BAB II
TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Penelitian sebelumnya telah banyak menggunakan arsitektur LSTM untuk keperluan analisis sentimen. Namun, akurasi yang didapat sangat rendah karena LSTM hanya mampu menerima konteks makna dari kata sebelumnya, untuk menangani masalah tersebut terciptalah arsitektur Bi-LSTM yang mampu menangkap konteks dari kiri ke kanan ataupun dari kanan ke kiri sehingga performa yang dihasilkan lebih baik dari LSTM. Penelitian sebelumnya [8] membandingkan kinerja BIGRU dan BiLSTM untuk klasifikasi sentimen pengguna. Sumber data untuk training dan testing berasal dari gabungan komentar di twitter pengguna JNT, JNE & Anteraja. Total data bersih 8.261 yang dibagi menjadi data testing 1.217 dan sisanya data training. Kinerja BiLSTM sebesar 71.5%, sedangkan BiGRU 66.5%. Penelitian ini menunjukkan BiLSTM memiliki kinerja yang lebih baik daripada BIGRU. Namun perkembangan terkini sudah ada anotasi model *pretrained* IndoBERT [4].

Model *pretrained* IndoBERT dibandingkan akurasinya dengan KNN, SVM, dan Naive Bayes. Dataset yang digunakan merupakan dataset komentar berjumlah 3184 baris dengan 3 label yaitu not abusive 2789 baris, abusive not offensive 110 baris, dan abusive and offensive 285 baris dengan perbandingan proporsi data latih 60% dan data tes 40%. Data didapatkan dari berita yang sedang *trend* pada bulan Maret - September 2019. Hasil akurasi *pretrained* IndoBERT mampu mengungguli model lain dengan akurasi 76.32% meskipun tanpa melakukan *fine tuning.*

Didukung penelitian [17] tentang penggunaan *pretrained* IndoBERT tanpa melakukan *fine tuning* dan komparasinya terhadap model CNN-LSTM dan BERT Base. Data yang dipakai berasal dari IndoLEM dan *crawling* data dari twitter sebanyak 536 komentar dengan data yang seimbang

seimbang yang berlabel positif dan negatif sert menggunakan 50% data latih dan dan 50% data uji. Tujuan penelitian ini adalah untuk mengetahui perbandingan performa 3 model, BERT Base, IndoBERT, dan CNN-LSTM. Meskipun hanya menggunakan *pre trained* IndoBERT ternyata, IndoBERT memperoleh akurasi yang paling tinggi sebesar 80%.

Dilanjutkan dengan penelitian [3] yang juga menggunakan *pre trained* model Indo-BERT untuk dibandingkan dengan model *Support Vector Machine* pada dataset yang *imbalance* sejumlah 1005 baris label negatif dan 1619 baris label positif atau jika dalam perbandingan rasio 38.3% label negatif berbanding 61.7% label positif. Penelitian ini bertujuan untuk mengetahui perbandingan *pretrained* IndoBERT dengan model *Support Vector Machine* (SVM) pada dataset yang tidak seimbang. Hasilnya *pretrained* IndoBERT menghasilkan performa yang lebih baik dengan akurasi 86%.

Penelitian selanjutnya [9] mengenai model arsitektur BiLSTM pada dataset imbalanced sebanyak 5000 baris ulasan aplikasi grab pada aplikasi google play store dengan rincian 1615 baris label negatif, 272 netral, dan 3113 baris label positif. Perbandingan data latih dan data uji sebesar 80% berbanding 20%. Penelitian ini bertujuan untuk membandingkan BiLSTM dengan LSTM, *MultinomialNB, Logistic Regression,* dan SVC. Hasil akurasi BiLSTM mampu mengungguli model lain dengan akurasi yang cukup tinggi yaitu 91%.

Semakin banyaknya penggunaan BiLSTM untuk analisis sentimen membuat para peneliti mencoba bereksperimen untuk menambahkan arsitektur *layer* tambahan yaitu TF-IDF . Penelitian [10] menggunakan arsitektur BiLSTM dengan *layer* tambahan TF-IDF pada dataset ulasan hotel sebanyak 15.000 baris dengan perbandingan rasio 4:1 untuk data latih dan data ujinya. Penelitian ini bertujuan untuk membandingkan kinerja BiLSTM TF-IDF dengan RNN, CNN, LSTM, dan Naive Bayes. Akurasi model BiLSTM mampu unggul dari model-model lain sebesar 91.54%. Hasil ini sudah cukup tinggi dibanding model BiLSTM tanpa tambahan

layer TF-IDF.

Penelitian selanjutnya [5] melihat bahwa beberapa penelitian sebelumnya tidak menerapkan *fine tuning* dan *hyperparameter tuning* pada model IndoBERT. Penelitian mencoba untuk membandingkan performa *fine tuning* IndoBERT dengan model BERT *Multilingual* pada dataset ulasan PUBG sebanyak 15.000 data dengan 3 macam label yaitu positif, negatif, dan netral. Proporsi pembagian data sebesar 80% data uji, 10% data latih, 10% data validasi, dari penelitian ini ternyata IndoBERT yang telah di-*fine tuning* mampu menghasilkan performa akurasi yang lebih baik yaitu sebesar 94% dengan *learning rate* 0.00002, *batch size* 32, serta 5 *epoch.*

Didukung penelitian berjudul [7] yang membahas mengenai *pretrained* IndoBERT dan *pretrained* IndoBERT RCNN pada dataset IndoNLU. Sumber dataset berasal dari berbagai *platform* sosial media antara lain, twiiter, zomato, tripadvisor, facebook, instagram, dan qraved. Dataset berisikan 11.000 data training, 1260 data validasi dan 500 data uji. Tujuan penelitian ini yaitu untuk membandingkan kinerja *pretrained* IndoBERT RCNN dengan model *pretrained* IndoBERT. Akurasi yang didapatkan *pretrained* IndoBERT RCNN mampu mengungguli *pretrained* IndoBERT sebesar 95.16% berbanding 93.27%.

Pada penelitian [11] memaparkan mengenai konsep BiLSTM dengan *tuning hyperparameter* serta tambahan *layer* CBOW Word2Vec sebagai vektorisasinya. Sumber data berasal dari teks berbahasa Inggris dengan label *fake* dan *real.* Penelitian ini bertujuan untuk mengetahui kinerja BiLSTM pada klasifikasi berita. Didapatkan akurasi sebesar 79.19% untuk data judul dan 92.8% untuk konten model.

Penelitian selanjutnya [12] membahas tentang BiLSTM dengan tambahan layer Word2Vec yang dikomparasi dengan CNN pada dataset yang berjumlah 58.984 baris ulasan dengan 3 label yaitu positif, negatif dan netral. Data latih yang digunakan 67% dan 33% data digunakan untuk data uji. Hasil akurasi terbaik yaitu BiLSTM dengan tambahan layer Word2Vec sebesar 69.48% pada analisis sentimen dan 84.36% pada analisis emosi.

