BAB II
TINJAUAN PUSTAKA DAN LANDASAN TEORI

2.1. Tinjauan Pustaka

Setelah peneliti melakukan telaah terhadap beberapa penelitian, ada beberapa penelitian yang bisa dijadikan sebagai bahan acuan dan referensi untuk digunakan oleh peneliti sehingga bisa menjadi pembanding dalam penelitian yang peneliti lakukan saat ini.

Pada penelitian [5], model *NER* dibangun dengan menggunakan pendekatan *Machine Learning* berbasis probabilitas yaitu metode *CRF* dari 8.000 *tweet* yang telah dikelompokkan menjadi *tweet* formal, *tweet* informal, dan *tweet* campuran. Memberikan hasil recall dan precision masing-masing sebesar 62% dan 87% untuk *tweet* formal, masing-masing 36% dan 90% untuk *tweet* informal, dan 60% dan 86% untuk *tweet* campuran. Pada penelitian ini tidak mencantumkan hasil *F1 score*, sehingga peneliti disini menghitung sendiri *F1 score* berdasarkan hasil precision dan recall yang dicantumkan. Maka didapatkan hasil *F1 score* untuk *tweet* formal 72%, untuk *tweet* informal 51%, dan untuk *tweet* campuran 70%. Sedangkan dalam penelitian selanjutnya model *NER* dibangun dengan menggunakan pendekatan *Machine Learning* berbasis statistik yaitu metode *Hidden Markov Model* (*HMM*) didaptkan hasil yang berbeda, sebagai berikut.

Penggunaan metode *HMM* pada penelitian [6] dalam mengembangkan sistem model *NER* pada *tweet* berbahasa Indonesia dengan penambahan fitur *POS tag*. Data pelatihan yang digunakan terdiri dari 500 *tweet* serta diberi label terlebih dahulu dengan menambahkan *POS tag* pada setiap kata yang akan dibuat dan digunakan dalam penelitian ini. Hasil percobaan dalam penelitian ini menunjukkan bahwa penambahan tag POS adalah fitur terbaik untuk pemodelan *NER* menggunakan *HMM* dan menghasilkan *F1 score* sebesar 64,06%. Ternyata hasil *F1 score* yang didapatkan lebih rendah meskipun sudah menambahkan fitur *POS tag* dibandingkan penelitian [5] yang hanya menggunakan pendekatan *Machine Learning* berbasis probabilitas. Dalam penelitian selanjutnya mengembangkan

model *NER* pada Twitter Indonesia lebih dalam menggunakan metode *Deep Learning*, sehingga didaptkan hasil sebagai berikut.

Penelitian [1] meneliti *NER* pada postingan Twitter bahasa Indonesia menggunakan pendekatan *Deep Learning* dengan 480 *tweet* yang akan diuji. Model arsitektur yang digunakan pada penelitian ini adalah *Bidirectional Long Short-Term Memory* (*BLSTM*), dengan Word Embedding (WE), dan *POS tag* sebagai fitur model arsitektur nya. Model memperoleh hasil dengan *F1 score* sebesar 77,08%. Berdasarkan hasil *F1 score* yang diperoleh, ternyata lebih baik dibandingkan penelitian [5],[6]. Maka dalam penelitian selanjutnya mengembangkan model *NER* pada Twitter Indonesia menggunakan kombinasi pendekatan *Deep Learning*, dan *Machine Learning*, sehingga didaptkan hasil sebagai berikut.

Penelitian [7] membangun model *NER* pada Twitter Indonesia dengan menggunakan kombinasi pendekatan *Deep Learning* dan *Machine Learning*, *Bidirectional Long Short-Term Memory* (*BLSTM*) dan *Conditional Random Field* (*CRF*) sebagai solusinya. Entitas yang teridentifikasi berupa Orang, Lokasi dan Organisasi. Corpus yang diuji termasuk 600 *tweet* Indonesia terdiri dari 250 *tweet* formal dan 350 *tweet* informal. Model mendapatkan hasil *F1 score* terbaik dengan menambahkan jenis word embedding *FastText*, yaitu 86,13% untuk *tweet* formal, 81,17% untuk *tweet* informal, dan 84,11% untuk *tweet* gabungan. Hasil penelitian menyimpulkan bahwa model *NER* yang menggunakan kombinasi pendekatan *BLSTM*, *CRF*, dan fitur *FastText* memiliki hasil yang lebih baik daripada model pada penelitian [5], [6], dan [1]. Rangkuman hasil-hasil penelitian sebelumnya di sarikan dalam Tabel 2.1.

