BAB 2
TINJAUAN PUSTAKA

2.1 Spesialis Sosial Media

Spesialis Sosial Media atau *Social Media Specialist* adalah profesional yang bertanggung jawab untuk membangun dan memelihara kehadiran brand perusahan di media sosial. Mereka bertugas untuk mengembangkan strategi konten, membuat konten yang menarik, dan membangun hubungan dengan audiens [15]. Pekerjaan tersebut memiliki tanggung jawab utama yang mencakup beberapa aspek penting dalam pengelolaan platform media sosial. Berikut merupakan tugas dari *Spesialis Media Sosial :*

1) Mengembangkan dan mengimplementasikan strategi konten yang sesuai dengan pengguna untuk meningkatkan *engganment rate* atau interaksi dengan pengguna [15].
2) Meningkatkan *branding* dari sosial media brand atau merek terhadap pengguna [15].
3) Menjaga hubungan dan komunikasi yang baik serta efektif terhadap pengguna [15].

Disisi lain, dari hasil temuan pada saat wawancara terdapat tanggung jawab dari pekerjaan dari *Spesialis Media Sosial* yang memiliki tanggung jawab seperti berikut [LAMPIRAN C]:

1) Mengelola komunitas online, menjawab pertanyaan pelanggan, dan memelihara hubungan baik pelanggan.
2) Mencari *influencer* yang cocok dengan merek yang dikelola, dan berkolaborasi dengan mereka untuk memperluas jangkauan dan meningkatkan kehadiran merek di media sosial.

2.2 Twitter atau X

Twitter atau X merupakan layanan jejaring sosial dan *online microblog*, pengguna dapat membagikan opininya sendiri hingga saling berinteraksi dengan pengguna lain [16]. Twitter sendiri layanan jejaring sosial online yang memungkinkan penggunanya untuk membuat dan berinteraksi dengan pesan singkat (tweet). *Tweet* dapat berisi teks, gambar, video, dan tautan. Pengguna dapat mengikuti tweet dari pengguna lain, membalas tweet, dan retweet (me-retweet) tweet orang lain [17]. Menurut Kwak et al. (2010), secara umum terdapat beberapa istilah umum yang sering muncul pada twitter [18]:

2.1.1. Tweet

Tweet merupakan pesan singkat yang dibagikan di Twitter dengan tweet batas karakter 140 (sekarang 280) yang dapat berisi teks, foto, video, GIF, dan tautan [16]. Tweet dapat digunakan untuk berbagai macam hal, seperti berbagi berita dan informasi, mengungkapkan pendapat dan ide, berkomunikasi dengan sesama pengguna dan mengikuti tren dan peristiwa terkini

2.1.2. Retweet

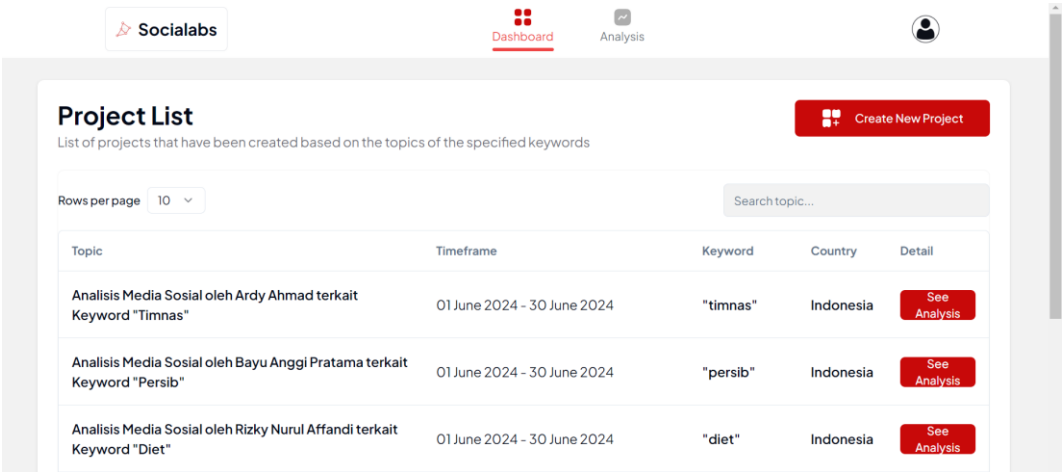
Retweet adalah fitur yang memungkinkan pengguna membagikan tweet orang lain ke pengikut [16]. *Retweet* dapat dilakukan dengan dua cara yaitu *retweet* standar dan *Quote Tweet* dengan menambahkan komentar atau pendapat sebelum membagikan tweet orang lain. *Retweet* bertujuan untuk menyebarkan informasi dan berpartisipasi dalam mengomentari suatu topik tertentu.

2.1.3. Follower

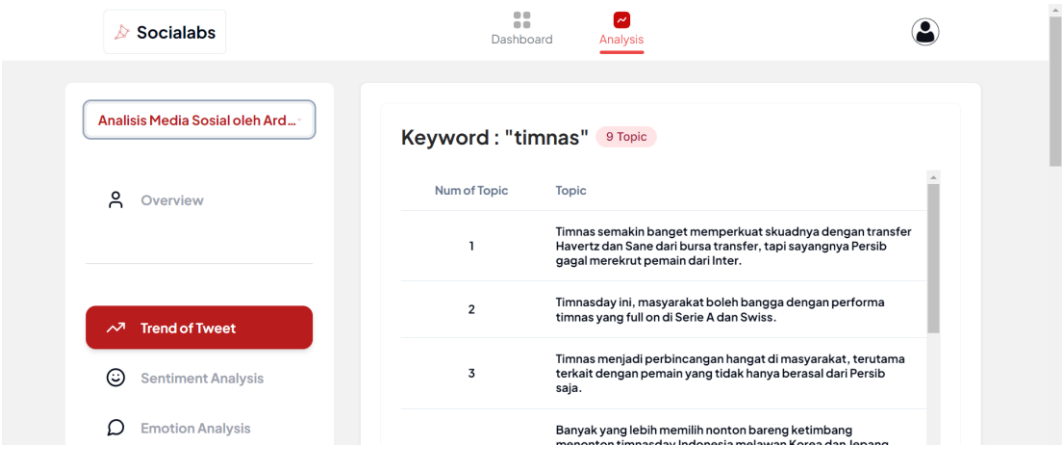
Follower merupakan pengguna Twitter yang mengikuti akun dan melihat tweet di *feed* pengguna, sehingga semakin banyak follower, semakin banyak orang yang akan melihat tweet bearti tweet tersebut akan trending [16].

2.3 Aplikasi Analisis Media Sosial

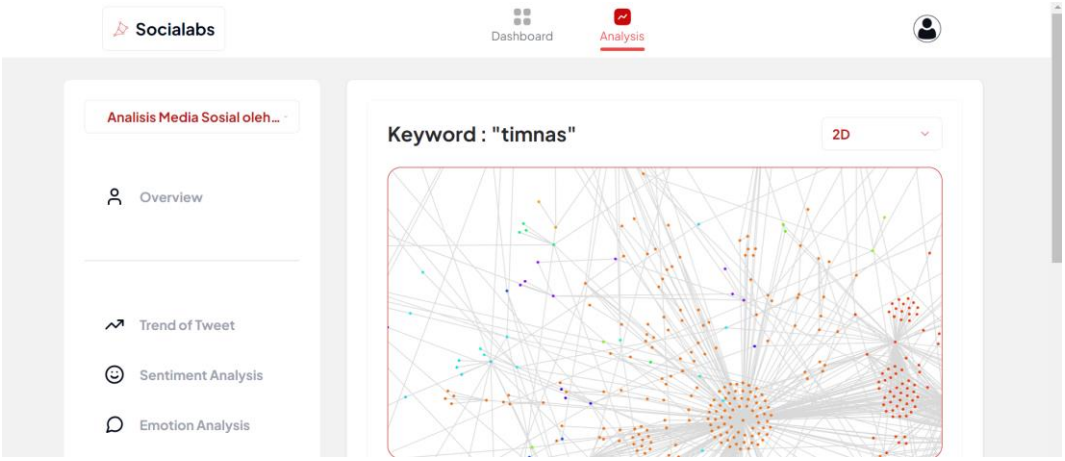
Aplikasi analisis media sosial adalah sebuah aplikasi yang dapat membantu dalam menganalisis data yang ada di sosial media seperti Twitter. Dalam aplikasi ini terdapat 2 fitur utama yaitu *Topic Modelling,* yaitu sebuah fitur untuk mengidentifikasi topik apa saja yang ada pada sebuah dokumen atau data. Misalnya dalam sebuah kumpulan tweet tentang "Politik". Dengan menggunakan fitur ini, aplikasi akan menganalisis teks dari tweet tersebut. Hasilnya berupa kumpulan kalimat-kalimat yang ada pada tweet atau dokumen tersebut. Fitur kedua adalah sebuah yang memanfaatkan *Social Network Analysis*, yaitu memvisualisasikan informasi yang didapat dari fitur topic modelling dari media sosial twitter. Misalnya aplikasi akan memvisualisasikan pengguna (*Buzzer*) yang sering di-*retweet* atau *reply* oleh pengguna lain atau sering muncul dalam percakapan tentang topik tertentu seperti "Politik". Aplikasi ini menggunakan proses *crawling* dengan memanfaatkan *tweet harvest* sebagai *library* untuk mengambil tweet dari sosial media lalu mengambil secara berkala untuk tweet terbaru dalam rentang 7 hari terakhir berdasarkan kata kunci yang ingin dicari. Dengan menggunakan metode ini, aplikasi dapat menghasilkan berbagai topik yang mencerminkan tweet-tweet yang relevan dengan kata kunci tersebut dalam periode waktu tertentu. Berikut merupakan beberapa tampilan dari aplikasi analisis media sosial pada Gambar 2.1 sampai Gambar 2.3.