Tabel 2. 1 Penelitian terdahulu

Literature Review		**Latar Belakang Penelitian**		**Desain Riset dan Metodologi**		
Penulis, Tahun	**Judul**	**Masalah Penelitian/ Rumusan Masalah**	**Tujuan**	**Metode**	**Data**	**Hasil / Temuan/ Kesimpulan**
Guixian Xu, Yueting Meng, Xiaoyu Qiu, Ziheng Yu, Xu Wui, 2019	*Sentiment Analysis of Comment Texts Based on BiLSTM*	Banyak sekali komentar di internet yang memiliki tendensi emosi positif atau negatif, maka dari itu diperlukan analisis komentar untuk mengetahui konteks komentar negatif atau positif menggunakan BiLSTM yang	Mengetahui hasil metode BiLSTM dengan pembobotan TF-IDF.	BiLSTM dengan pembobotan TF-IDF serta RNN, CNN, LSTM dan Naive Bayes sebagai pembandingnya.	Dataset berasal dari 15.000 komentar hotel pada situs Ctrip.com	Hasil metode BiLSTM dengan TF-IDF mengungguli metode yang lain dengan akurasi sebesar 91.54%, recall 92.82% dan f1-score 92.18%, dibandingkan model yang

		dikombinasikan dengan pembobotan TF-IDF.				lain yang hanya berkisar antara 84% - 89%
Salsabila Zahirah Pranida, Arrie Kurniawardhani, 2022	*Sentiment Analysis of Expedition Customer Satisfaction using BiGRU and BiLSTM*	Selama pandemi jasa ekspedisi seperti JNE, JNT & Anteraja semakin meningkat, untuk mencari kepuasan pelanggan, dilakukan analisis sentimen pada ulasan pengguna aplikasi twitter menggunakan menggunakan metode BiGRU & Bi-LSTM.	Membandingkan hasil metode antara BiGRU dan BiLSTM pada ulasan pengguna twitter mengenai 3 jasa ekspedisi yang berbeda	BiGRU dan BiLSTM.	Dataset sebanyak 1.217 baris hasil crawling dari twitter.	Bi-GRU mampu mengungguli Bi-LSTM dengan akurasi 71.5% berbanding 66.5%.

Junita Amalia, Juanda Pakpahan, Melani Pakpahan, Yeni Panjaitan, 2022	Model Klasifikasi Berita Palsu Menggunakan *Bidirectional* LSTM dan Word2Vec sebagai Vektorisasi	Pengklasifikasian berita fakta dan berita palsu menggunakan metode Bi-LSTM dengan tambahan *layer* CBOW pada *Word2Vec* sebagai vektorisasi kata.	Mengetahui kinerja Bi-LSTM dan Word2Vec sebagai vektorisasi pada klasifikasi berita.	Hyperparameter tuning pada embedding layer Bi-LSTM ditambah dengan CBOW pada *Word2Vec*	Dataset dari portal berita.	Akurasi sebesar 79.19% untuk data judul dan 92.8% untuk konten model.
Dloifur Rohman Alghifari, Mohammad Edi, Lutfi Firmansyah, 2022	Implementasi *Bidirectional* LSTM untuk Analisis Sentimen Terhadap Layanan Grab Indonesia	*Analisis sentiment* terhadap layanan grab di Indonesia dengan metode Bi-LSTM	Mengetahui kinerja Bi-LSTM pada sentimen analisis layanan grab di Indonesia	Menggunakan model Bi-LSTM, serta model LSTM, MultinomialNB , Logistic Regression , SVC, sebagai perbandingannya .	Dataset berasal dari *google play store* sebanyak 5000 data dengan 1615 label negatif, 272 netral, dan 3113 positif.	Akurasi Bi-LSTM mampu mengungguli model lain dengan capain 91% dibanding LSTM, Multinomial NB, Logistic Regression, dan SVC dengan akurasi

						berturut-turut sebesar 76%, 63%, 67%, 65%.
Mohamad Romli Firdaus Kamarula, Naim 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	Kebijakan pemerintah dalam menangani pandemi Covid-19 menuai berbagai macam tanggapan publik, terutama pada media sosial twitter. Diperlukan analisis sentimen menggunakan metode CNN dan Bi-LSTM untuk mengetahui respon sentimen dan emosi dari masyarakat.	Membandingkan akurasi CNN dan Bi-LSTM untuk analisis sentimen dan emosi menggunakan metode Word2Vec.	Algoritma CNN dan Bi-LSTM serta Word2Vec untuk analisis emosi.	Dataset berjumlah 58.984 baris dengan rincian 22.062 data dari kaggle dan 36.922 dari twitter.	Bi-LSTM mampu mengungguli CNN dengan akurasi 69.48% berbanding 68.58% pada analisis sentimen serta 84.36% berbanding 84.21% pada analisis emosi.

Hanvito Michael Lee, Yuliant Sibaroni, 2023	*Comparison of IndoBERTweet and Support Vector Machine on Sentiment Analysis of Racing Circuit Construction in Indonesia*	Pembangunan sirkuit memicu polemik publik, ada yang merespon dengan positif dan negatif. Metode *Support Vector Machine dan IndoBERT* diajukan untuk melabeli respon positif dan negatif pada opini publik tersebut.	Membandingkan kinerja akurasi antara *Support Vector Machine* dan IndoBERT pada opini publik mengenai pembangunan sirkuit balap.	*Support Vector Machine* dan *pretrained* IndoBERT	Dataset berasal twitter sebanyak 2624 baris data. Dengan rasio 61.7% label positif dan 38.3% label negatif.	IndoBERT memiliki nilai *akurasi, precision, recall* dan *f1-score* yang lebih tinggi dari SVM dengan urutan 86% berbanding 82%, 88.2% berbanding 87.3%, 88.6% berbanding 84.3% dan 88.4%

						berbanding 85.8%.
Hadiyan Kundrat Putra, Moch Arif Bijaksana, Ade Romadhony, 2021	Deteksi Penggunaan Kalimat Abusive Pada Teks Bahasa Indonesia Menggunakan Metode IndoBERT	Mendeteksi kalimat abusive pada teks bahasa Indonesia dengan beberapa metode antara lain, IndoBERT, Bert Multilingual Base, BERT Base, KNN, SVM, Naive Bayes	Mengetahui hasil metode IndoBERT, dengan Bert Multilingual Base, BERT Base, KNN, SVM, dan Naive Bayes	*Crawling data, preprocessing text,* klasifikasi menggunakan *pretrained* IndoBERT	Dataset berasal dari *crawling* twitter pada kurun waktu Maret 2019 - September 2019 dengan total 3184 baris data dengan 3 label yaitu not abusive, abusive not offensive, abusive and offensive	Model IndoBERT menghasilkan performa terbaik di antara model lain dengan nilai rata-rata *f1-score* pada tiap kelas 76.32%.

Alex Sander Prasetya Braja, Achmad Kodar, 2022	Implementasi Fine-Tuning BERT untuk Analisis Sentimen terhadap Review Aplikasi PUBG Mobile di Google Play Store	*PUBG Mobile* telah menjadi game online salah satu game populer di Indonesia pada tahun 2022. Dengan kepopuleran tersebut pasti tidak akan terlepas dari ulasan pengguna demi pengembangan aplikasi yang lebih baik lagi, sehingga diperlukan analisis sentimen dengan metode BERT, yaitu IndoBERT dan BERT *Multilingual*.	Membandingkan kinerja IndoBERT dan BERT *Multilingual.*	Preprocessing data, Pelabelan data berbasis *TextBlob, Hyperparameter Tuning* pada model*(Optimizer Learning rate, batch size, dropout, max epoch, max length*).	Dataset berasal dari ulasan *google play store* pada sebanyak 15.000 baris data pada tanggal 4 April 2022.	Model IndoBERT memiliki performa yang lebih baik dengan akurasi sebesar 94% menggunakan learning rate 0.00002, *batch size* 32, *epoch* 5.