Tabel 2.1 Kajian Penelitian *NER* Sebelumnya

Judul	**Penulis**	**Objek**	**Masalah**	**Metode**	**Hasil**
Named entity recognition model for Indonesian tweet using CRF classifier.	Y Munarko, M S Sutrisno, W A I Mahardika, I Nuryasin, Y Azhar. 2018.	*NER* pada Twitter berbahasa Indonesia.	Pengembangan *NER* pada Twitter berbahasa Indonesia.	*Conditional Random Field* (*CRF*).	*F1 score* sebesar 72% *tweet* fornal, 51% *tweet* informal, 70% campur.

Judul	Penulis	Objek	Masalah	Metode	Hasil
Named Entity Recognition on Indonesian Tweets using Hidden Markov Model.	I S Azarine, M A Bijaksana, I Asror. 2019.	*NER* pada Twitter berbahasa Indonesia.	Pengembangan *NER* pada Twitter berbahasa Indonesia.	*Hidden Markov Model* (*HMM*).	Model dapat memberikan *F1 score* sebesar 64,06%.
Named Entity Recognition on Indonesian Twitter Posts Using Long Short-Term Memory Networks	V Rahman, S Savitri, F Agustianti, R Mahendra. 2017.	*NER* pada Twitter berbahasa Indonesia.	Pengembangan *NER* pada Twitter berbahasa Indonesia.	*Bidirectional Long Short-Term Memory* (*BLSTM*).	Model dapat memberikan performa dengan *F1 score* sebesar 77,08%.
Named-Entity Recognition on Indonesian Tweets using Bidirectional LSTM-CRF	D C Wintaka, M A Bijaksana, I Asror. 2019.	*NER* pada Twitter berbahasa Indonesia.	Pengembangan *NER* pada Twitter berbahasa Indonesia.	*Bidirectional Long Short-Term Memory* dan *Conditional Random Field* (*BLSTM* dan *CRF*).	Model dapat memberikan *F1 score* sebesar 86,13% untuk *tweet* formal, 81,17% untuk *tweet* informal, dan 84,11% untuk *tweet* campuran.

2.2. Landasan Teori

2.2.1. *Natural Language Processing*

Pemrosesan Bahasa Alam (*Natural Language Processing/NLP*) adalah suatu cabang dalam bidang kecerdasan buatan yang memungkinkan sistem komputer untuk menyelidiki, menganalisis, dan menghasilkan bahasa sebagaimana yang dilakukan oleh manusia [9]. *NLP* merujuk pada disiplin ilmu komputer, khususnya di dalam cabang kecerdasan buatan atau AI, yang terfokus pada pemberian kemampuan kepada komputer untuk memahami dan merespons teks serta kata-kata

dengan cara serupa seperti yang dilakukan oleh manusia [10]. Bidang *NLP* sendiri dapat dibagi menjadi dua fokus penelitian utama, yaitu Pemahaman Bahasa Alam (*Natural Language Understanding/NLU*) dan Generasi Bahasa Alam (*Natural Language Generation/NLG*) [11]. *NLU*, sebagai salah satu bagian integral dari *NLP*, terfokus pada bagaimana suatu program dapat memahami maksud atau pesan yang ingin disampaikan oleh manusia [12]. *NLG* didefinisikan sebagai tugas menghasilkan teks atau ucapan dari input non-linguistik [13]. *NLP* mengintegrasikan pengolahan bahasa berbasis aturan, komputasi linguistik, dan pemodelan bahasa manusia dengan pendekatan statistik, Machine Learning, serta Deep Learning [13]. Melalui kombinasi ini, teknologi tersebut memberikan kemampuan pada komputer untuk mengolah bahasa manusia baik dalam bentuk teks maupun data suara, dan secara menyeluruh memahami maknanya, termasuk maksud serta sentimen yang disampaikan oleh pembicara atau penulis sehingga *NLP* mempunyai salah satu tugas yang disebut sebagai *Information Extraction*.