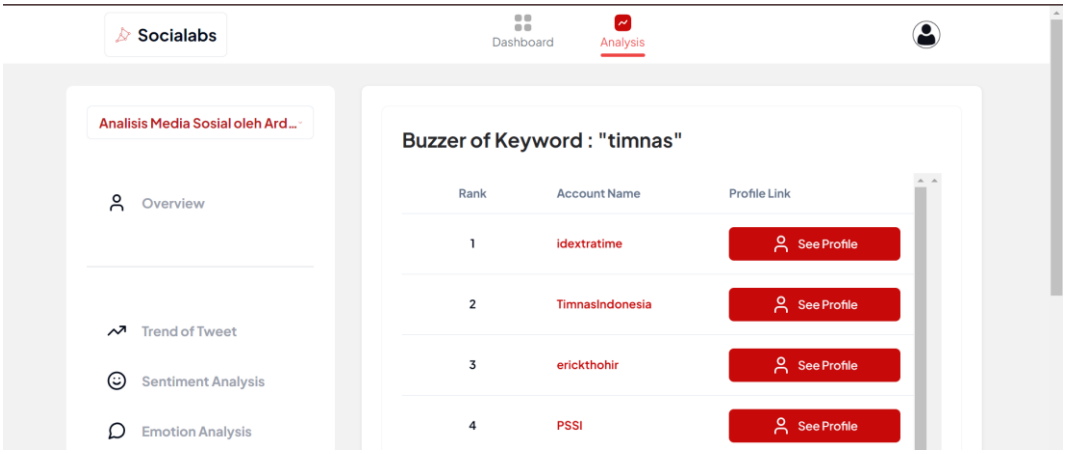
Gambar 2.1 Fitur *Project*



Gambar 2.2 Tampilan Fitur Tren Topik



Gambar 2.3 Tampilan Fitur Visualisasi *Social Network Analysis* (SNA)



Gambar 2.4 Tampilan Fitur *Buzzer*

2.4 Deep Learning

Deep learning merupakan subdisiplin dari *Machine learning* yang merupakan bagian integral dari bidang Kecerdasan Buatan. Teknik *Deep learning* merupakan bagian dari jaringan saraf yang terkenal karena menggunakan struktur multi*layer*, yang sering digunakan karena mampu menangani berbagai masalah sekaligus [19]. *Deep learning* memungkinkan pembelajaran dari data untuk menghasilkan representasi yang baik dari data tersebut, sehingga dapat melakukan prediksi dengan akurat. Terdapat tiga metode utama dalam Deep Learning, berikut metode dari *Deep learning* :

2.4.1 *Supervised*

Supervised learning adalah jenis *machine learning* di mana model dilatih menggunakan data berlabel. Data berlabel terdiri dari data *input* (fitur) dan *output* yang diinginkan (label) [19]. Model mempelajari hubungan antara data *input* dan *output*, dan kemudian menggunakan pengetahuan ini untuk membuat prediksi pada data baru yang tidak terlihat seperti klasifikasi. Klasifikasi adalah teknik dalam *machine learning* dan statistik yang digunakan untuk mengidentifikasi kategori atau kelas dari data baru berdasarkan model yang telah dilatih menggunakan data yang sudah diklasifikasikan sebelumnya.

2.4.2 *Semi - Supervised*

Algoritma semi-*supervised* terletak di antara algoritma *supervised* dan un*supervised*. Prinsip kerjanya melibatkan penemuan dan analisis struktur pada variabel *input*. Setelah identifikasi struktur, algoritma ini menghasilkan prediksi optimal untuk data yang tidak berlabel dan menggunakannya sebagai data latih tambahan untuk algoritma *supervised* guna membuat prediksi yang lebih baru [19].

2.4.3 *Unsupervised*

Un*supervised* learning adalah paradigma pembelajaran mesin di mana model dilatih pada data tidak berlabel. Data tidak berlabel terdiri dari fitur data tanpa label yang terkait [19]. Model mempelajari struktur atau pola yang tersembunyi dalam data, lalu menggunakan pengetahuan ini untuk membuat

prediksi atau menjalankan tugas lainnya seperti *clustering*. *Clustering* adalah teknik *machine learning* yang mengelompokkan data ke dalam kelompok (*clusters)* berdasarkan kemiripan atau kedekatan satu sama lain. Dalam model unsupervised yang digunakan proses ini adalah topik modelling.

2.5 Topic Modelling

Topic modelling merupakan data teks berdasarkan pengelompokan topik tertentu berdasarkan kata kunci. *Topic modelling* termasuk kedalam *clustering* dengan mengelompokkan dokumen berdasarkan kemiripannya [20]. Dokumen ini berasal dari hasil pengumpulan data yang sangat besar dari Twitter, yang kemudian akan digunakan dalam analisis *clustering topic modeling*. Berikut pada Gambar 2.5 merupakan cara kerja dari *topic modelling*.

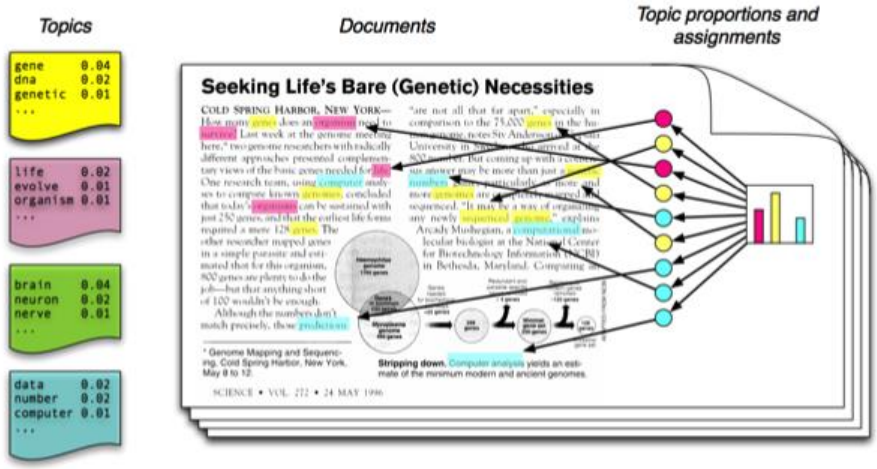


Figure source: Blei, D. M. (2012). Probabilistic topic models. *Communications of the ACM*, *55*(4), 77-84.

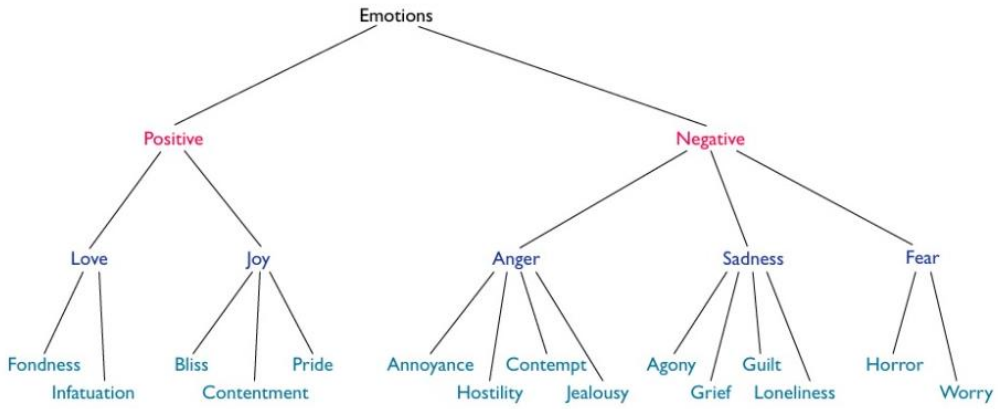
Gambar 2.5 Cara Kerja *Topic Modelling*

Dalam aplikasi analisis media sosial, model yang digunakan adalah Latent Directly Allocation (LDA). Latent Dirichlet Allocation (LDA) adalah model generatif yang digunakan untuk mengidentifikasi struktur topik dalam kumpulan dokumen teks yang besar dan tidak terstruktur [20]. Model ini mengasumsikan bahwa setiap dokumen adalah campuran dari sejumlah topik, dan setiap topik adalah distribusi probabilistik dari kata-kata. Sehingga, digunakan oleh aplikasi analisis media sosial dalam memperoses tweet menjadi sekumpulan topik.