Siti Saadah, Kaenova Mahendra Auditama, Ananda Affan Fattahila, Fendi Irfan Amorokhman, Annisa Aditsania, Aniq Atiqi Rohmawati, 2021	*Implementation of BERT, IndoBERT, and CNN-LSTM in Classifying Public Opinion about COVID-19 Vaccine in Indonesia*	Pemerintah menyarankan agar masyarakat melakukan vaksinasi *Covid-19,* tetapi hal ini menimbulkan opini pro dan kontra, untuk mengetahui polarisasi opini publik diperlukan analisis sentimen dengan 3 model antara lain BERT, IndoBERT, dan CNN-LSTM	Membandingkan performa 3 model yaitu BERT, IndoBERT dan CNN-LSTM pada opini publik mengenai vaksinasi *Covid-19* di Indonesia	*Data collection*, data *pre-processing*, *labelling* data, pembuatan model, prediksi model	Data berasal dari IndoLEM dan IndoSMA	Performa terbaik didapat model IndoBERT dengan kesuksesan mengklasifikasi teks sebesar 80%.
Herlina Jayadianti, Wilis Kaswidjanti, Agung Tri Utomo, Shoffan Saifullah, Felix Andika	*Sentiment analysis of Indonesian reviews using finetuning IndoBERT and R-CNN*	Ulasan merupakan informasi yang bisa didapatkan oleh user mengenai suatu produk atau pelayanan. Namun, untuk memproses pelabelan ulasan secara manual membutuhkan	Membandingkan kinerja IndoBERT RCNN dengan IndoBERT base pada dataset IndoNLU	*Pretrained* IndoBERT dan *pretrained* IndoBERT + RCNN	Dataset berasal dari IndoNLU	IndoBERT-RCNN memiliki akurasi yang lebih baik dari IndoBERT *base* dengan nilai akurasi dan *f1-score*

Dwiyanto, Rafal Drezewski, 2020		waktu yang lama. IndoBERT dengan tambahan layer *Recurrent Convolutional Neural Network* (RCNN) diujikan untuk melabeli data secara otomatis.				berturut-turut sebesar 95.16% dan 93.27%.

2.2. Landasan Teori

2.2.1. Anotasi Sentimen

Anotasi sentimen adalah proses menandai teks dengan label sentimen, seperti positif, negatif, atau netral. Proses anotasi sentimen diperlukan dalam membangun model anotasi sentimen yang akurat. Hal ini membutuhkan pemahaman yang mendalam mengenai emosi manusia yang diinterpretasikan secara tertulis melalui opini atau komentar mengenai sesuatu. Pada beberapa literatur, pendekatan anotasi sentimen biasanya dibagi menjadi dua, yaitu melalui *machine learning approaches*, *lexicon-based approaches* [18], [19]. Pendekatan melalui *machine learning* merupakan metode yang paling banyak digunakan karena bergantung pada semantik kalimat atau komprehensi kalimat untuk melakukan klasifikasi sentimen. *Lexicon-based approach* memanfaatkan banyaknya daftar kata dan frasa yang umum digunakan untuk mengungkapkan sentimen tersebut positif, negatif, atau netral.

2.2.2. *Text Preprocessing*

Dalam mengembangkan model klasifikasi teks, *text preprocessing* sangat penting, karena data teks yang didapat dari penambangan teks seringkali tidak ideal dan terstruktur untuk dianalisa. Noise dan fitur yang tidak relevan bisa berpengaruh buruk pada performa sistem dalam banyak algoritma, terutama algoritma pembelajaran statistik dan probabilistik. Maka dari itu, diperlukan suatu proses untuk membuat data teks lebih terstruktur dengan menggunakan berbagai metode, seperti *case folding*, *removing noise*, *replacement of slang and abbreviations*, *removing stopwords, tokenization.*

1. *Case Folding*

   Pengubahan huruf atau *case folding* merupakan proses mengubah huruf kapital kapital menjadi huruf kecil, atau sebaliknya [20]. Proses ini biasanya digunakan untuk menyeragamkan atau

menyamakan semua huruf dalam suatu teks agar mudah diolah pada proses *natural language processing.*

2. *Removing Noise*

   Proses penghapusan noise sangat penting untuk meningkatkan akurasi, hal ini bertujuan untuk menghilangkan informasi yang tidak relevan dari teks sehingga mampu meningkatkan kualitas data, contohnya antara lain, menghapus karakter khusus, menghapus angka yang tidak memiliki makna, menghapus *url, email, html tags* [21].

3. *Replacement of Slang and Abbreviations*

   Penggantian *slang* dan singkatan dalam pemrosesan bahasa alami melibatkan penggantian kata-kata informal atau singkatan menjadi bentuk formal atau bentuk lengkapnya dengan tujuan untuk mengubah teks menjadi lebih standard dan seragam, sehingga kualitas data menjadi lebih baik [22].

4. *Removing stopwords*

   Stopwords diartikan sebagai kata-kata yang tidak memiliki makna signifikan pada suatu teks sehingga perlu dihilangkan [23]. Di antara contohnya adalah aku, kamu, ke, dan, atau, dari, pada, saat, sambil, dll. Penghapusan *stopwords* berguna untuk efisiensi dan akurasi pemrosesan teks.

5. *Tokenization*

   Proses pemecahan teks atau kalimat menjadi bagian-bagian yang lebih kecil (*token*) merupakan perngertian dari tokenisasi. Token bisa berupa kata, frasa, atau karakter. Tokenisasi merupakan langkah awal dalam pemrosesan bahasa alami *natural language processing* (NLP) yang membuat komputer mampu mengerti dan menganalisa teks [24] .

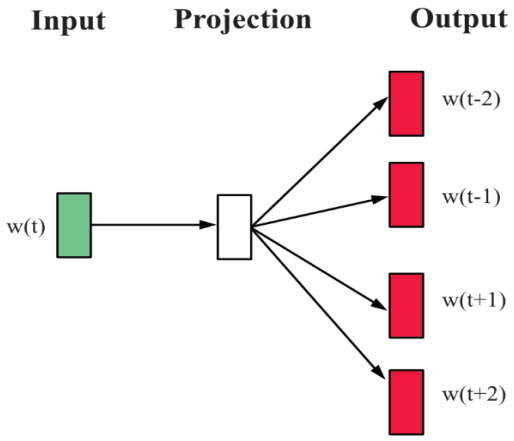
2.2.3. Word Embedding

Salah satu elemen penting dalam penelitian pemrosesan teks dan

input jaringan adalah *word embedding*. Dalam metode pembelajaran fitur yang dikenal sebagai "word embedding", setiap kata atau frasa dalam kosakata dimasukkan ke dalam vektor bilangan real berdimensi N. Fokus utama dari teknik ini adalah penetapan vektor yang mirip dengan kata-kata yang memiliki arti semantik yang sama. Salah satu contoh *word embedding* yaitu Word2Vec.

1. Word2Vec

   Dengan menggunakan pendekatan "*word to vector*", Mikolov menciptakan model yang meningkatkan arsitektur penerapan kata dari neural network language model (NNLM). Word2Vec mengadopsi jaringan syaraf dua lapisan tersembunyi. [25]. Word2Vec menggunakan *neural network* untuk menghasilkan vektor representasi kata, kedua strukturnya, *continous bag-of-words* (CBOW) dan *skip-gram*, digunakan untuk membuat vektor yang sangat besar yang mewakili setiap kata. Arsitektur Word2Vec sederhana terdiri dari tiga lapisan: *input*, *projection* (lapisan tersembunyi), dan *output*, seperti yang ditunjukkan pada gambar 2.1. Sewaktu memulai representasi kata dalam bentuk vektor, Word2Vec menggunakan vektor *one-hot encoded* dengan panjang yang sesuai dengan jumlah kata yang berbeda yang ada dalam data pelatihan