2.2.2. *Information Extraction*

Dalam ilmu komputer, *Information Extraction* (*IE*) adalah jenis pencarian informasi yang tujuannya adalah untuk secara otomatis mengekstrak informasi terstruktur. *IE* adalah proses mengekstraksi informasi terstruktur yang berguna dari entitas, hubungan, objek, peristiwa, dan banyak jenis data tidak terstruktur lainnya. Data yang tidak terstruktur diambil informasinya untuk mempersiapkan data agar siap untuk dianalisis [2]. *IE* terdiri dari beberapa sub-bidang yang lebih terfokus, yang masing-masing memiliki masalah yang sulit dipecahkan. Misalnya, dalam penelitian ini, *Named Entity Recognition* (*NER*) digunakan untuk mengekstraksi informasi. Sehingga, peningkatan efisiensi dan ketepatan konversi data tidak terstruktur dalam konteks proses Ekstraksi Informasi memberikan kontribusi positif terhadap peningkatan kapabilitas analisis data.

2.2.3. *Named Entity Recognition*

Pengenalan Entitas Bernama (*Named Entity Recognition/NER*) merupakan suatu sub-tugas yang krusial dalam konteks Ekstraksi Informasi (*IE*) [14]. *NER* juga

dapat didefinisikan sebagai metode penggalian informasi dengan memproses dokumen terstruktur dan tidak terstruktur serta mengidentifikasi entitas yang dapat berupa orang, lokasi, organisasi, atau perusahaan [14]. *NER* juga merupakan proses mengidentifikasi dan mengelompokkan entitas dalam teks dan merupakan dasar dan inti dari sistem *Natural Language Processing* (*NLP*). *NER* secara otomatis mengenali segmen teks yang dianggap sebagai entitas penting, seperti identifikasi nama individu, organisasi, dan lokasi, dengan cermat [15]. Penggunaan *NER* terkait dengan jenis bahasa yang akan diaplikasikan, karena perbedaan bahasa memengaruhi penanganan yang berbeda. Oleh karena itu, algoritma yang diterapkan pada *NER* juga bervariasi. *NER* melibatkan dua tugas pokok, yakni mengenali dengan tepat entitas yang ada dan mengklasifikasikannya ke dalam kategori entitas yang telah ditentukan sebelumnya.

Metode untuk *Named Entity Recognition* (*NER*) umumnya dikelompokkan menjadi tiga arus utama, yakni pendekatan berbasis aturan, pendekatan berbasis pembelajaran, dan pendekatan hibrid [14]. *NER* Berbasis Aturan menitikberatkan pada penemuan entitas dengan menerapkan aturan buatan sebelumnya yang terdiri dari rangkaian pola, menggabungkan karakteristik tata bahasa, sintaksis, dan ejaan. Di sisi lain, pendekatan *NER* berbasis pembelajaran mengatasi tantangan ini dengan mengubah tugas identifikasi menjadi klasifikasi melalui penerapan model klasifikasi statistik. Dalam konteks ini, sistem mencari pola dan relasi dalam teks untuk membentuk model menggunakan metode statistik dan algoritma pembelajaran. Sementara itu, *NER* hibrid memadukan pendekatan *NER* berbasis aturan dengan metode *NER* berbasis pembelajaran mesin, memanfaatkan keunggulan masing-masing metode untuk menghasilkan suatu pendekatan yang lebih inovatif. Pada penelitian ini, peneliti melakukan proses *NER* dengan menggunakan pendekatan *Deep Learning* dan *Machine Learning*. Beberapa contoh penerapan *Named Entity Recognition* (*NER*):

1. Analisis sentiment

   NER dapat digunakan untuk mengidentifikasi nama-nama orang dalam sebuah teks, sehingga dapat membantu dalam menganalisis sentiment terhadap orang-orang tersebut. *NER* dapat digunakan untuk mengidentifikasi nama

pembuat produk tersebut dan kemudian menganalisis apakah review tersebut positif atau negatif terhadap pembuat produk tersebut.