Dalam fitur topik modelling, menerapkan juga evaluasi topik dengan menggunakan agregasi topik untuk menentukan seberapa banyak jumlah topik. Sehingga, hasil dari topik yang dihasilkan sesuai dengan konteks yang tersedia pada tweet yang telah di-*crawling*.

2.6 Emosi

Emosi, menurut Feldman, emosi merupakan perasaan yang secara umum memiliki elemen fisiologis dan kognitif yang berpengaruh pada perilaku. Emosi ini merupakan suatu proses dalam mempersiapkan suatu tindakan dan kemudian dapat membentuk suatu pola yang dapat memprediksi perilaku berikutnya [21]. Sementara Emosi sendiri terbagi kedalam "Hirarki", yang menjelaskan bahwa 5 emosi dasar (*Love*, *Joy*, *Anger*, *Sad* dan *Fear*) dapat bercampur dan menghasilkan emosi yang lebih kompleks [11]. Sebagaimana yang ditunjukan pada Gambar 2.6 yang merupakan hirarki dari berbagai jenis emosi.



Gambar 2.6 Hirarki Emosi

Emosi sendiri dibagi menjadi dua yaitu verbal dan non verbal, emosi verbal dapat diartikan emosi langsung atau menggunakan lisan maupun tulisan sebagai medianya. Sedangkan non verbal biasanya menggunakan bahasa tubuh untuk mengekspresikannya [22] . Berbagai jenis emosi tersebut khususnya verbal dapat teridentifikasi dalam bentuk teks yang merujuk pada kemampuan untuk mengetahui atau menafsirkan keadaan emosional seseorang melalui kata-kata yang digunakan dalam komunikasi tertulis [23]. Ketika seseorang mengekspresikan diri

mereka dalam bentuk tulisan, mereka cenderung mencerminkan perasaan, pikiran, atau suasana hati mereka melalui pilihan kata, frasa, dan gaya penulisan [24]. Dari berbagai jenis emosi tersebut, terdapat 5 emosi dipakai dalam penelitian terhadap analisis emosi untuk teks dan kebutuhan dari responden terhadap jenis emosi yang dibutuhkan [25] [LAMPIRAN E]. Penambahan label netral sebagai label keenam dilakukan untuk mencakup data dengan karakteristik yang tidak termasuk dalam lima label emosi dasar tersebut. Pemilihan jenis emosi ini merujuk pada jenis emosi yang digunakan, berdasarkan pada pengetahuan yang diperoleh dari paper "A Review on Teks-Based Emotion Detection" [26]. Pada Tabel 2.1 merupakan jenis serta deskripsi dari emosi yang sering dipakai dalam peneltian.

Tabel 2.1 Jenis Emosi

No.	Jenis Emosi	Deskripsi
1	Senang atau "*Joy*"	Perasaan positif yang dicirikan oleh rasa gembira, bahagia, dan puas. Senang sering dikaitkan dengan senyuman, tawa, dan peningkatan energi [27] **Seperti :** perasaan senang terkait tayangan atau jasa yang dilakukan **Perasaan yang bersifat sementara**
2	Takut atau "*Fear*"	Perasaan tidak nyaman yang muncul sebagai respons terhadap ancaman atau bahaya yang dirasakan [27] **Seperti :** perasaan takut terhadap tayangan horor
3	Cinta atau "*Love*"	Perasaan kasih sayang, perhatian, dan kepedulian terhadap orang lain [27] **Seperti :** perasaan cinta kepada karakter tertentu

No.	Jenis Emosi	Deskripsi
		Perasaan yang bersifat selamanya
4	Sedih atau "*Sad*"	Perasaan negatif yang dicirikan oleh rasa kehilangan, kesedihan, dan keputusasaan [27] **Seperti :** perasaan sedih terkait keluhan yang diterima atau dari tayangan yang sifatnya emosional bagi penonton
5	Marah atau "*Anger*"	Perasaan negatif yang dicirikan oleh rasa kesal, frustrasi, dan kejengkelan [27] **Seperti :** perasaan marah terhadap suatu peran antagonis tertentu atau protes terhadap layanan yang diterima
6.	Netral atau "*Neutral*"	Perasaan yang tidak termasuk label emosi manapun

2.7 Teks Mining

Teks mining merupakan sebuah proses yang menggabungkan berbagai teknik dari *data mining*, *machine learning*, *natural language processing*, *information retrieval*, dan *knowledge management* untuk mengekstrak wawasan yang bermakna dari sejumlah besar data teks yang tidak terstruktur [28]. Proses ini memanfaatkan berbagai alat dari berbagai bidang untuk menganalisis teks secara efektif. Tujuan dari *text mining* adalah untuk mengidentifikasi pola, tren, dan pengetahuan tersembunyi dalam teks [9].

Dalam proses *text mining* yang diterapkan, terdapat tahapan yang akan dipakai dalam natural language processing adalah sebagai berikut yang bisa di lihat pada Gambar 2-4 Tahapan Proses Dari Teks Mining [26]**.**

Gambar 2.7 Tahapan Proses Dari Teks Mining

Berikut penjelasan dari tahapan-tahapan *teks mining* :

1) Pengumpulan Data *(Crawling*) merupakan proses pengumpulan data *tweet* dari media sosial Twitter menggunakan alat web scraping [29] . Dalam proses *crawling* untuk mengumpulkan data dari Twitter, terdapat beberapa tahapan yang perlu dilalui mulai dari analisis kata kunci hingga perolehan banyak tweet yang sesuai dengan kata kunci tersebut. *Crawling* data bisa dilakukan secara manual melalui website atau aplikasi pihak ketiga, atau secara otomatis menggunakan script *crawling* [30]. Contoh salah satu aplikasi pihak ketiga yang bisa digunakan adalah "*Tweet Harvest"*, yang mengambil data tweet berdasarkan kata kunci dengan rentang waktu terbaru.
2) Pembersihan *(Preprocessing)* merupakan serangkaian langkah untuk membersihkan, mengatur, dan mengubah data mentah menjadi bentuk yang lebih mudah dianalisis [28]. Proses ini berfokus pada menghilangkan noise, seperti tautan, tanda baca yang tidak relevan, serta format khusus seperti *retweet* atau *mention*, sehingga data yang dihasilkan lebih terfokus dan sesuai dengan kebutuhan analisis [26]. Tujuan pembersihan tersebut untuk mempersiapkan data agar siap digunakan dalam analisis yang lebih lanjut.
3) Ekstrasi fitur *(Feature Extraction)* merupakan proses transformasi data asli ke representasi yang lebih rendah dimensinya, yang menangkap informasi relevan untuk tugas belajar tertentu [31]. Proses ini menggunakan *Word2vec* untuk mengubah kata-kata menjadi vektor numerik yang merepresentasikan makna semantiknya. Ekstraksi fitur dengan *Word2vec* merupakan teknik yang efektif untuk meningkatkan performa model pembelajaran mesin dalam berbagai tugas yang melibatkan teks. *Word2vec* membantu model memahami makna

semantik teks dan menghasilkan representasi data yang lebih ringkas dan informatif [32].

4) Pembuatan Model *(Model Development)* merupakan proses membangun model statistik atau *machine learning* dari data untuk mengekstrak informasi dan pengetahuan yang berguna [28]. Dalam penerapannya ke metode *Deep learning* dengan algoritma CNN dalam menganalisis informasi dari klasifikasi tweet berdasarkan emosi, *kita dapat mengemba*ngkan pendekatan yang canggih untuk memahami dan menginterpretasi perasaan yang terungkap dalam *tweet* tersebut.
5) Pengujian Model *(Model Assesment)* merupakan proses kritis dalam evaluasi kinerja sebuah model statistik atau algoritma dalam memprediksi atau mengklasifikasikan data baru yang tidak terlihat selama pelatihan [28]. Tujuan dari pengujian model adalah untuk memastikan bahwa model yang dibangun memiliki kemampuan yang baik untuk menggeneralisasi pola dari data latih ke data baru yang belum pernah dilihat sebelumnya.

2.8 Natural Language Processing (NLP)

Natural Language Processing (NLP) atau pengolahan bahasa alami merupakan sub-bidang dari kecerdasan buatan yang terkait dengan linguistik [33]. Linguistik adalah studi yang berkaitan dengan bahasa, yang di dalamnya termasuk struktur, tata bahasa, makna, dan frase [34]. NLP berfokus pada bagaimana cara agar komputer dapat memahami dan memproses teks atau ucapan dalam bahasa manusia. Dengan NLP, suatu teks dapat diubah ke dalam kumpulan kata-kata yang dapat diklasifikasi berdasarkan maknanya. Untuk mengekstrasi makna dari sebuah bahasa manusia, sebagian besar teknik NLP menggunakan metode *machine learning* [34].

2.9 Crawling (Tweet Harvest)

Crawling, atau penjelajahan web, adalah proses otomatis yang dilakukan oleh program komputer (*crawler*) untuk mengunduh dan menjelajahi halaman web

[35]. *Crawler* mengikuti tautan dari satu halaman ke halaman lain, mengunduh konten halaman, dan kemudian menambahkan tautan yang ditemukan ke daftar halaman yang akan dikunjungi selanjutnya. Proses ini berulang hingga crawler mencapai batas yang ditentukan, seperti jumlah halaman maksimum yang ingin dijelajahi atau waktu maksimum yang ingin dihabiskan.

