_{i=1}^{|W_k|}\sum_{j=i+1}^{|W_k|} C(w_{ki}, w_{kj}) \quad \text{.....(2.4)}$$

Rumus tersebut digunakan untuk menghitung kesepadanan topik T_k. Ini merupakan rata-rata kesepadanan dari semua pasangan kata unik dalam topik T_k, dimana setiap pasangan kata dihitung hanya sekali, menghindari perhitungan ulang untuk pasangan kata yang sama namun berurutan. Perhitungan coherence dari pasangan kata $C(w_1, w_2)$, sebagai berikut :

$$C(w_1, w_2) = \log\frac{P(w_1,w_2)}{P(w_1)\cdot P(w_2)} \quad \text{.....(2.5)}$$

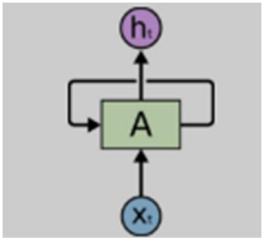
Rumus tersebut digunakan untuk menghitung kesepadanan antara pasangan kata w_1 dan w_2 . Hal ini untuk mengevaluasi seberapa sering kata-kata tersebut muncul bersama dalam suatu topik dibandingkan dengan kehadiran mereka secara

individual. Semakin besar nilai *coherence* (C), semakin kuat keterkaitan antara kata-kata tersebut dalam konteks topik tersebut.

Tabel 2.6. Penelitian Terdahulu Definisi *Deep Learning*

No.	Peneliti	Judul	Definisi
1.	Russakovsky dkk. (2015)	*ImageNet Large Scale Visual Recognition Challenge*	*Deep learning* menggunakan jaringan saraf berlapis untuk mempelajari data secara hierarkis, memungkinkan model kompleks untuk tugas seperti klasifikasi gambar, deteksi objek, dan penerjemahan bahasa, serta meningkatkan kemampuan pemrosesan informasi dan kemajuan *artificial intelligence.*
2.	LeCun dkk. (2015)	*Deep learning*	*Deep learning* memanfaatkan jaringan saraf berlapis dan *dataset* besar untuk memahami data secara hierarkis, sangat penting dalam pengenalan gambar, pemrosesan bahasa, dan pembelajaran penguatan, serta mendukung kemajuan *artificial intelligence.*

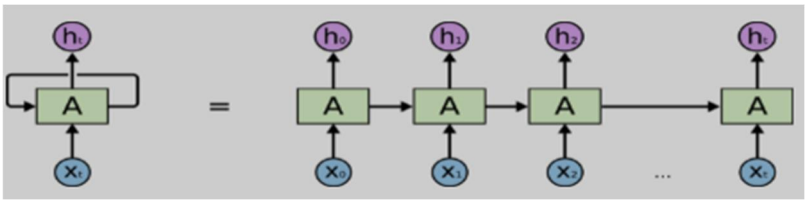
Dari beberapa definisi dalam Tabel 2.6 dapat disimpulkan bahwa *deep learning* menggunakan jaringan saraf berlapis untuk memahami data secara hierarkis, mendukung model canggih dalam klasifikasi gambar, deteksi objek, dan penerjemahan bahasa, serta kemajuan kecerdasan buatan. RNN, yang juga bagian dari deep learning, memproses urutan data dengan memori internal berkat koneksi siklik, memungkinkan pemahaman konteks temporal dan unggul dalam tugas seperti pengenalan ucapan dan prediksi urutan dibandingkan dengan jenis jaringan saraf lainnya (Kurniadi & Maliki, 2018). Gambar 2.3. menunjukkan proses perulangan informasi dalam RNN (Smagulova & James, 2019).



Gambar 2.3. Proses perulangan informasi RNN

Xt untuk input, ht untuk output, dan ada jalur loop yang memungkinkan data diteruskan dari satu langkah jaringan ke langkah berikutnya. RNN dapat dianggap sebagai banyak salinan dari jaringan yang sama. Setiap jaringan mengirim pesan ke

jaringan berikutnya seperti yang ditunjukkan pada Gambar 2.4 (Smagulova & James, 2019).