2. Ekstraksi informasi

   NER dapat digunakan untuk mengekstraksi informasi tentang entitas nama dari sebuah teks, seperti nama orang, tempat, atau organisasi. *NER* dapat digunakan untuk mengekstraksi nama perusahaan yang terlibat dalam pertemuan tersebut.

3. Klasifikasi dokumen

   NER dapat digunakan untuk membantu dalam mengklasifikasikan dokumen berdasarkan entitas nama yang terdapat di dalamnya. *NER* dapat digunakan untuk mengklasifikasikan dokumen berdasarkan nama perusahaan yang disebutkan di dalamnya.

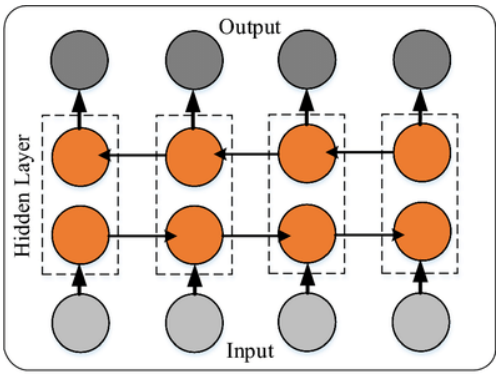
4. Pencarian informasi

   NER dapat digunakan untuk memudahkan pencarian informasi dengan mengidentifikasi entitas nama dalam sebuah teks dan kemudian mencari informasi terkait dengan entitas tersebut. Misalnya, jika sebuah teks menyebutkan nama sebuah perusahaan, *NER* dapat digunakan untuk mengidentifikasi nama perusahaan tersebut dan kemudian mencari informasi tentang perusahaan tersebut menggunakan mesin pencari.

2.2.4. *Bidirectional Long Short-Term Memory*

Long Short-Term Memory (*LSTM*) merupakan pengembangan dari fungsi *Recurrent Neural Network* (*RNN*). *LSTM* dapat menangani masalah ketergantungan jangka panjang dengan mekanisme gerbang dan sel memori [1]. *Bidirectional Long Short-Term Memory* (*BLSTM*) adalah jenis *Recurrent Neural Network* (*RNN*) yang memproses urutan input dalam dua arah, membuatnya peka terhadap urutan-urutan input. Secara umum cara kerja *BLSTM* yaitu dengan menggunakan dua jaringan *LSTM*, satu yang memproses urutan input dari awal hingga akhir (*forward*), dan satu lagi yang memproses urutan input dari akhir ke awal (*backward*). Output dari *LSTM forward* dan *backward* kemudian digabungkan, memungkinkan model

memanfaatkan konteks masa lalu dan masa depan untuk membuat prediksi tentang urutan input. Maka cara kerja dan ilustrasi *BLSTM* dari penelitian [7] dapat dilihat yaitu:



Gambar 2.1 Ilustrasi *BLSTM*

1. Input diterima oleh lapisan *LSTM* yang pertama dan diolah oleh sel *LSTM* yang terdapat di dalamnya.
2. Sel *LSTM* yang pertama akan mengolah input dengan menggunakan *input gate*, *forget gate*, dan *output gate*. Menurut penelitian [7] rumus dasar untuk *LSTM* ditunjukan dalam persamaan 2.1 sampai 2.5.

$$input\ gate: i_t = \delta(W_i x_t + W_i h_{t-1} + b_i) \quad (2.1)$$

Proses *input gate* bertanggung jawab untuk menyaring dan menentukan informasi khusus yang akan diperbarui ke dalam bagian *cell state*, menggunakan fungsi aktivasi *sigmoid*. Fungsi aktivasi *sigmoid* membantu mengubah nilai yang berada dalam rentang -1 hingga 1 menjadi rentang 0 hingga 1. Fase ini terbukti bermanfaat saat melakukan pembaruan atau pengabaian data, karena nilai yang diubah menjadi 0 akan diabaikan (lupakan), sementara nilai yang dikalikan dengan 1 akan tetap tidak berubah (disimpan). Dengan demikian, *sigmoid* membantu dalam memutuskan apakah sebuah nilai harus disimpan atau dilupakan.