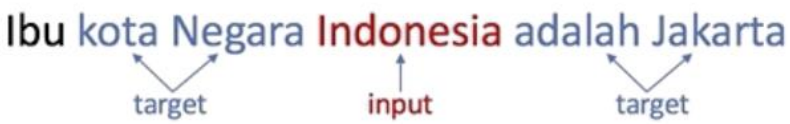


Gambar 2. 1 Arsitektur word2vec sederhana

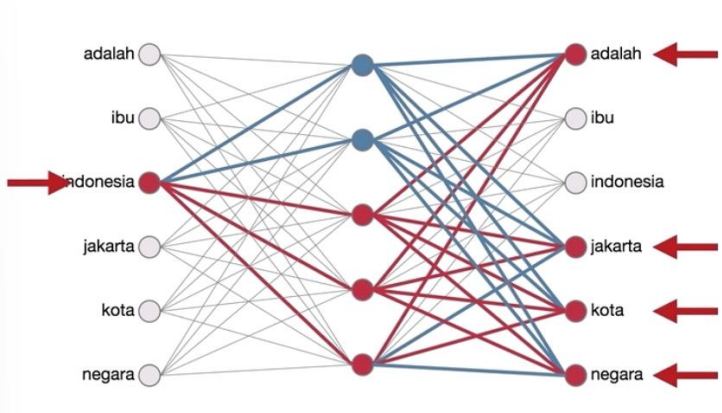
a). Continuous Skip-gram

Arsitektur *skip-gram adalah* model yang dikembangkan untuk memahami hubungan antar kata dan menghasilkan vektor kata yang merefleksikan aspek-aspek semantiknya, yang bertujuan untuk memprediksi konteks (*output*) di sekitar *current word (input)* [24]. *Skip-gram* memiliki performa yang lebih baik dari CBOW pada sebagian besar evaluasi, tapi CBOW lebih cepat untuk dilatih, Arsitektur *skip-gram* dapat dilihat pada Gambar 2.1 di atas.

Misalkan kita memiliki data *training* dengan kalimat "Ibu kota negara Indonesia adalah jakarta" dengan *window size* = 2 maka kita bisa ilustrasikan pada Gambar 2.2 dan 2.3.



Gambar 2. 2 Ilustrasi *windowing* skip-gram [26]



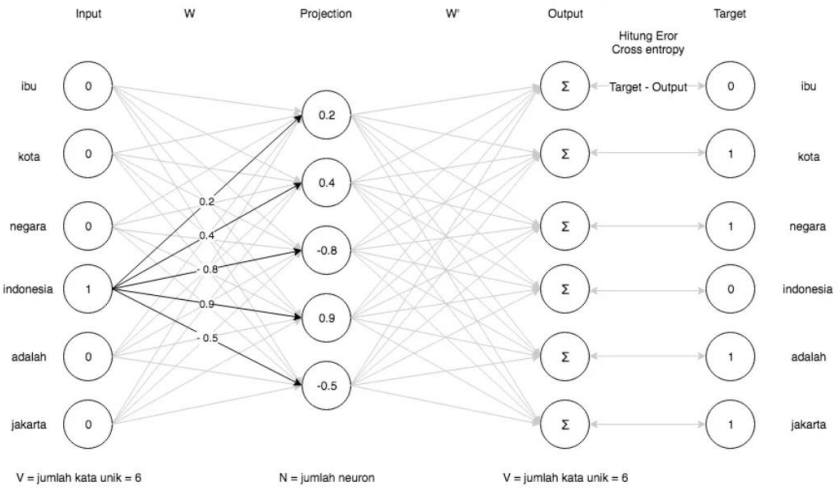
Gambar 2. 3 Ilustrasi struktur neural network skip-gram [26]

Dikarenakan data yang diberikan dimasukkan dalam bentuk vektor yang dikodekan satu kali, data tersebut akan berbentuk seperti yang ditunjukkan pada tabel 2.2 berikut.

Tabel 2. 2 *One-hot encoded vector*

No	Kata	***One-hot Encoded Vector***
1	ibu	[1,0,0,0,0,0]
2	kota	[0,1,0,0,0,0]
3	negara	[0,0,1,0,0,0]
4	indonesia	[0,0,0,1,0,0]
5	adalah	[0,0,0,0,1,0]
6	jakarta	[0,0,0,0,0,1]

Contoh, misalnya *current word* = Indonesia, sehingga ilustrasi bentuk struktur *neural network* akan tampak seperti pada Gambar 2.4



Gambar 2. 4 *Feed forward neural network* skip-gram [26]

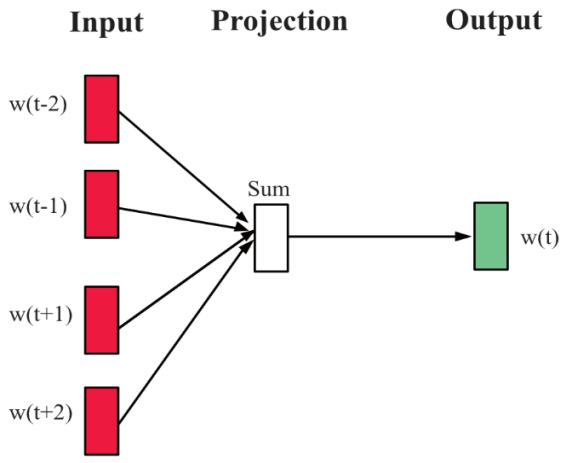
Pembobotan dimulai dengan nilai acak untuk W dan W'. Matriks W dan W' memiliki ukuran W = V x N dan W' = N x V. Untuk mendapatkan nilai pada lapisan proyeksi, langkah *feedforward* melibatkan perkalikan vektor *input* (*dot product*) dengan matriks bobot W. Untuk menghasilkan vektor *output*, nilai pada lapisan proyeksi akan dikalikan kembali dengan matriks bobot

W'. Setelah nilai *output* selama proses feedforward diperoleh, nilai *output* dikurangkan dari nilai target untuk menghitung kesalahan. Hal Ini dilakukan dengan menggunakan metode *cross entropy*. Langkah berikutnya adalah backpropagation. Dalam proses ini, bobot W dan W' diperbarui dengan menggunakan metode *gradient descent*. Semua langkah ini diulang dari tahap *feedforward* hingga nilai kesalahan minimum dicapai.

Vektor yang menggambarkan kata tersebut diambil dari matriks bobot W setelah mencapai nilai kesalahan minimum dalam *cross entropy*. Ini dilakukan dengan cara mengalikan *dot product* antara *one-hot encoded vector* dengan bobot W untuk setiap kata. Sebaliknya, bobot pada matriks W' diabaikan pada tahap ini.

b). Continuous Bag-of-Words

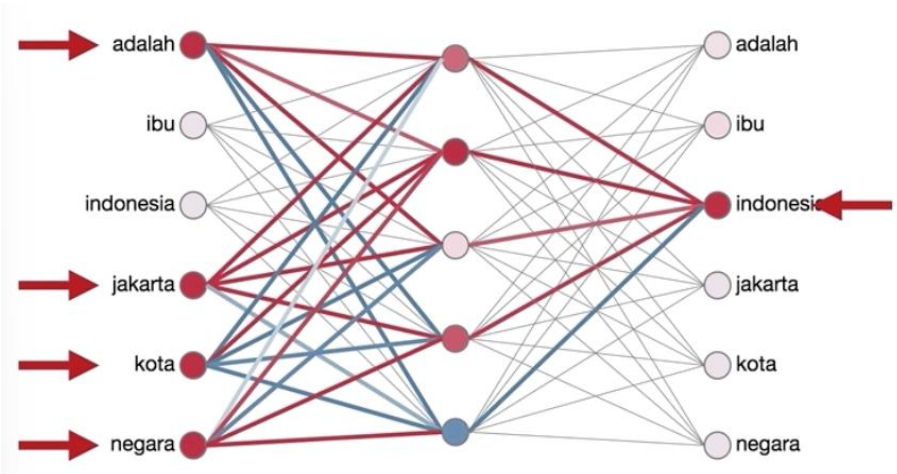
Word2Vec CBOW memiliki arsitektur yang berkebalikan dengan Word2Vec skip-gram. Fokusnya adalah memprediksi kata (*output*) berdasarkan konteks yang diberikan di sekitar kata tersebut (*input*). Ini merupakan tujuan utama dari model tersebut, sebaliknya dengan pendekatan skip-gram yang bertujuan untuk memprediksi konteks berdasarkan kata target. Ilustrasi dari konsep ini dapat dilihat pada gambar 2.5 berikut.