Tweet harvest merupakan *web crawler* yang digunakan pada aplikasi analisis media sosial adalah, *tweet harvest* yang mendapat data platform twitter. Proses ini melibatkan pemindaian dan pengambilan informasi secara sistematis dari berbagai akun dan cuitan di Twitter, sehingga memungkinkan aplikasi untuk menganalisis tren, sentimen, dan berbagai metrik penting lainnya dari data yang telah dikumpulkan. *Tweet harvest* bekerja dengan cara mengakses API Twitter atau menggunakan teknik *scraping* untuk memperoleh data yang diperlukan guna mendukung analisis media sosial yang lebih mendalam dan komprehensif.

2.10 Text Preprocessing

Teks *Preprocessing* merupakan tahapan pra-pemrosesan meningkatkan kualitas dan akurasi analisis terhadap *tweet* yang menjadi fokus penelitian atau pemantauan. Tahapan preprocessing penting dilakukan dalam menyusun teks yang tidak terstruktur serta menjaga kata kunci yang dapat berguna untuk mewakili topik pembicaraan tertentu [36]. Dalam tahap *preprocessing*, terdapat beberapa metode yang dipakai diantaranya yaitu :

2.10.1 *Casefolding*

Casefolding merupakan proses yang mengubah format huruf dalam suatu teks menjadi huruf kecil atau *lowercase* [36]. Hal ini bertujuan untuk membuat format huruf menjadi seragam dalam teks tweet yang beragam, sehingga memudahkan dalam menjaga konsistensi dalam pengolahan teks tanpa memperhitungkan variasi kapitalisasi. Sebagai contoh, kata "Real Madrid" akan diubah menjadi "real madrid" melalui proses *casefolding*.

2.10.2 *Cleansing*

Cleansing merupakan pada langkah kedua dalam pemrosesan data di mana data yang tidak valid, tidak lengkap, atau tidak relevan diidentifikasi dan dihapus dari *dataset* [36]. Dalam tahap ini, dilakukan penanganan terhadap karakter non-alfabet seperti *mention,* hashtag, link, tanda baca, dan spasi ekstra untuk mengurangi gangguan pada dokumen.

2.10.3 *Tokenization*

Tokenization atau tokenisasi merupakan pemanggal kalimat *tweet* menjadi bagian-bagian atau kata – kata yang lebih kecil [36]. Pada dasarnya, tokenisasi adalah langkah penting dalam analisis teks yang memungkinkan pemahaman yang lebih baik terhadap struktur dan makna teks yang panjang seperti *tweets*.

2.10.4 *Normalization*

Normalization atau Normalisasi kata adalah memperbaiki kata yang disingkat agar menjadi lebih dipahami dan menghilangkan *noise* yang di peroleh dari *tweet* [36]. Kata-kata yang diperbaiki merupakan hasil pencocokan variasi kata dari *dataset* KBBA (Kamus Besar Bahasa Indonesia) yang menginterpretasikan singkatan yang biasa digunakan dalam *tweet*. Contohnya: iy menjadi iya.

2.10.5 *Stemming*

Stemming adalah proses dalam pemrosesan bahasa alami yang digunakan untuk menghilangkan infleksi dan afiks dari kata, sehingga hanya meninggalkan akar kata atau "stem" [36]. Tujuan utamanya adalah untuk menyatukan kata-kata yang berbeda bentuk tetapi memiliki akar kata yang sama ke dalam bentuk dasarnya, sehingga mempermudah analisis dan pengolahan teks. Contohnya, kata "berlari", "berlari-lah", dan "berlari-nya" akan diubah menjadi "lari". Teknik *Stemming* ini sering digunakan dalam pengindeksan dan pencarian teks, serta dalam analisis teks untuk ekstraksi fitur atau klasifikasi dokumen.

2.10.6 *Stopword Removal*

Stopword removal adalah proses penghapusan kata-kata umum yang tidak memberikan informasi penting dalam analisis teks [36]. Stopwords merupakan

kata-kata yang sering muncul dalam dokumen teks, seperti "dan", "atau", "adalah", "yang", dan sebagainya. Kata-kata ini dianggap tidak memiliki makna signifikan dalam konteks analisis teks, terutama dalam tugas-tugas seperti pemrosesan bahasa alami (Natural Language Processing - NLP), text mining, dan machine learning. Tujuan dari stopword removal adalah untuk mengurangi dimensi data teks dengan menghapus kata-kata yang tidak relevan sehingga algoritma dapat lebih fokus pada kata-kata yang lebih bermakna dan penting. Stopword removal dapat dilakukan menggunakan daftar kata-kata umum yang sudah ditentukan sebelumnya (stopword list) atau menggunakan metode yang lebih adaptif seperti thresholding berdasarkan frekuensi kemunculan kata.

2.10.7 *Indexing*

Indexing adalah proses pembuatan struktur data yang memungkinkan akses cepat terhadap dokumen atau bagian dari dokumen berdasarkan kata-kata kunci [36]. Dalam konteks sistem pencarian informasi (Information Retrieval - IR), indexing sangat penting untuk mempercepat proses pencarian dokumen yang relevan. Indexing melibatkan pemetaan antara kata kunci dan lokasi di mana kata tersebut muncul dalam dokumen atau kumpulan dokumen. Salah satu teknik indexing yang populer adalah inverted index, di mana setiap kata dikaitkan dengan daftar dokumen yang mengandung kata tersebut. Inverted index memungkinkan pencarian kata kunci dalam skala besar menjadi lebih efisien karena sistem hanya perlu mencari di dalam indeks, bukan di seluruh dokumen [36].

2.10.8 *Padding*

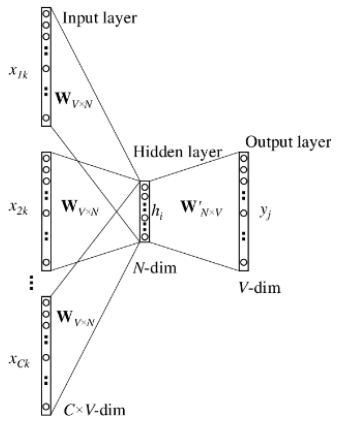
Padding adalah teknik yang digunakan dalam pemrosesan data teks, terutama dalam konteks model machine learning dan deep learning, untuk memastikan bahwa semua *input* memiliki panjang yang sama [36]. Dalam tugas-tugas seperti klasifikasi teks atau pemodelan urutan (sequence modeling), *input* teks sering kali memiliki panjang yang bervariasi. Hal ini bisa menjadi masalah karena banyak algoritma machine learning membutuhkan *input* dengan dimensi tetap.

2.11 Classification

Classification atau klasifikasi adalah proses pengelompokan objek-objek ke dalam kelas-kelas yang telah ditentukan berdasarkan pada atribut-atribut atau fitur-fitur yang diamati dari objek tersebut [37]. Tujuan utama dari klasifikasi adalah untuk menemukan pola yang tersembunyi di dalam data dan untuk menghasilkan prediksi tentang kelas atau label dari objek-objek baru, berdasarkan pada pengalaman dari data yang telah diamati sebelumnya.

2.12 Word Embedding

Word embedding adalah teknik pembelajaran yang menghasilkan representasi kata dalam bentuk vektor kontinu di ruang dimensi rendah. Umumnya, teknik ini menggunakan jaringan saraf tiruan (JST). Salah satu metode *Word embedding* yang terkenal adalah *Word2vec* [38]. *Word2vec* menggunakan jaringan saraf tiruan untuk menghasilkan vektor kata dengan kemiripan semantik yang tinggi. Dua arsitektur yang digunakan adalah continuous bag-of-words (CBOW) dan Skip-gram. Penelitian ini menggunakan model Skip-gram, yang bertujuan memprediksi kata-kata di sekitar kata yang diberikan (conteks word). Dengan menggunakan ukuran window, model dapat menentukan kata-kata target dengan memperhitungkan jumlah kata yang diamati sebelum dan sesudah konteks word. Berikut adalah visualisasi jaringan saraf dari algoritma skip-gram [38].



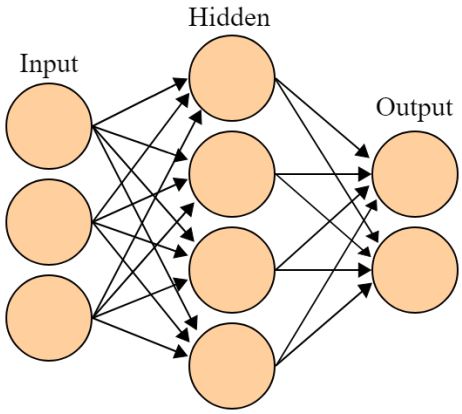
Gambar 2.8 *Word embedding* Skip Diagram

2.13 Neural Network

Neural Network adalah merupakan bidang penelitian yang mencoba meniru cara pemrosesan informasi dan penghasilan hasil oleh otak manusia. Ini dilakukan dengan mengadopsi prinsip pembelajaran yang terinspirasi oleh fungsi sistem saraf manusia dan organisme biologis lainnya. Dikenal juga sebagai jaringan saraf tiruan, *Neural Network* adalah salah satu metode pembelajaran mesin yang sangat terkenal. *Neuron*, sebagai sel khusus yang membentuk sistem saraf, memiliki akson dan dendrit yang bertindak sebagai mekanisme penghubung antar *neuron* [39].