$$forget\ gate: f_t = \delta(W_f x_t + W_f h_{t-1} + b_f) \quad (2.2)$$

Forget gate adalah komponen yang bertanggung jawab untuk mengeliminasi informasi yang dianggap kurang signifikan atau tidak relevan dalam konteks pengolahan yang sedang berlangsung, menggunakan fungsi *sigmoid*. Data x_t merujuk pada data input (vektor input *x* pada timestep *t*), sementara h_{t-1} merupakan vektor *hidden state* pada timestep sebelumnya, yaitu *t-1*.

$$cell\ state: c_t = f_t * c_{t-1} + i_t * tanh(W_c x_t + W_c h_{t-1} + b_c) \quad (2.3)$$

Cell state merupakan informasi yang disimpan oleh sel *LSTM* dan dapat dipertahankan selama beberapa periode waktu. *Cell state* bertindak sebagai ingatan jangka panjang yang dapat dibawa ke periode waktu yang akan datang. *Cell state* diupdate setiap kali sel *LSTM* menerima input baru.

$$output\ gate: o_t = \delta(W_o x_t + W_o h_{t-1} + b_o) \quad (2.4)$$

Output gate digunakan untuk menghasilkan nilai output pada *hidden state*, dengan menjalankan fungsi *sigmoid* pada satu jalur dan menempatkan nilai *cell state* melalui fungsi *tanh* pada jalur yang lain. Setelah mendapatkan nilai output dari kedua fungsi aktivasi tersebut, yaitu *sigmoid* dan *tanh*, keduanya dikalikan secara bersamaan sebelum melanjutkan ke tahap berikutnya. Tahap ini akhirnya mengarah pada proses klasifikasi dari semua perhitungan yang dilakukan oleh mekanisme *LSTM*.

$$hidden\ state: h_t = o_t * tanh(c_t) \quad (2.5)$$

Hidden state merupakan output dari sel *LSTM* untuk satu periode waktu tertentu. *Hidden state* terdiri dari informasi yang disimpan di dalam *cell state* dan input yang diterima pada periode waktu tersebut.

Dimana, i_t , f_t , c_t adalah input, forget, dan output gate pada timestep *t*. c_t , h_t adalah *cell state*, dan *hidden state* pada timestep *t*. x_t adalah input dari timestep *t*. *W*, *b* adalah bobot dan bias yang dioptimalkan dalam proses pelatihan. δ adalah fungsi aktivasi *sigmoid*.

3. Output dari sel *LSTM* yang pertama akan digunakan sebagai input untuk sel *LSTM* yang selanjutnya, dan proses ini akan diulangi hingga lapisan *LSTM* terakhir.
4. Setelah proses di lapisan *LSTM* selesai, maka hasil output dari lapisan *LSTM* terakhir akan digabungkan dengan hasil output dari proses yang sama namun dari arah yang berlawanan, yaitu dari belakang ke depan, sehingga dapat menghasilkan output *bidirectional*.
5. Maka pada proses *BLSTM* hanya perlu menambahkan dua output dari rangkaian *LSTM* yang berjalan secara terpisah, satu dari depan ke belakang dan satu lagi dari belakang ke depan. Kedua rangkaian *LSTM* ini kemudian dijumlahkan (*concatenated*) untuk membentuk output dari *BLSTM*. Menurut penelitian [7] rumus *BLSTM* ditunjukkan dalam persamaan 2.6.

$$h_t = concat(\rightarrow h_t, h_t \leftarrow) \quad (2.6)$$

Dimana, $\rightarrow h_t$ adalah *hidden state* dari rangkaian *LSTM* yang berjalan dari depan ke belakang (*forward*) pada timestep *t*. $h_t \leftarrow$ adalah *hidden state* dari rangkaian *LSTM* yang berjalan dari belakang ke depan (*backward*) pada timestep *t*. *concat* adalah fungsi yang digunakan untuk menggabungkan *(concatenate)* dua vector.