Gambar 2.4. Multi Salinan Jaringan RNN

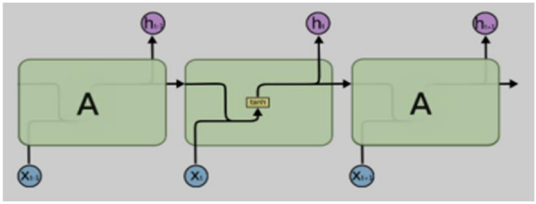
Tabel 2.7. Penelitian Terdahulu Definisi *Sentiment Analysis*

No.	Peneliti	Judul	Definisi
1.	B. Liu, (2014)	*Sentiment Analysis: Mining Opinions, Sentiments, and Emotions*	*Sentiment Analysis* menggunakan klasifikasi data untuk membagi teks ke dalam kategori yang sudah ada, dengan fokus pada mengidentifikasi polaritas (positif, negatif, atau netral). Ini dapat diterapkan pada tiga tingkat (dokumen, kalimat, aspek) secara terpisah atau digabungkan, tergantung pada kebutuhan analisis.
2.	Beigi dkk. (2016)	*An Overview of Sentiment Analysis in Social Media and Its Applications in Disaster Relief*	*Sentiment Analysis* merupakan metode evaluasi perasaan dan pendapat dalam teks media sosial, yang menitikberatkan pada sikap (positif, negatif, atau netral) pengguna terhadap topik spesifik. Tujuannya adalah untuk memahami emosi dan opini yang terungkap dalam konteks sosial.
3.	Singh & Sharma (2023)	*Sentiment Analysis Study of Human Thoughts using Machine Learning Techniques*	*Sentiment Analysis* adalah teknik untuk mengidentifikasi dan mengklasifikasikan opini atau perasaan dalam teks menggunakan metode terbaru dalam *machine learning* dan *deep learning*.
4.	Sande dkk. (2024)	*Social Media Sentiment Analysis: A Comprehensive Analysis*	*Sentiment Analysis* adalah teknik untuk menilai dan mengklasifikasikan sentimen dalam teks menggunakan berbagai metode, termasuk teknik *lexicon based* dan *machine learning*.

Dari beberapa definisi dalam Tabel 2.7 dapat disimpulkan bahwa *sentiment analysis* adalah teknik untuk mengidentifikasi dan mengklasifikasikan sentimen dalam teks menggunakan *lexicon based* dan *machine learning*, diterapkan pada berbagai tingkat teks untuk memahami emosi dan opini dalam konteks sosial dan media.

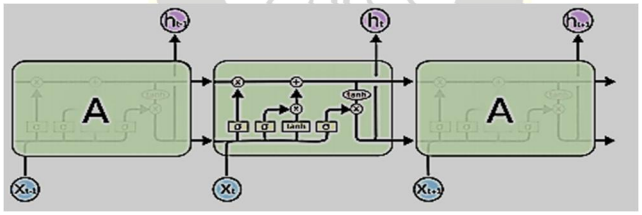
LSTM dalam sentiment analysis digunakan untuk mengidentifikasi perasaan dalam teks, terutama di media sosial, dengan efektif mengenali sikap dan

emosi. Dengan *memory cells* dan *gate*, LSTM unggul dalam prediksi deret waktu, NLP, dan pemodelan urutan (Hochreiter, 1998). Jaringan perulangan RNN hanya menggunakan satu *layer* sederhana, yaitu *layer tanh*, seperti yang ditunjukkan pada Gambar 2.5 (Smagulova & James, 2019).



Gambar 2.5. Layer tanh pada RNN

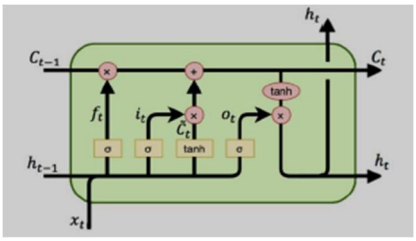
Gambar 2.6 ini menunjukkan struktur dasar model LSTM, di mana setiap unit menerima input sekuensial x_t dan berfungsi sebagai blok memori. Unit LSTM terdiri dari tiga gerbang : *forget gate* untuk melupakan informasi, *input gate* untuk menyimpan informasi baru, dan *output gate* untuk mengeluarkan informasi. Status memori h_t menyimpan informasi dari langkah sebelumnya dan mempengaruhi output ke langkah berikutnya. Gambar ini juga menggambarkan koneksi berurutan antara unit LSTM untuk memproses data sekuensial (Smagulova & James, 2019).