2.13.1 Arsitektur Neural Network

Neural Network pada dasarnya mirip dengan metode lain yang menghubungkan variabel *input* dengan variabel *output*. Namun, *Neural Network* memiliki perbedaan utama yaitu adanya satu atau lebih lapisan tersembunyi yang menghubungkan lapisan *input* dengan lapisan *output*. Lapisan tersembunyi ini berfungsi untuk mengubah data *input* menggunakan fungsi aktivasi sebelum mencapai lapisan *output*. Struktur *Neural Network* ini dapat dilihat pada gambar . Lapisan-lapisan yang menyusun *Neural Network* terbagi menjadi tiga bagian: lapisan *input*, lapisan tersembunyi (*Hidden*), dan lapisan *output* [39].



Gambar 2.9 Struktur Dasar *Neural Network*

2.13.1.1 Lapisan Masukan (*Input*)

Neuron di dalam lapisan *input* disebut *neuron input*. *Neuron input* menerima *input* dari luar, *input* yang diberikan merupakan penggambaran suatu permasalahan.

2.13.1.2 Lapisan Tersembunyi (*Hidden*)

Neuron di dalam lapisan tersembunyi disebut *neuron* tersembunyi. *Output* dari lapisan ini tidak dapat diamati secara langsung.

2.13.1.3 Lapisan Keluaran (*Output*)

Neuron di dalam lapisan *output* disebut *neuron output*. Keluaran dari lapisan ini merupakan hasil dari *Neural Network* terhadap suatu permasalahan.

2.13.2 Fungsi Aktivasi

Fungsi merupakan aktivasi menentukan *output* dari sebuah unit dengan mengubah sinyal *input* menjadi sinyal *output* yang kemudian dikirim ke unit lain [40].Fungsi ini berperan untuk mengaktifkan *Neural Network* (NN), sehingga setiap *neuron* dalam jaringan dapat diaktifkan. Ada berbagai fungsi aktivasi yang dapat digunakan untuk mengaktifkan NN, yaitu sebagai berikut :

2.13.2.1 Fungsi Aktivasi *Sigmoid* ($\boldsymbol{\sigma}$)

Fungsi *sigmoid* adalah jenis fungsi aktivasi yang sering digunakan dalam *Neural Network*. Fungsi ini mengubah nilai *input* menjadi *output* dalam rentang 0 hingga 1, membuatnya sangat berguna untuk model yang perlu menghasilkan probabilitas [41]. Fungsi *sigmoid* memiliki bentuk S dan didefinisikan sebagai:

$$g(x) = \frac{1}{1 + e^{x}} \qquad 2.1$$

Keterangan :

- g(x) : *output* dari fungsi *sigmoid*
- x : *input* ke fungsi
- e : basis dari logaritma natural (se*kitar 2.7828)*

*Fungs*i *sigmoid* melakukan transformasi nilai dari *input* x ke dalam rentang antara 0 dan 1. Ini dilakukan dengan menggunakan fungsi matematika yang memiliki bentuk kurva S. Saat nilai *input* x meningkat, *output* g(x) mendekati 1, sementara ketika nilai *input* x menurun, *output* g(x) mendekati 0. Hal ini membuatnya berguna dalam menghasilkan *output* yang dapat diinterpretasikan sebagai probabilitas atau dalam memetakan nilai-nilai yang beragam ke dalam rentang yang lebih terbatas [41].

2.13.2.2 Fungsi Aktivasi *Sigmoid* (*Tanh*)

Fungsi *Tanh* (tangen hiperbolik) adalah fungsi aktivasi yang sering digunakan dalam jaringan saraf. Fungsi ini memetakan nilai *input* menjadi nilai antara -1 dan 1 [41]. Rumus matematika untuk fungsi *Tanh* adalah:

$$Tanh(\mathrm{x}) = \frac{e^{x} - e^{-x}}{e^{x} - e^{-x}} \qquad \textbf{2.2}$$

Keterangan :

- *Tanh*(x) : *output* dari fungsi tangen hiperbolik
- x : *input* ke fungsi
- e : basis dari logaritma natural (se*kitar 2.7828)*

Dalam konteks jaringan saraf, fungsi *Tanh* membantu dalam normalisasi *output* ke rentang -1 hingga 1, yang sering mempercepat konvergensi selama pelatihan. Dibandingkan dengan fungsi *sigmoid*, *Tanh* memiliki nilai *output* yang terpusat di se*kitar nol, yang dap*at membantu mengurangi masalah *vanishing gradient* (gradien yang menghilang) saat melatih jaringan saraf yang dalam.

2.13.2.3 Fungsi Aktivasi Softmax ($Softmax$)

Softmax adalah fungsi aktivasi yang sering digunakan dalam jaringan saraf untuk tugas klasifikasi, terutama pada lapisan *output* dari model klasifikasi dengan beberapa kelas. Fungsi ini mengubah sekumpulan nilai menjadi distribusi probabilitas [41]. Rumus matematika untuk fungsi *Softmax* adalah:

$$Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \qquad \textbf{2.3}$$

Keterangan :

- $Softmax(z_i)$: *output* dari fungsi *Softmax* untuk elemen ke-i dari vektor *input*.
- z_i : elemen ke-i dari vektor *input z*
- K : jumlah total elemen dalam vektor *input* z
- e : basis dari logaritma natural (se*kitar 2.7828)*

*Fung*si *Softmax* memiliki beberapa sifat penting yang membuatnya berguna dalam pembelajaran mesin. Pertama, *output* dari fungsi *Softmax* adalah sekumpulan nilai yang berjumlah 1, sehingga dapat diinterpretasikan sebagai probabilitas. *Softmax* memperbesar perbedaan antara nilai *input* yang besar dan kecil, yang dapat membantu dalam membedakan kelas-kelas dengan lebih jelas.

Fungsi ini sangat penting dalam jaringan saraf karena memungkinkan model untuk menghasilkan prediksi probabilistik untuk setiap kelas dalam tugas klasifikasi multi-kelas. Dalam konteks jaringan saraf tiruan, fungsi *Softmax* sering digunakan pada lapisan *output* untuk memastikan bahwa hasil prediksi dapat ditafsirkan sebagai probabilitas yang valid.

2.13.2.4 Fungsi Aktivasi ReLU ($ReLU$)

Fungsi ReLU atau *Rectified Linear Unit*, telah menjadi salah satu fungsi aktivasi yang paling banyak digunakan dalam jaringan saraf tiruan modern. Fungsi ini pertama kali diperkenalkan oleh Hahnloser et al. (2000) dan kemudian dipopulerkan oleh Nair dan Hinton (2010). Fungsi ReLU didefinisikan sebagai:

$$\mathrm{ReLU}(x) = \max(0, x) \quad \textbf{2.4}$$

Keterangan :

- ReLU(x) : ReLU(x)
- x : *input* ke fungsi

Fungsi ReLU akan mengembalikan nilai x jika x lebih besar dari nol, dan akan mengembalikan nol jika x kurang dari atau sama dengan nol. Fungsi ini sederhana namun sangat efektif dalam jaringan saraf tiruan karena membantu mengatasi masalah vanishing gradient yang sering terjadi pada fungsi aktivasi lain seperti *sigmoid* dan *Tanh*[41].

2.13.2.5 Fungsi Aktivasi Glorot Uniform ($Limit$)

Inisialisasi Glorot Uniform adalah salah satu metode untuk menginisialisasi bobot pada jaringan saraf tiruan, yang bertujuan untuk menjaga varians *output* di

setiap lapisan tetap konstan, sehingga memungkinkan gradien mengalir lebih efektif selama proses backpropagation [42]. Metode ini pertama kali diperkenalkan oleh Xavier Glorot dan Yoshua Bengio pada tahun 2010. Metode ini menggunakan distribusi uniform untuk menginisialisasi bobot, yang didefinisikan sebagai:

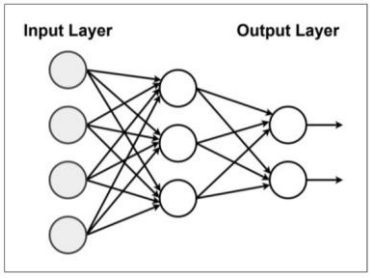
$$-Limit, Limit = \left(-\sqrt{\frac{6}{fan\_in + fan\_out}}, \sqrt{\frac{6}{fan\_in + fan\_out}} \right) \quad \textbf{2.5}$$

Keterangan :

- **fan_in**: Jumlah unit *input* (neuron) yang masuk ke dalam lapisan tersebut.
- **fan_out**: Jumlah unit *output* (neuron) yang keluar dari lapisan tersebut.