Gambar 2. 5 Skip-gram

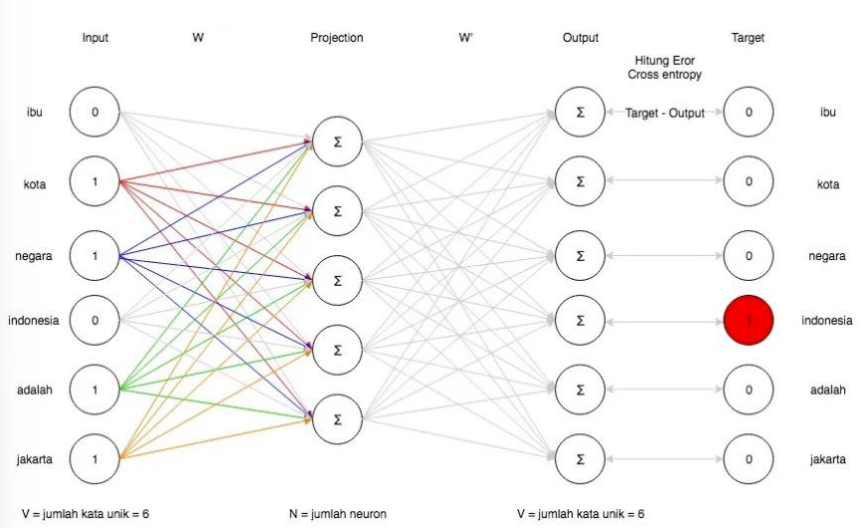
Misalkan contoh kalimat yang digunakan yaitu kalimat “ibu kota negara indonesia adalah jakarta”, maka akan tampak ilustrasi *windowing* CBOW pada gambar 2.6 dan *neural network* dari gambar 2.7.

Gambar 2. 6 Ilustrasi *windowing* CBOW [26]



Gambar 2. 7 Ilustrasi *neural network* CBOW [26]

Bentuk *n-hot encoded vector* mewakili data input Word2Vec *skip-gram* di mana kata yang menjadi *input* memiliki nilai 1, sementara kata lainnya memiliki nilai 0 selama proses pelatihan. Karena tujuan pelatihan hanya untuk memprediksi satu kata, maka target *output* direpresentasikan dalam bentuk *one-hot encoded vector*. Berikut ilustrasi pada Gambar 2.8 jika target *output* adalah kata “indonesia”.



Gambar 2. 8 Ilustrasi *feedforward* Word2vec CBOW [26]

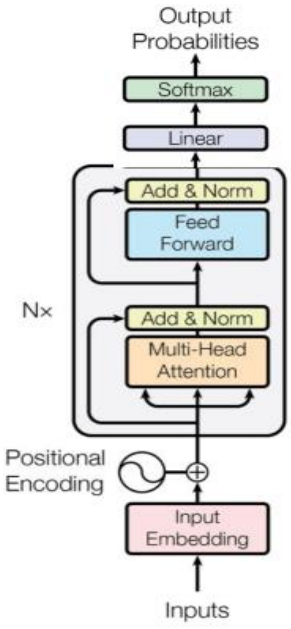
Proses ini hampir sama dengan arsitektur *skip-gram*, satu-satunya perbedaan adalah jenis data yang dimasukkan vektor *n-hot encoded* dan target output vektor *one-hot encoded*. Kita dapat mengambil vektor representasi kata dengan mengalikan *vektor one-hot encoded* untuk setiap kata dengan matriks bobot W setelah pelatihan selesai dengan kesalahan minimal.

2.2.4. IndoBERT

IndoBERT adalah modifikasi dari BERT-Base yang sudah ada, seperti yang dijelaskan dalam penelitian [27] Ini menggunakan konfigurasi BERT-Base *(uncased)*, dengan 12 hidden layer berdimensi 768, 12 *attention heads*, dan *hidden layer feed-forward* dengan dimensi 3,072. Lebih dari 220 juta kata digunakan untuk melatih IndoBERT, data yang digunakan bersal dari tiga sumber utama antara lain, Indonesia Web Corpus (90 juta kata), artikel Kompas Tempo & Liputan6 (55 juta kata), serta Wikipedia Indonesia (74 juta kata). IndoBERT terdiri dari tiga model utama antara lain, IndoBERT-liteBase, IndoBERTBase, dan IndoBERTLarge, ketiganya merupakan model BERT monolingual untuk bahasa Indonesia.

BERT berbasis pada transformer, suatu mekanisme *attention* yang memahami hubungan kontekstual antara kata atau subkata dalam teks.

Transformer terdiri dari dua mekanisme yang berbeda yaitu *encoder* untuk membaca teks *input* serta *decoder* untuk memprediksi tugas tertentu.. Namun, karena BERT difokuskan pada pengembangan model bahasa, hanya mekanisme *encoder* yang digunakan. Berbeda dengan model sekuensial yang membaca teks masukan secara berurutan dari kiri ke kanan atau sebaliknya, *encoder transformer* membaca seluruh urutan kata secara simultan. Oleh karena itu, meskipun dianggap sebagai non-arah, *transformer* dianggap memiliki dua arah karena memperhitungkan konteks kata dari kedua sisi (kiri dan kanan) [28]. Keunggulan ini memungkinkan model untuk memahami konteks kata berdasarkan seluruh kata dalam kalimat, arsitektur *transformer encoder* bisa dilihat pada Gambar 2.9.



Gambar 2. 9 Arsitektur *transformer encoder* [29]

Pada tahap ini terjadi serangkaian proses seperti pada gambar 2.10 tahap pertama yaitu memberlakukan tokenisasi pada semua kata, kemudian mengaplikasikan *embedding vector* pada tiap kata dan setiap kata memiliki vektor yang berdimensi 512 agar satu kata dengan kata lain memiliki kedekatan yang bisa direpresentasikan dalam bentuk vektor dua dimensi.

Misalnya, kita memiliki kata “Jakarta”, “Indonesia”, “Malaysia”,

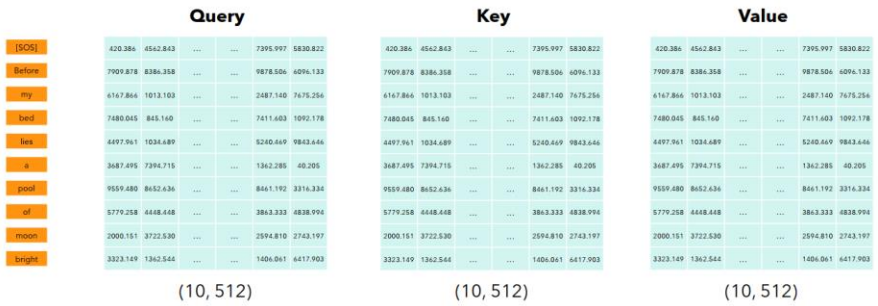
maka dalam representasi vektor dua dimensi kata “Jakarta” akan memiliki jarak yang lebih dekat dengan kata “Indonesia” dibanding dengan kata “Malaysia”. *Embedding vector* ini bisa dihitung menggunakan *cosine similarity*. Langkah selanjutnya yaitu menghitung *positional embedding. Positional embedding* bisa dihitung berdasarkan posisi *token* dalam suatu kalimat, perhitungan nilai dari *positional encoding* hanya dilakukan sekali. Sehingga nilai dari *encoder input* yang keluar setelah dilakukan *positional embedding* berasal dari *embedding vector* ditambah dengan nilai *positional embedding.*