Dengan menggunakan *BLSTM*, model dapat memproses informasi dari dua arah sekaligus, sehingga dapat lebih baik dalam mengolah informasi yang memiliki kaitan dengan konteks sekitarnya. *BLSTM* biasanya digunakan dalam tugas-tugas *NLP* seperti pembuatan kalimat dan ekstraksi informasi, dimana informasi yang diproses memiliki kaitan dengan kata-kata yang ada di sekitarnya.

2.2.5. *Hidden Markov Model*

Hidden Markov Model (*HMM*) adalah suatu model probabilistik yang menggambarkan urutan pengamatan dalam hal sekumpulan keadaan tersembunyi yang mendasarinya. Dalam *HMM*, kemungkinan transisi dari satu keadaan tersembunyi ke keadaan lain dan kemungkinan memancarkan pengamatan tertentu

dengan keadaan tersembunyi semuanya ditentukan oleh model. Hal ini memungkinkan model untuk menangkap ketergantungan temporal antara pengamatan dan keadaan tersembunyi yang mendasarinya. Pada dasarnya bahwa peristiwa yang diamati tidak akan sesuai dengan status langkah demi langkahnya tetapi terkait dengan sekumpulan distribusi probabilitas. Jika menurut penelitian [6] dalam *Hidden Markov Model* (*HMM*), keadaan tidak dapat diobservasi secara langsung. Sebaliknya, yang dapat diobservasi adalah parameter-parameter yang dipengaruhi oleh keadaan. Setiap keadaan memiliki distribusi peluang yang mungkin muncul. Oleh karena itu, urutan output yang dihasilkan oleh *HMM* memberikan informasi mengenai urutan pernyataan.

Sistem yang sedang dimodelkan diasumsikan sebagai rantai Markov dan dalam prosesnya, ada beberapa status tersembunyi. Dalam hal ini status tersembunyi adalah proses yang bergantung pada proses/rantai Markov utama. Tujuan utama *HMM* adalah mempelajari rantai Markov dengan mengamati status tersembunyinya. Mempertimbangkan proses Markov X dengan status tersembunyi Y di sini *HMM* memantapkan bahwa untuk setiap stempel waktu distribusi probabilitas Y tidak boleh bergantung pada sejarah X menurut waktu itu. *HMM* bisa diterapkan dalam berbagai tugas *NLP*, salah satunya adalah *NER*.

Menurut penelitian [6] bentuk umum dari *HMM* terdiri dari lima elemen (M, π, A, B). Jika dianggap $\lambda = (\pi, A, B)$ maka *HMM* memiliki elemen tertentu *N* dan *M*. Maka penjelasan dari unsur-unsur tersebut menurut penelitian [6] yaitu:

a. *N* yaitu banyaknya kondisi tersembunyi yang dilambangkan dengan himpunan terbatas kemungkinan keadaan adalah $Q = \{q_1, q_2, \ldots, q_N\}$.
b. *M* yaitu banyaknya kondisi yang teramati dilambangkan dengan himpunan terbatas kemungkinan pengamatan adalah $Q = \{V_1, V_2, \ldots, V_M\}$.
c. *Transition Opportunity Matrix (A):*

$$A = \{a_{ij}\}, a_{ij} = P(X_{i+1} = q_j \| X_t = q_i) \quad (2.7)$$

Dimana, untuk masing-masing $1 \leq i \leq N$ dan $1 \leq j \leq N$. a_{ij} adalah elemen dari *A* yang merupakan himpunan kemungkinan distribusi transfer keadaan (probabilitas transisi) saat $t + 1$, jika diketahui *X* saat *t*. Jadi *A* berukuran $N \; x \; N$.

d. *Emission Opportunity Matrix (B):*

$B = \{b_i(v_k)\}$ merupakan matriks probabilitas keluaran dengan $b_i(v_k)$ merupakan peluang pengamatan saat *t* dikategorikan v_k dengan ketentuan keadaan saat *t* berada pada keadaan q_i. Dapat ditulis dengan persamaan 2.8.