Gambar 2.6. Perulangan empat *layer* LSTM

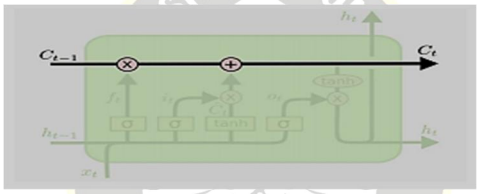
Gambar 2.7 menggambarkan mekanisme internal unit LSTM yang mengelola informasi dalam jaringan saraf. Input sekuensial x_t masuk ke unit, sementara status memori sebelumnya C_{t-1} digunakan untuk pembaruan. Terdapat tiga gerbang: *forget gate* (f_t) yang memutuskan informasi mana yang akan dilupakan, *input gate* (i_t) yang mengontrol informasi baru yang akan ditambahkan, dan *output gate* (o_t) yang menentukan informasi yang akan dikeluarkan sebagai *output* h_t. Calon status memori C_t dihitung dari input saat ini dan informasi sebelumnya, memungkinkan

model untuk menangkap dependensi jangka panjang dalam data sekuensial (Smagulova & James, 2019).



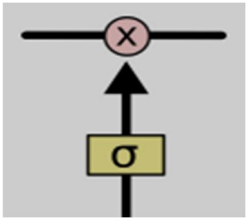
Gambar 2.7. Ilustrasi sel LSTM

Panah hitam menunjukkan aliran informasi di dalam, antar, dan keluar dari sel LSTM. Sel LSTM menghasilkan dua keluaran: *hidden state* (ht) untuk input ke sel berikutnya, dan *cell state* (Ct), yang merupakan kunci utama LSTM. *Cell State* adalah garis horizontal yang menghubungkan semua lapisan keluaran LSTM, seperti yang ditunjukkan pada Gambar 2.8 (Smagulova & James, 2019).



Gambar 2.8. *Cell State*

LSTM memiliki cara menambah dan menghapus data dari *cell state* yang disebut *gate*. Fungsi *gate* secara selektif memeriksa data yang masuk menggunakan *sigmoid layer* yang ditunjukkan pada Gambar 2.9. *Output* dari *sigmoid layer* menunjukkan data yang ditransmisikan atau ditangguhkan. Angka 0 untuk berhenti dan angka 1 untuk melanjutkan (Smagulova & James, 2019).



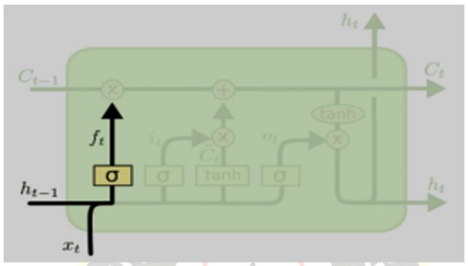
Gambar 2.9. *Sigmoid Layer*

Forget gate (ft), *input gate* (it), dan *output gate* (ot) adalah pintu utama dalam LSTM. Forget gate menghapus data dari memori, input gate memperbarui nilai

masukan, dan output gate mengontrol keluaran berdasarkan masukan dan memori (Smagulova & James, 2019).

Pendekatan LSTM terdiri dari empat fase utama yang melibatkan pintu-pintu tersebut adalah sebagai berikut :

1) Fase pertama adalah *forget phase*. *Forget gate* menentukan apakah informasi perlu dihapus dari *memory cells* dengan bantuan lapisan *sigmoid* yang disebut *forget gate layer*. Ini didasari oleh ht-1 dan Xt untuk memproduksi output 0 atau 1 dalam sel sebelumnya Ct-1, seperti yang ditunjukkan dalam Gambar 2.10.



