BAB 2
LANDASAN TEORI

2.1 Lirik Lagu

Lirik lagu merupakan suatu bentuk karya sastra berbentuk puisi pendek, yang mengandung emosi dari penulis lagu. Lirik lagu digunakan sebagai media penyampaian pikiran dan perasaan dari pengarang lagu kepada para pendengar lagu. Dengan adanya lirik lagu, diharapkan pesan pengarang lagu dapat tersampaikan ke masyarakat luas [15].

Secara khusus, lirik lagu merupakan puisi pendek yang disusun dari kata-kata berirama dengan struktur dan aturan tertentu, berkaitan dengan suara musik dan penyanyi lagu [15]. Berkaitan dengan struktur penulisan puisi, lirik lagu termasuk ke dalam bentuk puisi modern. Hal ini dikarenakan tidak terdapat aturan penulisan rima, baris, dan bait, pada penulisan lirik lagu. Pengarang lagu pun dapat lebih bebas berkreasi dalam menulis lirik, sehingga lagu yang dikembangkan sesuai dengan keinginan pengarang lagu [16].

2.2 Analisis Sentimen

Analisis sentimen merupakan studi komputasi yang mempelajari sentimen atau emosi dari suatu teks. Dengan analisis sentimen, suatu teks dapat diketahui nilai polaritasnya, baik itu positif, negatif ataupun netral. Nilai polaritas tersebut akan menentukan sentimen, emosi, ekspresi yang terdapat pada suatu teks yaitu positif, negatif, atau netral [17].

Analisis sentimen juga dapat disebut dengan *opinion mining*. Hal ini disebabkan karena dengan melakukan analisis sentimen, opini yang terkandung pada suatu teks subjektif dapat dianalisis. Opini yang ditemukan kemudian akan diklasifikasi menjadi opini positif atau negatif [18].

Analisis sentimen dapat diterapkan pada teks opini orang-orang mengenai suatu topik, produk, layanan, organisasi, individu, ataupun hal-hal lainnya. Dari hasil analisis sentimen tersebut, dapat diketahui ekspresi atau emosi orang-orang terhadap objek yang dimaksud. Hasil dari analisis sentimen juga dapat dikembangkan ke dalam bentuk *rating*, dimana teks opini dapat diklasifikasikan nilai *rating*-nya [19].

2.3 Preprocessing

Tahap *preprocessing* adalah salah satu tahap yang penting dalam pembuatan model *Natural Language Processing* (NLP). Pada tahap *preprocessing*, data teks diolah agar lebih terstruktur dan siap diproses lebih lanjut [20]. Ada beberapa hal yang dilakukan pada tahap *preprocessing*, yaitu *data cleaning*, *case folding*, *stopwords removal*, dan *lemmatization* [21][22].

1. *Data Cleaning*

   Pada proses *data cleaning*, data teks dibersihkan dari angka, simbol, dan karakter-karakter yang tidak relevan. Karakter-karakter non-alfabet perlu dihapus karena dapat mempengaruhi penelitian ini. Selain itu, *single character* dan spasi berlebihan (*multiple whitespaces*) juga dihapus dari teks, sehingga struktur teks menjadi lebih rapi [20].

2. *Case Folding*

   Pada proses *case folding*, teks diubah ke dalam bentuk yang seragam, baik itu kecil semua (*lower case*) ataupun besar semua (*upper case*) [22]. Dengan itu, teks menjadi lebih seragam dan proses klasifikasi dapat berjalan lebih mudah.

3. *Stopwords Removal*

   Stopwords adalah kata-kata umum, yang tidak memiliki makna. Pada proses *stopwords removal*, kata-kata yang bersifat umum dan tidak memiliki makna dihapus dari teks. Kata-kata yang kurang penting akan ditandai, kemudian dihapus, sehingga teks tersaring menjadi lebih bersih [21].

4. *Lemmatization*

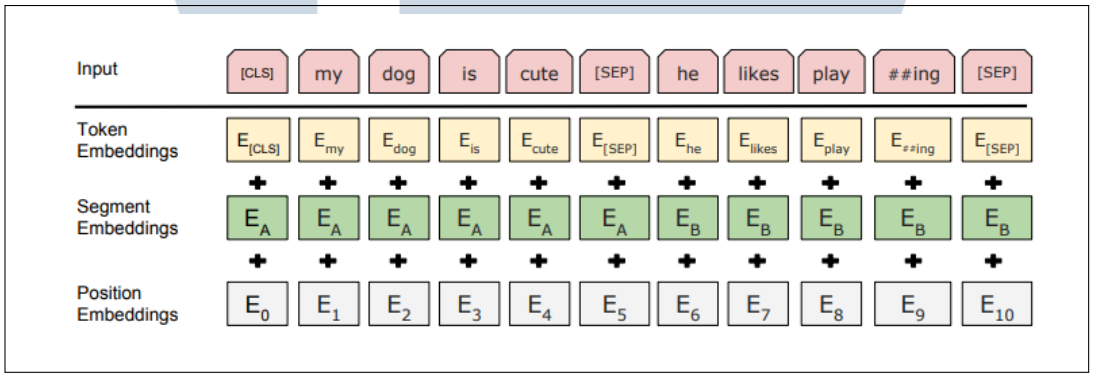
   Proses *lemmatization* adalah proses transformasi suatu kata ke bentuk dasar, atau akar, dari kata tersebut [22]. Dengan proses *lemmatization*, imbuhan di awal, tengah, ataupun akhir dari suatu kata dapat dihapus.

2.4 Bidirectional Encoder Representations From Transformers (BERT)

Bidirectional Encoder Representations from Transformers (BERT) merupakan sebuah *pre-trained language model*, sebuah model *deep learning* yang dapat digunakan untuk proyek *Natural Language Processing* (NLP) [8].