2.14 Artificial Neural Network (ANN)

Artificial Neural Network (ANN) adalah kumpulan unit pemrosesan sederhana yang dikenal sebagai *node*, yang saling terhubung dan bekerja dengan cara yang meniru sel saraf (*neuron*) di otak. Kemampuan jaringan ini dalam memproses informasi bergantung pada kekuatan hubungan antar *node*, yang disebut bobot, yang disesuaikan melalui proses adaptasi atau pembelajaran berdasarkan pola yang ditemukan dalam data latih [43]. Berikut pada Gambar 2.10 merupakan cara kerja dari Artifical *Neural Network* (ANN).

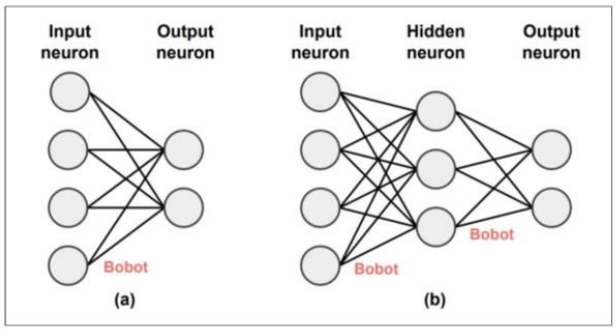


Gambar 2.10 Artifical *Neural Network* [43]**.**

Pada sel saraf manusia, koneksi sinaptik bisa diubah di bawah beberapa kondisi, memungkinkan *neuron* untuk berubah perilakunya sesuai dengan *input*nya. Di dalam sel saraf buatan, proses yang sama bisa terjadi dengan mengubah nilai bobot koneksi. Ketika mengolah informasi, pengetahuan disimpan dalam bobot yang berubah sesuai dengan pola *input* yang diamati. Dalam metode yang disebut *supervised*, jaringan diberikan pola *input* untuk direspons, dan hasilnya dibandingkan dengan pola *output* yang diharapkan [43].

Setelah menjalani proses pelatihan, jaringan saraf akan mampu menangani *input* dengan pola baru yang belum pernah dikenali sebelumnya. Kemampuan adaptasi jaringan ini disebabkan oleh kemampuannya untuk memodifikasi bobot koneksi antara *neuron*, yang memungkinkan proses pembelajaran yang dinamis dan fleksibel. Dengan demikian, jaringan dapat terus mengembangkan pemahaman dan respons terhadap lingkungannya seiring waktu, memperluas kapabilitasnya dalam mengenali dan memproses berbagai pola *input* [43].

Pada 1957, Frank Rosenblatt mengembangkan model linear revolusioner untuk klasifikasi biner yang disebut "*perseptron*", memperkenalkan dasar dari jaringan saraf tiruan [43]. Meskipun sederhana, ia mampu mempelajari pola-pola dari data *input* dan mengklasifikasikannya. Namun, kelemahan ditemukan, terutama dalam menangani masalah nonlinear seperti logika AND dan OR. Ini mendorong pengembangan *Multi-Layer Perceptron* (MLP) yang menggunakan banyak lapisan untuk menangani masalah yang lebih kompleks.

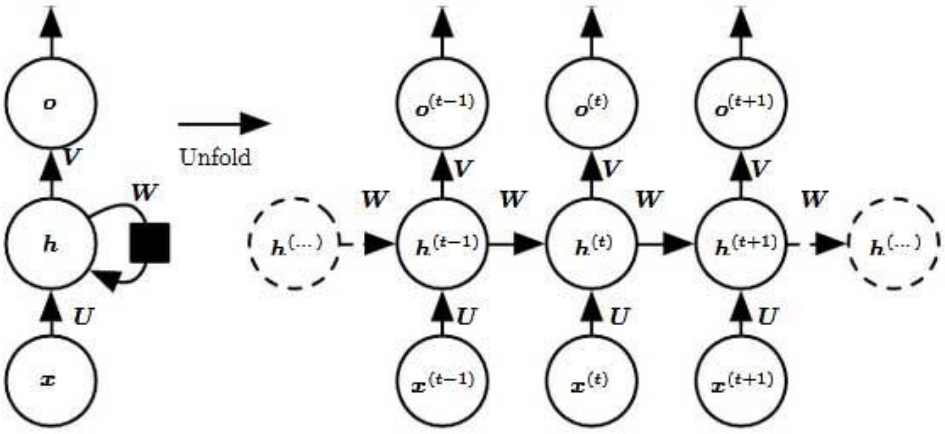


Gambar 2.11 Arsitektur Single-*Layer* Perceptron[44]

MLP adalah tipe jaringan saraf yang terdiri dari satu lapisan *input*, satu atau beberapa lapisan tersembunyi, dan satu lapisan *output*. Setiap lapisan ini terdiri dari satu atau lebih *neuron*. *Neuron* dalam MLP serupa dengan single *layer* perceptron, tetapi jenis fungsi aktivasi yang digunakan dapat bervariasi tergantung pada tujuan spesifik dari setiap lapisan. Perbedaan antara arsitektur *single layer* dan *multi layer* perceptron terletak pada jumlah lapisannya.

2.15 Recurrent Neural Network (RNN)

Recurrent Neural Network (RNN) adalah jenis jaringan saraf tiruan yang dirancang untuk memproses data berurutan, seperti teks, sinyal waktu, atau video. Berbeda dengan jaringan saraf feed*forward*, RNN memiliki koneksi melingkar yang memungkinkan informasi untuk dipertahankan dari satu langkah waktu ke langkah berikutnya. Hal ini membuat RNN sangat efektif untuk tugas-tugas yang melibatkan data sekuensial. *Recurrent Neural Network* (RNN) memiliki arsitektur yang dirancang untuk memproses data sekuensial dengan mempertahankan informasi dari satu langkah waktu ke langkah waktu berikutnya melalui koneksi berulang. Berikut adalah penjelasan lebih rinci mengenai bagaimana RNN bekerja :

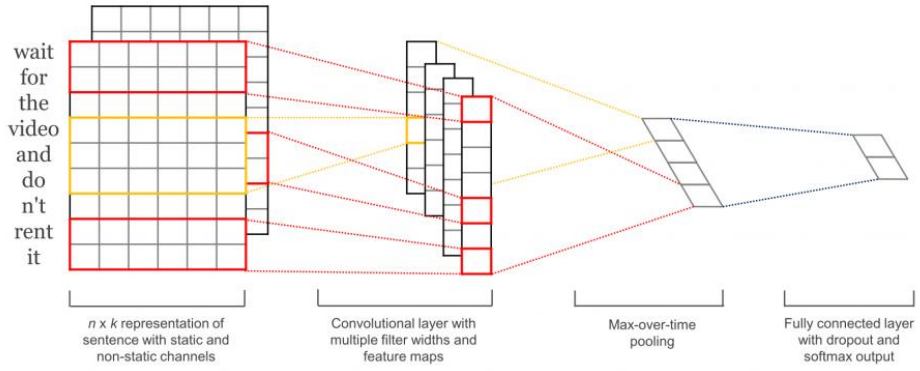


Gambar 2.12 Bagaimana RNN Bekerja

2.16 Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN) awalnya dikembangkan untuk pengenalan pola dalam gambar, namun, arsitektur ini juga telah terbukti efektif dalam tugas-tugas terkait pemrosesan bahasa alami (Natural Language Processing, NLP), termasuk klasifikasi teks [45]. CNN dapat menangkap fitur spasial dalam data teks, seperti urutan kata atau frasa, yang berguna untuk tugas-tugas seperti klasifikasi emosi.

Cara kerja CNN mencakup beberapa langkah penting, mulai dari penerapan filter untuk mendeteksi fitur-fitur dasar, pengurangan dimensi spasial untuk mengurangi kompleksitas, hingga proses klasifikasi akhir menggunakan jaringan saraf yang lebih tradisional. Setiap lapisan dalam CNN memainkan peran spesifik dalam mengidentifikasi pola yang semakin kompleks di setiap tahap jaringan, memungkinkan CNN untuk mengenali objek atau pola bahkan dalam gambar yang sangat kompleks. Bagian ini akan menguraikan secara mendetail bagaimana CNN melakukan operasi ini dan bagaimana setiap komponen berkontribusi terhadap keberhasilan CNN dalam tugas-tugas pengenalan pola.



Gambar 2.13 Cara Kerja CNN

2.16.1 Lapisan Konvolusi (*Convolutional Layer*)

Convolutional Layer adalah lapisan utama dalam CNN yang bertugas mengekstraksi fitur-fitur dari data *input*. Lapisan ini menggunakan filter (kernel) yang digeser di sepanjang data *input* untuk mendeteksi fitur lokal seperti tepi, sudut,

dan tekstur [45]. Pada lapisan konvolusi, filter diterapkan secara lokal pada bagian-bagian kecil dari *input* untuk menghasilkan feature map. Setiap filter memproses data *input* dengan cara menggeser secara spasial di sepanjang *input* dan menghitung produk titik antara filter dan bagian yang tumpang tindih dari *input*. Filter dapat dilatih untuk mengenali pola spesifik dalam data.

$$Q_{i,j} = \sum_{k=1}^{40} I_{i,k} \cdot K_{k,j} + B_j \qquad \textbf{2.6}$$

Keterangan :

- $I_{i,k}$ adalah *input* pada posisi *i,k*
- $K_{k,j}$ adalah nilai kernel (filter) pada posisi *k,j*
- B_j adalah bias,
- $Q_{i,j}$ adalah *output* pada peta fitur (feature map).