Gambar 2.10. *Forget gate layer*

Persamaan forget gate diuraikan pada persamaan berikut :

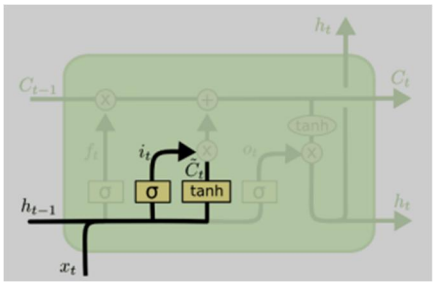
$$ft = \sigma(Wf.[ht-1, xt] + bf \quad \text{.....(2.6)}$$

Dimana ft adalah *forget gate*, σ merupakan fungsi *sigmoid*, Wf adalah nilai *weight* untuk *forget gate*, $ht-1$ mewakili output sebelum urutan ke t, Xt adalah input pada urutan ke t, dan bf adalah bias untuk *forget gate*. Nilai weight diuraikan pada persamaan berikut :

$$W = \left(-1\ \sqrt{d}\ 1\ \sqrt{d}\right) \quad \text{.....(2.7)}$$

Dimana W adalah *weight* dan d adalah jumlah data.

2) Fase kedua adalah *input phase*. Nilai masukan yang akan disimpan dalam *memory cells* diputuskan. Ini terdiri dari dua bagian: *input gate layer* menentukan nilai yang diperbarui, sementara lapisan *tanh* menciptakan nilai kandidat baru ($\overline{Ct}$) untuk ditambahkan ke *cell state*. Output dari kedua lapisan ini digabungkan untuk memperbarui *cell state* sesuai dengan Gambar 2.11.



Gambar 2.11. *Input gate layer* dan *tanh layer*

Persamaan *input gate* diuraikan pada persamaan berikut :

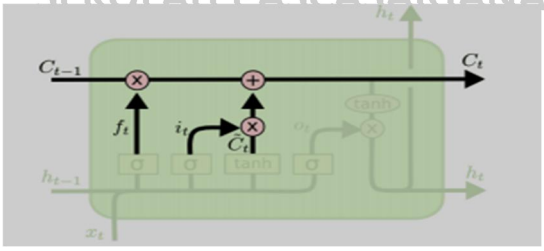
$$it = \sigma(Wi.[ht-1, Xt] + bi) \quad \text{.....(2.8)}$$

Dimana it adalah *input gate*, σ adalah fungsi *sigmoid*, Wi adalah nilai *weight* untuk *input gate*, $ht-1$ adalah nilai *output* sebelum orde ke t, Xt adalah nilai *input* pada orde ke t dan bi adalah nilai *bias* pada *input gate*. Persamaan kandidat baru diuraikan pada persamaan berikut :

$$C\bar{t} = \tanh(WC.[ht-1, Xt] + bC) \quad \text{.....(2.9)}$$

Dimana $C\bar{t}$ adalah nilai baru yang dapat ditambahkan ke *cell state*, $tanh$ adalah fungsi *tanh*, WC adalah nilai *weight* untuk *cell state, ht*−1 adalah nilai *output* sebelum orde ke t, Xt adalah nilai *input* pada orde ke t dan bC adalah nilai *bias* untuk *cell state.*

3) Fase ketiga adalah *update phase*, di mana informasi baru diintegrasikan ke dalam *memory cells*. Tujuannya adalah memperbarui *cell state* lama (Ct-1) ke *cell state* baru (Ct) seperti yang ditunjukkan pada Gambar 2.12.