Gambar 2. 10 Ilustrasi proses *postional encoding* [30]

Setelah nilai tersebut didapat kemudian nilai vektor itu dibagi menjadi tiga matrix dengan nilai vektor yang sama persis, ilustrasi nilai matriks dapat kita lihat pada Gambar 2.11. Mekanisme ini terjadi saat proses *multi head self attention,* pada proses ini juga terjadi perhitungan matematis dengan persamaan 2.1.

$$Softmax\ (x_i) = \frac{\square(\square)}{\square\square\square\square\square(\square)} \tag{2.1}$$

Sehingga nanti akan dihasilkan nilai matriks seperti pada gambar 2.12 di mana hal ini bertujuan agar setiap kata dengan kata lain memiliki nilai dan hubungan kontekstual. Kemudian matriks tersebut akan dikali dengan matriks Value agar outputnya dimensinya menjadi seperti awal.



Gambar 2. 11 Ilustrasi matriks *self attention mechanism* [28]

	[SOS]	Before	my	bed	lies	a	pool	of	moon	bright
[SOS]	0.62	0.19	0.02	0.02	0.04	0.01	0.00	0.09	0.00	0.02
Before	0.15	0.00	0.00	0.01	0.00	0.00	0.17	0.00	0.67	0.00
my	0.09	0.02	0.56	0.02	0.01	0.08	0.11	0.02	0.05	0.03
bed	0.10	0.06	0.03	0.00	0.53	0.12	0.01	0.11	0.00	0.04
lies	0.02	0.00	0.00	0.05	0.80	0.00	0.02	0.04	0.01	0.06
a	0.01	0.00	0.02	0.02	0.00	0.03	0.68	0.16	0.03	0.06
pool	0.00	0.16	0.02	0.00	0.03	0.56	0.00	0.00	0.22	0.01
of	0.22	0.00	0.01	0.05	0.19	0.44	0.00	0.00	0.04	0.04
moon	0.00	0.67	0.01	0.00	0.02	0.03	0.23	0.01	0.00	0.03
bright	0.06	0.00	0.03	0.03	0.43	0.21	0.03	0.06	0.13	0.03

Gambar 2. 12 Ilustrasi hasil perhitungan [28]

Pada langkah berikutnya, teks bahasa Indonesia akan diklasifikasikan dengan menggunakan IndoBERTBase. Penggunaan metode IndoBERT dilakukan dalam dua tahap, pretraining dan fine tuning. Pretraining terdiri dari dua langkah, masked language modelling (MLM) dan next sentence prediction (NSP). Sedangkan pada tahap *fine tuning* hanya secara spesifik ditraining untuk task tertentu.

1. *Pre training,* langkah pada pre training terdapat 2 tahap yaitu *masked language modelling* dan *next sentence prediction.*

   A). *Masked Language Modeling* (MLM)

   Sekitar 15% kata dari setiap urutan token diganti secara acak dengan token [MASK]. Model kemudian akan mencoba

memprediksi nilai sebenarnya dari kata yang telah di-MASK. Prediksi ini didasarkan pada konteks yang diberikan oleh kata-kata lain yang tidak di-MASK dan kata-kata yang ada dalam urutan. Sebagai ilustrasi contoh berikut.

Kalimat : “before my bed lies a pool of moon bright”

Encode : ‘before’, ‘my’, ‘bed, ’liest’, ‘a, ‘pool’, ‘of, ‘moon’, ‘bright’.

B). *Next Sentence Prediction* (NSP)

Pada setiap proses *pretraining*, model memilih sepasang kalimat pada tahap pelatihan BERT, yang disebut kalimat A dan kalimat B. Dalam 50% kasus pre-training, kalimat B benar-benar merupakan kelanjutan dari kalimat A (*IsNext*), sementara 50% sisanya, kalimat B dipilih secara acak dari korpus (NotNext). Hal ini untuk membantu model agar mampu membedakan dua kalimat selama pelatihan.

1. Token [CLS] disisipkan di awal kalimat dan token [SEP] disisipkan di akhir kalimat.

2. Setiap token yang memiliki *embedding* vektor tertentu, yang terdiri dari kalimat A dan kalimat B, dan *embedding* posisi ditambahkan ke setiap token untuk menunjukkan posisinya dalam urutan.

3. Setiap token memiliki *embedding* posisi yang menunjukkan posisinya dalam urutan. Berikut ini adalah contoh kalimat yang menggambarkan proses prediksi frasa berikutnya (NSP) pada Gambar 2.13.

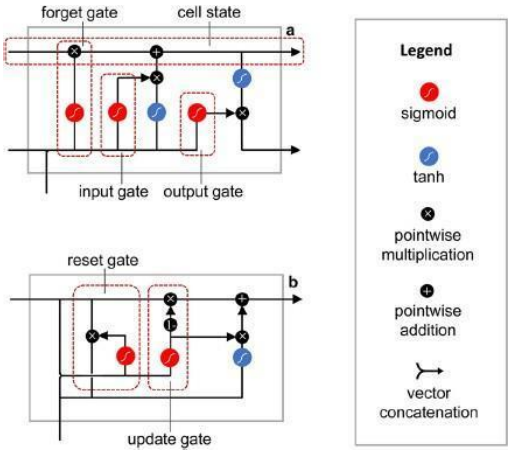
Input = [CLS] Indonesia merupakan salah satu negara asean [SEP]
Indonesia terletak di benua asia
Label = IsNext

Input = [CLS] Indonesia merupakan salah satu negara asean [SEP]
Malaysia terletak di benua asia
Label = NotNext

Gambar 2. 13 Contoh kalimat *next sentence prediction*

2.2.5. LSTM

LSTM merupakan kependekan dari *Long Short-Term Memory*, yaitu varian dari jaringan saraf rekursif (RNN) yang mampu menyimpan informasi dalam durasi yang panjang dan mengatasi masalah vanishing gradient yang dialami oleh RNN standar. LSTM dibuat untuk mengolah dan memprediksi data sekuensial, seperti teks, suara, dan video. LSTM memiliki tiga gerbang (*gate*) yaitu *forget gate, input gate*, dan *output gate* yang berperan mengendalikan aliran informasi di dalam jaringan [31].

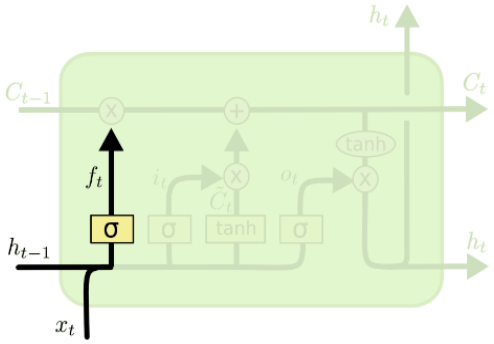


Gambar 2. 14 Arsitektur LSTM [32]

Seperti yang ditunjukkan pada gambar 2.14, LSTM memiliki struktur yang serupa tetapi memiliki gerbang pada setiap sel, gerbang ini berfungsi untuk menentukan apakah keadaan memori internal dipertahankan atau dihapus pada setiap *gate*. Secara sederhana *forget gate* berfungsi untuk meneruskan atau tidak informasi kata ke-t terhadap kata

sebelumnya. *Input gate* memberikan pembobotan berdasarkan *forget gate*, serta *output gate* akan menghasilkan pembobotan akhir dari kata ke-t untuk diteruskan ke kata selanjutnya.