2.16.2 Lapisan Pooling (*Pooling Layer*)

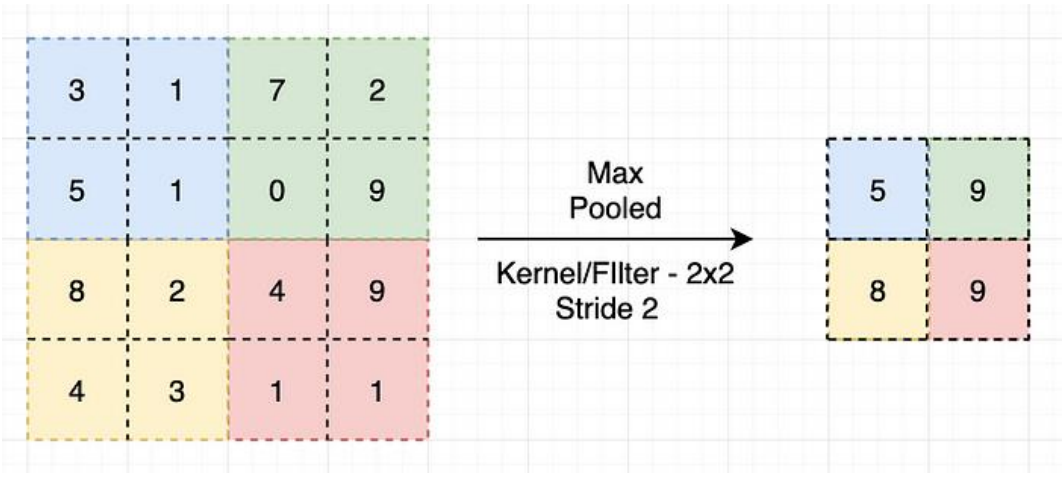
Pooling Layer digunakan untuk mengurangi dimensi spasial dari peta fitur, sekaligus mempertahankan fitur yang paling signifikan. Pengurangan dimensi ini membantu mengurangi jumlah parameter dan kompleksitas komputasi, serta meningkatkan generalisasi model [45]. Pooling dilakukan dengan menerapkan fungsi agregasi seperti Max Pooling pada bagian-bagian kecil dari peta fitur. Max Pooling, misalnya, memilih nilai maksimum dalam setiap jendela pooling, mengabaikan nilai lainnya.

$$P(i,j) = \max 1_{m,n \in k \times k} Q(i+m, j+n) \qquad \textbf{2.7}$$

Keterangan :

- $P(i,j)$ adalah nilai pada peta fitur setelah pooling
- $Q(i+m,j+n)$ adalah nilai pada peta fitur sebelum pooling.

Pada Gambar 2.14 merupakan ilustrasi dari pooling layer pada CNN yang merubah kata dari ukuran yang besar menjadi setengahnya :



Gambar 2.14 Cara Kerja Pooling layer

2.16.3 Lapisan Terhubung Penuh (*Fully Connected Layer*)

Lapisan Fully Connected menghubungkan semua neuron dari lapisan sebelumnya ke setiap neuron di lapisan ini. Lapisan ini berfungsi untuk melakukan klasifikasi berdasarkan fitur yang diekstraksi oleh lapisan konvolusi dan pooling [45]. Fitur-fitur yang telah diratakan (flattened) dari peta fitur sebelumnya dihubungkan secara penuh ke setiap neuron dalam lapisan ini. Setiap neuron menghitung kombinasi linear dari semua *input*nya, yang kemudian digunakan untuk menentukan kelas *output*. Operasi pada lapisan fully connected dapat dirumuskan sebagai:

$$J_i = \sum_{j=1}^{n} i_j \times w_{ij} + b_i \qquad \textbf{2.8}$$

Keterangan :

- J_i adalah *output* dari neuron ke-*i,*
- i_j adalah *input* dari neuron ke-*j,*
- w_{ij} adalah bobot antara neuron ke-*j* dan neuron ke-*i,*
- b_i adalah bias yang ditambahkan pada neuron ke-*i,*

2.16.4 Lapisan Keluaran (*Output Layer*)

Lapisan *output* menghasilkan prediksi akhir dari CNN berdasarkan perhitungan yang dilakukan di lapisan sebelumnya. Jika CNN digunakan untuk klasifikasi, biasanya lapisan ini diikuti oleh fungsi aktivasi Softmax untuk menghasilkan distribusi probabilitas.

2.17 Bidirectional Long Short-Term Memory (BI-LSTM)

Bidirectional Long Short-Term Memory (BI-LSTM) adalah pengembangan dari Long Short-Term Memory (LSTM) yang memungkinkan model untuk belajar dari konteks kedua arah dalam urutan data [46]. Struktur ini sangat bermanfaat dalam pemrosesan bahasa alami (NLP) dan tugas-tugas lain yang melibatkan urutan data, di mana konteks sebelumnya dan selanjutnya sama pentingnya dalam menentukan makna. BI-LSTM menggunakan dua LSTM: satu untuk mengakses urutan dalam arah maju (*forward*) dan yang lain untuk arah mundur (*backward*). Alasan menggunakan arah *forward* dan *backward [47]*:

1) **Konteks Lebih Kaya**: Banyak tugas NLP memerlukan pemahaman konteks dari kata-kata yang muncul sebelum dan sesudah kata yang sedang diproses. Dengan memproses urutan dalam dua arah, BI-LSTM dapat menangkap informasi dari seluruh urutan, yang membantu dalam membuat keputusan lebih akurat.

2) **Mengatasi Ambiguitas**: Informasi dari arah yang berlawanan dapat membantu mengklarifikasi makna kata-kata yang mungkin ambigu jika hanya dilihat dari satu arah saja.

3) **Ketergantungan Jangka Panjang**: Dalam banyak kasus, kata-kata di akhir urutan mungkin bergantung pada kata-kata di awal urutan, dan sebaliknya. BI-LSTM dapat mempertahankan informasi ini dengan menggabungkan kedua arah, sehingga lebih baik dalam menangani ketergantungan jangka panjang.

Dengan menggunakan kedua arah, BI-LSTM mengintegrasikan informasi dari masa lalu dan masa depan untuk memberikan pemahaman yang lebih lengkap terhadap urutan data, yang sangat penting dalam tugas-tugas seperti terjemahan bahasa, analisis sentimen, dan pengenalan suara.

2.17.1 Struktur *Long Term Short Memory* (LSTM)

LSTM adalah jenis jaringan saraf berulang (RNN) yang mampu mempertahankan informasi dalam jangka waktu yang panjang dan mengatasi masalah *vanishing gradient* yang sering ditemui dalam RNN konvensional. LSTM ini terdiri dari beberapa komponen utama, termasuk tiga jenis gerbang yang mengatur aliran informasi. Pada tahapan ini penjelasan rinci dari setiap tahapan LSTM:

2.17.1.1 Gerbang Lupa (*Forget Gate*)

Forget Gate memutuskan informasi mana yang harus dilupakan dari sel memori berdasarkan *input* baru dan status sebelumnya. Alasan di balik *Forget Gate* adalah untuk memungkinkan model membuang informasi yang tidak lagi relevan, sehingga hanya informasi penting yang diteruskan ke waktu berikutnya. Ini penting untuk mencegah sel memori menjadi penuh dengan informasi yang tidak relevan, yang dapat mengganggu pembelajaran.

$$f_t = \sigma\left(W_f \cdot x_t + U_f \cdot h_{t-1} + b_f\right) \quad \textbf{2.9}$$

Keterangan :

- f_t adalah *output* dari *Forget Gate*.
- W_f adalah bobot untuk *input* x_t
- U_f adalah bobot untuk *hidden state* sebelumnya h_{t-1}.
- b_f adalah bias yang ditambahkan.
- h_{t-1} adalah nilai dari *hidden state* sebelumnya.
- σ adalah fungsi *sigmoid*, yang mengeluarkan nilai antara 0 dan 1, menentukan seberapa banyak informasi yang akan dilupakan.

2.17.1.2 Gerbang Masukan (*Input gate*)

Input gate menentukan informasi baru apa yang akan disimpan dalam sel memori. *Input gate* penting untuk memperbarui memori dengan informasi yang relevan dari *input* baru, sementara mengabaikan data yang kurang relevan.

$$I_t = \sigma(W_i \cdot x_t + U_i \cdot h_{t-1} + b_i) \quad \textbf{2.10}$$

Keterangan :

- I_t adalah *output* dari *input gate*.
- W_i dan U_i adalah bobot untuk *input* dan *hidden state*.
- b_i adalah bias untuk *input gate*.
- h_{t-1} adalah nilai dari *hidden state* sebelumnya.
- σ adalah fungsi *sigmoid*, yang mengeluarkan nilai antara 0 dan 1, menentukan seberapa banyak informasi yang akan dilupakan.