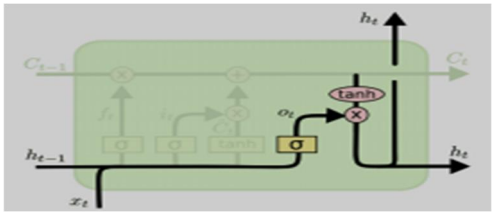
Gambar 2.12. Pembuatan *cell state* baru

Kemudian mengalikan mode lama dengan ft untuk menghapus data yang ditentukan sebelumnya di langkah *forget gate layer*. Selanjutnya ditambahkan ke dalamnya it * Ct, yang merupakan nilai baru dan digunakan untuk memperbarui *state*. Persamaan *cell state* diuraikan pada persamaan berikut:

$$Ct = ft * Ct - 1 + it * C\bar{t} \quad \text{.....(2.10)}$$

Dimana Ct adalah *cell state,* ft adalah *forget gate,* $Ct-1$ adalah *cell state* sebelum orde ke t, it adalah *input gate* dan $C\bar{t}$ adalah nilai baru yang dapat ditambahkan ke *cell state*

4) Fase keempat adalah *output phase*, di mana pintu keluaran mengendalikan keluaran berdasarkan informasi dari sel memori. Fase ini merupakan langkah terakhir dari metode LSTM, tujuannya adalah untuk memutuskan hasil *output*, seperti yang ditunjukkan pada Gambar 2.13.



Gambar 2.13. Penentuan *output*

Output LSTM harus disesuaikan dengan *cell state* yang telah diproses sebelumnya melalui dua langkah utama berikut ini :

a) Pertama, *sigmoid layer* berperan dalam menentukan kontribusi mana dari *cell state* yang akan mempengaruhi *output*. Ini menghasilkan koefisien *sigmoid* untuk mengontrol seberapa besar setiap elemen dari *cell state* akan mempengaruhi pembentukan *output*, dengan tujuan memilih informasi yang relevan untuk dimasukkan ke dalam *output*.

b) Kedua, *cell state* diteruskan ke *tanh layer* yang mengubah nilai *cell state* ke rentang -1 hingga 1, memastikan *output* sesuai dengan batasan tersebut. *Output* dari *tanh layer* kemudian dikalikan dengan koefisien *sigmoid* sebelumnya, memastikan *output* akhir sesuai dengan preferensi dan penyesuaian yang telah ditetapkan.

Dengan demikian, kombinasi lapisan *sigmoid* dan *tanh layer* dalam LSTM berperan penting dalam menghasilkan output yang tepat berdasarkan keadaan sel yang telah diproses sebelumnya. Persamaan *output gate* diuraikan pada persamaan sebagai berikut :

$$ot = \sigma(Wo.[ht-1, xt] + bo) \quad \text{.....(2.11)}$$

Dimana ot adalah *output gate*, σ adalah fungsi *sigmoid*, Wo adalah nilai *weight* untuk *output gate*, $ht-1$ adalah nilai *output* sebelum orde ke t, Xt adalah nilai

input pada orde ke t dan bo adalah nilai *bias* pada *output gate*. Persamaan nilai *output* order ke t diuraikan pada persamaan berikut :

$$ht = ot * \tanh(Ct) \quad (2.12)$$

Dimana ht adalah nilai *output* orde ke t, ot adalah *output gate,* $tanh$ adalah fungsi tanh dan Ct adalah *cell state.*

Akurasi model dapat dievaluasi dengan menggunakan confusion matrix, yang merupakan alat analisis yang menyajikan data tentang jumlah prediksi yang benar dan salah untuk setiap kategori kelas, yaitu negatif, netral, dan positif. Dalam confusion matrix, setiap elemen menunjukkan jumlah prediksi yang dibuat oleh model dan membagi hasilnya menjadi empat kategori: true positives (TP), false positives (FP), true negatives (TN), dan false negatives (FN). Dengan demikian, confusion matrix memungkinkan kita untuk melihat dengan jelas seberapa baik model dalam mengklasifikasikan input ke dalam kelas yang tepat. Selain itu, analisis terhadap confusion matrix dapat membantu mengidentifikasi pola kesalahan yang terjadi, sehingga memberikan wawasan tentang area di mana model mungkin perlu diperbaiki untuk meningkatkan kinerjanya. Akurasi model menggunakan confusion matrix dapat dihitung dengan menggunakan rumus :

$$Akurasi = \frac{Jumlah\ Prediksi\ Benar}{Total\ Sampel} \quad (2.13)$$