Langkah pertama, misalnya kita memiliki suatu kata x1 = aku, x2 = cinta, x3 = Indonesia, kemudian kita masukkan *input* x2 ke dalam *cell state* menjadi xt. Dan x1 menjadi ht-1, ht-1 sendiri adalah state dari fase *short term memory* sebelumnya seperti pada gambar 2.15, lalu diaplikasikan persamaan 2.2 dan 2.3 sebagai berikut.



Gambar 2. 15 Struktur *forget layer* pada LSTM [31]

Keterangan:

$f_t = \sigma(W_f.[ht\text{-}1, xt] + b_f)$ (2.2)

$\sigma(x) = \frac{1}{(1+\epsilon^{-x})}$ (2.3)

σ fungsi sigmoid

f_t: forget gate

ϵ: konstanta matematika (2,71828)

x: data input

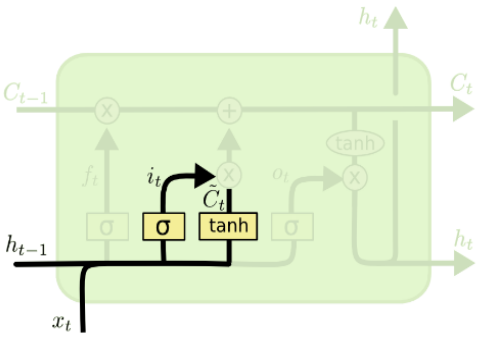
W_f: bobot forget gate

h_t-1: nilai keluaran sebelum orde ke t.

x_t: input pada orde ke t.

b_f: nilai bias pada forget gate.

Setelah didapatkan persamaan dari ft, maka nanti hasilnya akan diteruskan pada jalur *long term memory*, kemudian akan diteruskan kembali xt ke jalur fase *short memory* dengan mengkalikannya dengan persamaan it pada persamaan 2.4 melalui fungsi aktivasi sigmoid dan persamaan $\bar{C}_t$ pada persamaan 2.5 melalui fungsi aktivasi tanh seperti pada gambar 2.16 berikut.



Gambar 2. 16 Struktur *input gate* fase 1 [31]

Dimana:

it = σ(Wi.[ht-1, xt]+bi) (2.4)

σ fungsi sigmoid

Wi: nilai bobot untuk input gate

ht-1: keluaran sebelum orde ke-t

xt: input orde ke-t

bi: bias input gate

$\bar{C}_t$ = tanh(Wc [h_{t-1}, x_t] + bc) (2.5)

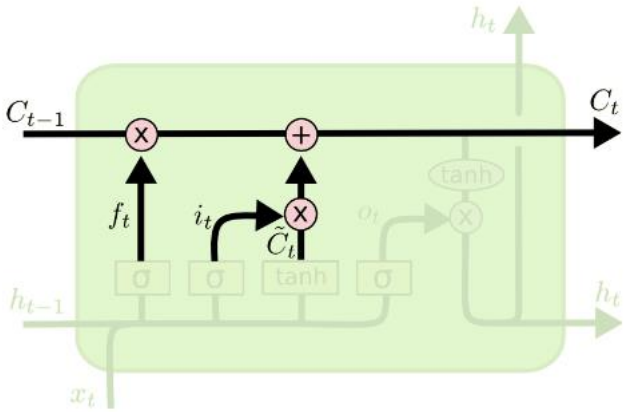
Wc : bobot untuk nilai cell state

h_{t-1}: keluaran sebelum orde ke-t.

xt: input orde ke-t

b_c bias cell state

Nilai dari forget gate yang sudah didapatkan dari persamaan 2.2 selanjutnya akan dikalikan dengan nilai ct-1 di mana ct-1 = fase long term memory sebelumnya. Selanjutnya nilai akan ditambah dengan hasil dari perkalian antara persamaan 2.4 dan persamaan 2.5. Di sini terjadi proses untuk suatu kata tersebut dikatakan relevan atau tidak dengan kata sebelumnya, jika tidak maka informasi tersebut akan dihilangkan, dan jika kata itu relevan, informasi pada sell lstm akan tetap disimpan dan akan diteruskan. Pada jalur long term memory pada gambar 2.17 di bawah ini.



Gambar 2. 17 Struktur *input gate* fase 2 [31]

Selanjutnya contoh kata dari xt juga akan melewati persamaan ot pada persamaan 2.6 di mana hasil dari persamaan tersebut nilainya akan dikali dengan tanh dari hasil ct baru pada persamaan 2.7 berdasarkan jalur *long term memory* pada gambar 2.18. Hasil dari perkalian tersebut akan menghasilkan persamaan ht yang baru pada persamaan 2.8.

Keterangan:

$ot = \sigma(Wo. [h_{t-1}, x_t] + b_o)$ (2.6)

 fungsi sigmoid

W bobot nilai output gate.

H l: nilai keluaran sebelum orde ke-t.

x nilai input pada orde ke t.

bo: bias output gate.

 = x + + (2.7)

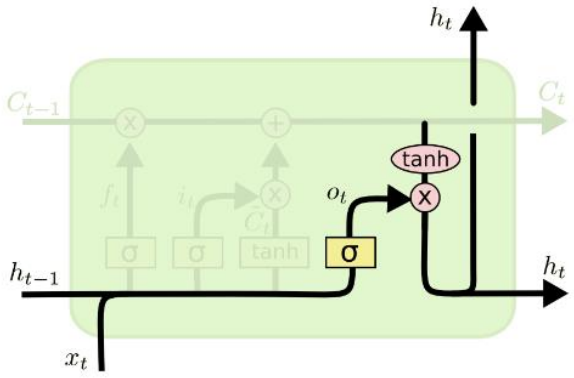
f forget gate.

C-1: nilai sel *state* sebelum orde ke t.

it: nilai input gate.

C hasil penambahan nilai baru pada cell state.

h = C × tanh × o (2.8)

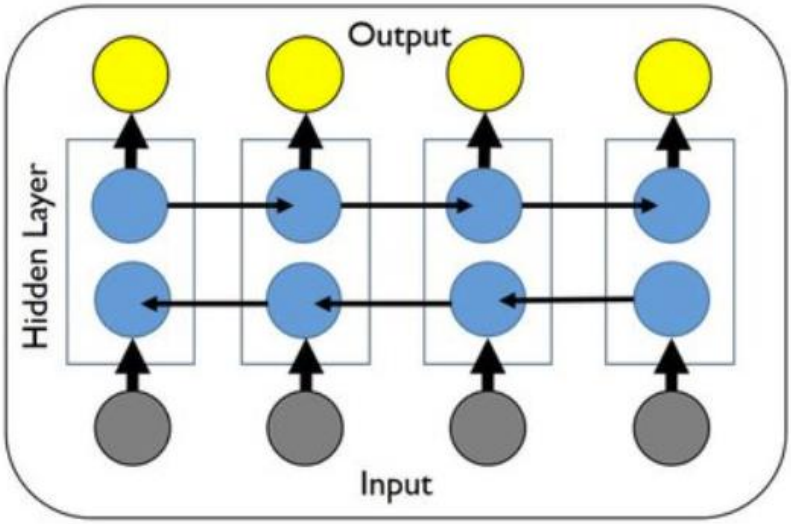


Gambar 2. 18 Struktur *output gate* pada LSTM [31]