2.17.1.3 Candidate Cell

Candidate cell state ($\tilde{c}_t$) adalah nilai sementara yang dihasilkan oleh LSTM untuk memperbarui cell state ($\tilde{c}_t$). Ini mencerminkan informasi baru yang mungkin

akan disimpan dalam cell state berdasarkan *input* saat ini dan *hidden state* sebelumnya. Candidate cell state dihasilkan setelah *input* melalui *input gate* dan diolah melalui fungsi aktivasi tangens hiperbolik.

$$\widetilde{c_t} = \tanh(W_c \cdot x_t + U_c \cdot h_{t-1} + b_c) \quad \textbf{2.11}$$

Keterangan :

- I_t adalah *output* dari Candidate cell state.
- W_c dan U_c adalah bobot untuk *input* dan *hidden state*.
- h_{t-1} adalah nilai dari *hidden state* sebelumnya.
- b_c adalah bias untuk Candidate cell state.
- Tanh adalah fungsi aktivasi tangens hiperbolik.

2.17.1.4 Pembaruan Status Memori (*Cell State Update*)

Setelah *Forget Gate* dan *input gate* bekerja, status memori diperbarui dengan menggabungkan informasi lama yang relevan dan informasi baru yang penting. Mempertahankan informasi yang penting dari waktu ke waktu sambil memperbarui dengan informasi terbaru, memungkinkan model untuk belajar lebih efektif.

$$c_t = f_t \cdot c_{t-1} + I_t \cdot \widetilde{c_t} \quad \textbf{2.12}$$

Keterangan :

- c_t adalah status memori saat ini,
- c_{t-1} adalah status memori dari waktu sebelumnya.
- f_t adalah nilai dari *Forget Gate*
- I_t adalah nilai dari *input gate*

- $\tilde{c_t}$ adalah nilai candidate cell

2.17.1.5 Gerbang Keluaran (*Output Gate*)

Output gate memutuskan bagian mana dari status memori yang akan digunakan untuk menghitung *output* LSTM. Ini memastikan bahwa hanya informasi yang relevan digunakan untuk menentukan *output* pada setiap langkah waktu, mendukung prediksi yang lebih akurat.

$$O_t = \sigma(W_o \cdot x_t + U_o \cdot h_{t-1} + b_o) \quad \textbf{2.13}$$

Keterangan :

- I_o adalah *output* dari *input gate*.
- W_o dan U_o adalah bobot untuk *input* dan *hidden state*.
- b_o adalah bias untuk *input gate*.
- h_{t-1} adalah nilai dari *hidden state* sebelumnya.
- σ adalah fungsi *sigmoid*, yang mengeluarkan nilai antara 0 dan 1, menentukan seberapa banyak informasi yang akan dilupakan.

2.17.1.6 *Hidden state* Update

Hidden state (h_t) adalah *output* dari LSTM pada waktu (*t)*, yang juga digunakan sebagai *input* untuk langkah berikutnya dalam urutan. *Hidden state* menyimpan informasi yang diolah oleh LSTM dan digunakan untuk menghasilkan prediksi atau keputusan pada waktu tersebut. *Hidden state* memungkinkan LSTM untuk mempertahankan informasi penting dari waktu ke waktu dan menggunakannya untuk membuat keputusan pada setiap langkah. Ini membantu LSTM untuk menangkap pola yang lebih kompleks dalam data sekuensial.

$$h_t = O_t \cdot Tanh\,(c_t) \quad \textbf{2.14}$$

Keterangan :

- h_t adalah nilai dari *hidden state*
- c_t adalah nilai dari cell state update
- O_t adalah nilai dari *output* gate
- Tanh adalah fungsi aktivasi tangens hiperbolik.

2.18 Model Evaluation

Pengujian model adalah tahap penting dalam siklus hidup pengembangan model machine learning, yang bertujuan untuk mengevaluasi kinerja model dengan metrik tertentu. Dalam konteks klasifikasi, metrik yang sering digunakan meliputi Akurasi, Presisi, Recall, dan F1-Score. Metrik-metrik ini memberikan wawasan tentang seberapa baik model bekerja pada data yang belum pernah dilihat sebelumnya.

2.18.1 Akurasi (*Accuracy*)

Akurasi adalah proporsi prediksi yang benar terhadap keseluruhan prediksi. Ini adalah metrik dasar yang sering digunakan untuk menilai kinerja model, terutama dalam kasus di mana kelas seimbang.

$$Akurasi = \frac{Jumlah\ Prediksi\ Benar}{Jumlah\ Prediksi\ Salah} \quad \textbf{2.15}$$

Akurasi yang tinggi menunjukkan bahwa model sering membuat prediksi yang benar. Namun, akurasi bisa menyesatkan dalam kasus data yang tidak seimbang, di mana model bisa mendapatkan akurasi tinggi dengan hanya memprediksi kelas mayoritas.

2.18.2 Presisi (*Precision*)

Presisi mengukur seberapa tepat prediksi model untuk kelas positif. Ini adalah proporsi prediksi positif yang benar dari semua prediksi positif.

$$Presisi = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Positive\ (FP)} \quad \textbf{2.16}$$

Keterangan:

- *True Positive* (TP): Jumlah data yang diklasifikasikan dengan benar sebagai positif oleh model.
- *False Positive* (FP): Jumlah data yang diklasifikasikan secara salah sebagai positif oleh model padahal sebenarnya negatif.

Presisi yang tinggi berarti ketika model memprediksi suatu instance sebagai positif, prediksi tersebut sering benar. Ini penting dalam situasi di mana kesalahan positif palsu harus diminimalkan, seperti dalam diagnosa medis.

2.18.3 Recall (Sensitivitas)

Recall, juga dikenal sebagai sensitivitas, adalah proporsi kasus positif yang benar-benar terdeteksi oleh model dari semua kasus positif sebenarnya.

$$Recall = \frac{True\ Positive\ (TP)}{True\ Positive\ (TP) + False\ Negative\ (FN)} \quad \textbf{2.16}$$

Keterangan:

- *True Positive* (TP): Jumlah data yang diklasifikasikan dengan benar sebagai positif oleh model.
- *False Negative* (FN): Jumlah data yang secara salah diklasifikasikan sebagai negatif oleh model padahal sebenarnya positif.

Recall yang tinggi menunjukkan bahwa model dapat mendeteksi sebagian besar kasus positif. Ini penting dalam situasi di mana terlewatnya positif palsu harus diminimalkan, seperti dalam deteksi penipuan.

2.18.4 F1-Score

F1-Score adalah metrik gabungan yang mengharmonisasikan presisi dan recall. Ini adalah rata-rata harmonis dari presisi dan recall, memberikan metrik yang lebih seimbang ketika keduanya perlu dipertimbangkan secara bersama-sama.

$$F1\ Score\ =\ 2\ x\ \frac{Presisi\ x\ Recall}{Presisi\ +\ Recall} \qquad \textbf{2.17}$$

Keterangan:

- Presisi (*Precision*): Proporsi prediksi positif yang benar dari keseluruhan prediksi positif yang dibuat oleh model.
- Recall (Daya Ingat/Sensitivitas): Proporsi total instance positif yang berhasil diidentifikasi dengan benar oleh model.

F1-Score yang tinggi menunjukkan bahwa model memiliki keseimbangan yang baik antara presisi dan recall, dan sangat berguna dalam skenario di mana ada trade-off antara keduanya.

2.19 Pengujian Unit

Pengujian unit adalah proses verifikasi yang dilakukan pada komponen terkecil dari perangkat lunak, yang dikenal sebagai unit, untuk memastikan bahwa setiap bagian berfungsi sebagaimana mestinya. Unit dalam konteks ini biasanya berupa fungsi, metode, atau kelas yang membentuk satu kesatuan fungsional dalam kode program. Tujuan dari pengujian unit adalah untuk memvalidasi bahwa setiap unit individual bekerja sesuai dengan spesifikasinya sebelum unit-unit tersebut digabungkan menjadi sistem yang lebih besar [48].

Pengujian unit dapat dilakukan dengan berbagai metodologi, tergantung pada bahasa pemrograman dan framework yang digunakan. Beberapa metode yang umum digunakan dalam pengujian unit meliputi:

- Pengujian *White Box*: Menguji struktur internal dari unit, dengan fokus pada jalur kode, kondisi, dan cabang yang ada dalam kode tersebut[48] .
- Pengujian *Black Box*: Menguji fungsi unit berdasarkan spesifikasi eksternalnya tanpa memperhatikan implementasi internal, dengan fokus pada input dan output yang dihasilkan[48]
- *Mocking dan Stubbing*: Penggunaan objek tiruan (mock) atau stubs untuk mengisolasi unit yang diuji dari dependensi eksternal seperti database, layanan jaringan, atau kelas lain [48].

Pengujian unit adalah fondasi yang kuat dalam proses pengembangan perangkat lunak yang berkualitas. Dengan memastikan bahwa setiap unit individu berfungsi dengan benar, pengembang dapat membangun aplikasi yang lebih andal, mudah dipelihara, dan mudah diperbaiki.