$$b_i(v_k) = P(O_t = v_k \,||\, X_t = q_i) \qquad (2.8)$$

untuk $1 \leq i \leq N$ dan $1 \leq k \leq N$ Dimana matriks peluang pengamatan memiliki karakteristik bahwa setiap barisnya adalah satu.

e. *Initial Opportunity Matrix (π)*

$$\pi = \{\pi_i\}, \pi\, i = P(X_o = q_i) \qquad (2.9)$$

untuk $i = 1,2,3,\ldots,N, \pi_i \geq 0$ dimana $\pi_i = P(X_o = q_i)$ adalah himpunan probabilitas awal.

2.2.6. Confusion Matrix

Kinerja dari model *NER* dalam penelitian kali ini dihitung dengan mencari nilai *F1 score* nya. *Confusion matrix*, atau yang umumnya dikenal sebagai *error matrix*, adalah struktur tabel khusus yang digunakan untuk memvisualisasikan kinerja algoritma dan seringkali diterapkan dalam konteks *supervised learning* [16]. *F1 score* mengambil rata-rata harmonis dari nilai *precision* dan *recall*. Tabel 3.6 menunjukan *confusion matrix*.

Tabel 3.8 *Confusion Matrix*

	Actual Positive	***Actual Negative***
Predicted Positive	*True Positive (TP)*	*False Positive (FP)*
Predicted Negative	*False Negative (FN)*	*True Negative (TN)*

True Positive (TP) adalah kasus dimana model memprediksi hasil positif dengan benar. *False Positive (FP)* adalah kasus dimana model salah memprediksi hasil positif. *False Negative (FN)* adalah kasus dimana model salah memprediksi

hasil negatif. *True Negative (TN)* adalah kasus dimana model memprediksi hasil negatif dengan benar. Untuk mencari nilai *F1 score*, peneliti perlu mencari nilai *recall* dan *precision* terlebih dahulu yang dijelaskan sebagai berikut.

1. Precision

Precision mengukur jumlah instans yang benar yang diperoleh lalu dibagi dengan semua instans yang diperoleh [8]. *Precision* adalah ukuran keakuratan model ketika memprediksi hasil yang positif. Ini didefinisikan sebagai jumlah prediksi positif sebenarnya yang dibuat oleh model dibagi dengan jumlah total prediksi positif yang dibuat oleh model. Nilai presisi yang tinggi menunjukkan bahwa model tersebut bagus dalam mengidentifikasi kasus positif, tetapi tidak memperhitungkan jumlah negatif palsu. Dapat di lihat dalam persamaan 3.1.

$$Precision = \frac{TP}{TP+FP} \qquad (3.1)$$

2. Recall

Recall mengukur jumlah instans yang benar yang diperoleh lalu dibagi dengan semua instans yang benar [8]. *Recall* adalah ukuran kemampuan model untuk mengidentifikasi semua kasus positif dalam data. Ini didefinisikan sebagai jumlah prediksi positif sebenarnya yang dibuat oleh model dibagi dengan jumlah total kasus positif aktual dalam data. Nilai perolehan yang tinggi menunjukkan bahwa model tersebut bagus dalam menemukan semua kasus positif, tetapi tidak memperhitungkan jumlah positif palsu. Dapat di lihat dalam persamaan 3.2.

$$Recall = \frac{TP}{TP+FN} \qquad (3.2)$$

3. F1 score

F1 score adalah rata-rata harmonis antara akurasi dan recall [8]. *F1 score* juga merupakan metrik yang digunakan untuk mengevaluasi kinerja model klasifikasi. Ini didefinisikan sebagai rata-rata harmonik presisi dan daya ingat, dengan nilai antara 0 dan 1, di mana nilai yang lebih tinggi menunjukkan kinerja yang lebih baik. *F1 score* sering digunakan sebagai keseimbangan antara presisi dan daya ingat,

karena menghukum model yang memiliki keseimbangan yang buruk di antara keduanya. Dapat di lihat dalam persamaan 3.3.

$$F1\ score = 2 * \frac{Precision*Recall}{Precision+Recall} \quad (3.3)$$