2.2.6. BILSTM

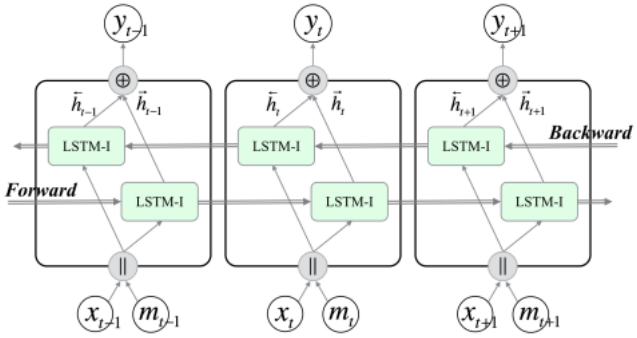
LSTM mengalami kesulitan ketika harus berhadapan dengan konteks yang memiliki dua arah sehingga pada tahun 2005 dikemukakan suatu penelitian [15] yang menerapkan jaringan LSTM yang dilatih secara

paralel. Satu jaringan membaca data dari urutan kiri ke kanan, jaringan yang lain membaca urutan data sebaliknya. Kemampuan ini dianggap mampu untuk menanggulangi kelemahan pada arsitektur LSTM, sehingga BiLSTM dapat digunakan untuk berbagai keperluan tugas *natural language processing,* seperti analisis sentimen, terjemahan mesin, dan pengenalan bicara, pada tugas analisis sentimen, BiLSTM juga mampu untuk menangkap nuansa emosional kalimat dengan mempertimbangkan kata-kata sebelumnya dan selanjutnya, sehingga pemahaman akan konteks suatu kalimat lebih komprehensif dan bermakna.



Gambar 2. 19 Arsitektur BiLSTM [33]

BiLSTM menggunakan data dari kedua arah untuk memanfaatkan informasi sebelumnya dan sesudahnya. Lapisan maju bertanggung jawab untuk mereprentasikan informasi sebelumnya, sementara lapisan mundur bertanggung jawab untuk merepresentasikan informasi setelahnya, seperti yang terlihat pada gambar 2.20 sebagai berikut.



Gambar 2. 20 Arsitektur *hidden layer* BiLSTM [34]

Hidden layer yang memiliki dua arah yang berlawanan menjadikan model mampu memahami data dengan baik dari *forward* maupun *backward*. Hal ini memungkinkan proses pelatihan untuk memahami data dalam rangkaian waktu karena konsep *forget, input,* dan *output gate* menghasilkan pembobotan apakah suatu kata tersebut relevan atau seberapa penting kata tersebut berhubungan dengan kata sebelum atau seudahnya. Karena kemampuan BiLSTM untuk mengakses informasi dari kedua arah sebelum dan sesudahnya BiLSTM sangat berguna dalam pelatihan sekuensial.

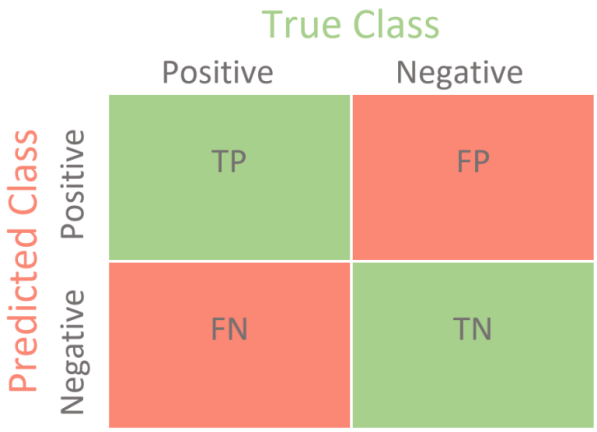
2.2.7. *Classification Report*

Classification report adalah alat evaluasi kinerja yang umum digunakan dalam konteks klasifikasi pada kasus yang memiliki output dua kelas atau lebih, evaluasi ini dapat diambil berdasarkan matriks konfusi. Matriks ini terdiri dari empat atribut yang mewakili kombinasi antara nilai yang diprediksi oleh model (*predicted*) dan nilai yang sebenarnya dari data (*actual*)

a. TP (*True Positive*) merupakan jumlah kasus di mana model memprediksi dengan benar bahwa suatu sampel termasuk dalam kelas positif.
b. TN (*True Negative*) merupakan jumlah kasus di mana model memprediksi dengan benar bahwa suatu sampel termasuk dalam kelas negatif.
c. FP (*False Positive*) merupakan jumlah kasus di mana model salah

memprediksi bahwa suatu sampel termasuk dalam kelas positif, padahal seharusnya sampel termasuk kelas negatif..

d. FN (*False Negative*) merupakan jumlah kasus di mana model salah memprediksi bahwa suatu sampel termasuk dalam kelas negatif, padahal seharusnya sampel termasuk kelas positif.



Gambar 2. 21 Konfusi matriks [35]

Keempat atribut tersebut akan menjadi dasar perhitungan beberapa matrik evaluasi, antara lain:

a). Akurasi adalah rasio prediksi yang benar (positif atau negatif) dengan semua data. Metrik ini paling populer karena mudah dihitung dan digunakan. Namun, metrik ini kurang akurat untuk data yang tidak seimbang. Untuk mengetahui seberapa akurat, persamaan 2.9 dapat digunakan.

$$\text{Accuracy} = \frac{\text{TP}+\text{TN}}{\text{TP}+\text{TN}+\text{FP}+\text{FN}} \tag{2.9}$$

b). Presisi merupakan rasio antara *true positive* dengan keseluruhan data yang diprediksi positif. Sehingga, *precision* berusaha memperkecil terjadinya false positive. Nilai presisinya dapat dihitung dengan menggunakan persamaan 2.10 berikut:

$$\text{Precision} = \frac{\text{TP}}{\text{TP}+\text{FP}} \tag{2.10}$$

c). *Recall* merupakan rasio antara *true positive* dengan keseluruhan

data yang kenyataannya bernilai positif. Sehingga, recall berusaha memperkecil terjadinya false negative. Nilai *recall* dapat diperoleh dengan persamaan 2.11.

$$\text{Recall} = \frac{\text{TP}}{\text{TP+FN}} \tag{2.11}$$

d). *F1-score* adalah matrik evaluasi yang mengukur keseimbangan antara presisi (*precision*) dan *recall* (sensitivitas) dalam model klasifikasi. Nilai F1-score dapat diperoleh dengan menggunakan persamaan 2.12.

$$\text{F1} - \text{Score} = 2 \times \frac{\text{Precision x Recall}}{\text{Precision+Recall}} \tag{2.12}$$

Jika nilai true negatif (TN) dan true positive (TP) lebih besar daripada nilai false negative (FN) dan false positive (FP), model klasifikasi dikatakan baik. Tabel Confusion Matrix menunjukkan jumlah hasil prediksi benar dan salah, yang membuktikan model secara akur.

2.2.8. *Cross entropy*

Cross entropy adalah fungsi loss yang mengukur perbedaan antara dua distribusi probabilitas antara distribusi yang sebenarnya dan distribusi yang diprediksi oleh model [36]. Dalam konteks pembelajaran mesin dan statistik, *cross entropy* digunakan untuk menilai seberapa baik model klasifikasi memprediksi label kelas yang benar. Semakin kecil nilai *cross entropy*, semakin baik model tersebut memprediksi. Klasifikasi untuk masalah biner bisa dilakukan dengan *binary cross entropy*. Berikut persamaan *binary cross entropy.*

$$\text{L} = -\frac{1}{\square} = \sum_{\square=1}^{\square} [\square\square\log(\square\square) + \log(1 - \square\square)] \tag{2.13}$$

yi: label sebenarnya untuk sampel I (0 atau 1)

pi: probabilitas yang diprediksi oleh model untuk kelas 1

N: jumlah sampel