2.4.1 Representasi Input/Output Model BERT

Model BERT dapat menjalankan berbagai macam *downstream task*. *Downstream task* adalah tugas spesifik yang dapat dilakukan oleh suatu model, fungsi yang dapat menghasilkan keluaran yang diharapkan. Contoh dari *downstream task* yang dapat dikerjakan oleh model BERT adalah fungsi tanya-jawab (*question answering*). Untuk menjalankan *downstream task*, model BERT memerlukan *input* yang telah ditokenisasi, membentuk sebuah *sequence*. Adapun token [CLS] adalah token pertama pada *sequence* yang digunakan sebagai representasi suatu *sequence*, dan token [SEP] digunakan sebagai penghubung antarsegmen pada suatu *sequence*. Token [CLS] merupakan token spesial yang digunakan pada awal *sequence* [12].



Gambar 2.1. Proses tokenisasi *input* model BERT

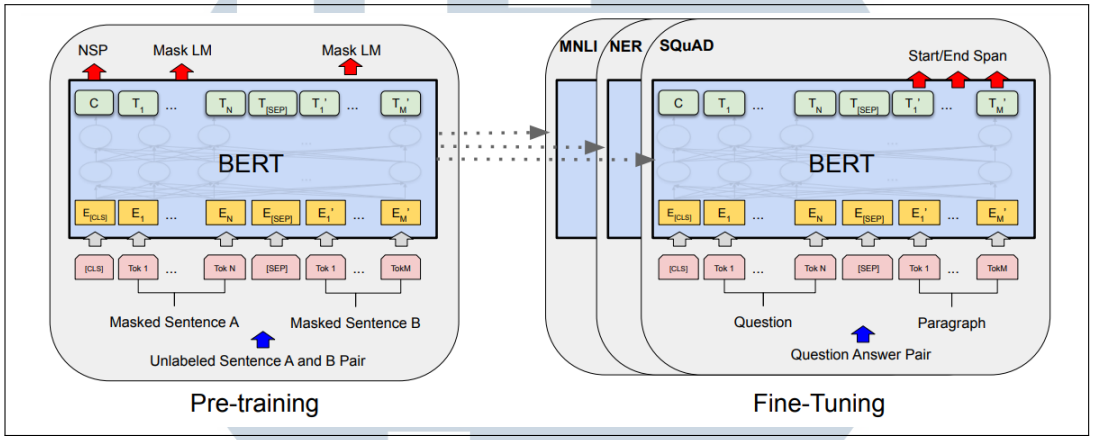
sumber: [12]

Proses tokenisasi teks meliputi *token embeddings*, *segment embeddings*, dan *position embeddings*. Pada proses *token embeddings*, *input* teks direpresentasikan ke dalam bentuk vektor, menggunakan *WordPiece embedding*. Pada proses *segment embeddings*, token-token pada *sequence* dikelompokkan sesuai dengan segmennya. Segmen-segmen pada *sequence* dapat dipisahkan dengan adanya token [SEP]. Pada proses *position embeddings*, token-token pada *sequence* ditentukan referensi posisinya, berdasarkan urutan token pada *sequence* tersebut [8] [12].

2.4.2 Pemodelan BERT

Perancangan model BERT terdiri dari 2 tahap, yaitu tahap *pre-training* dan tahap *fine tuning*. Pada tahap *pre-training*, data tanpa label akan dilatih dengan 2 metode *unsupervised pre-training*, yaitu metode *masked language model* dan

metode *next sentence prediction*. Setelah itu, pada tahap *fine-tuning*, model yang sudah di-*pre-train* akan disesuaikan sehingga dapat melakukan tugas klasifikasi yang lebih spesifik (*downstream task*). Adapun model yang dibangun merupakan model dengan *multi-layer bidirectional transformer encoder*. Model BERT terdiri dari 12 *layer*, 768 *hidden states*, dan 12 *self-attention-heads* [12].



Gambar 2.2. Tahap-tahap pemodelan BERT

sumber: [12]

A. Tahap Pre-training

Pada pemodelan BERT, representasi-representasi akan dibentuk dari suatu teks. Representasi-representasi teks tersebut akan diproses dari dua arah yang berbeda, dengan metode *masked language model* (MLM) dari kiri ke arah kanan dan kanan ke arah kiri. Tujuan dijalankannya MLM pada BERT adalah untuk *masking* kata-kata secara acak pada suatu kalimat. Model akan *masking* beberapa token kata, kemudian menggantinya dengan token [MASK] [12]. Kata-kata yang telah di-*mask* akan diprediksi model, dengan pendekatan konteks dari sebelah kiri dan kanan kata yang di-*mask*. Dengan menggunakan pendekatan kiri dan kanan secara bersamaan, lebih banyak fitur teks yang dapat diekstraksi untuk kegiatan *pre-training* [23].

Pada kegiatan MLM, 15% dari jumlah token keseluruhan dipilih secara acak oleh BERT, untuk diprediksi. Token-token ini akan di-*masking*, tetapi tidak selalu token-token tersebut diubah ke dalam bentuk token [MASK]. Dari token-token yang di-*mask*, 80% token diubah menjadi token [MASK], 10% token diubah menjadi token lain secara acak, dan 10% token tidak diubah bentuknya. Hal ini dilakukan agar tidak terjadi ketidaksesuaian token yang diproses pada *pre-training* dan *fine-tuning*, dikarenakan token [MASK] yang tidak muncul pada tahap *fine-tuning* [12].

Selain MLM, dilakukan *next sentence prediction* (NSP). Dengan NSP, BERT dapat mempelajari hubungan suatu kalimat dengan kalimat yang lain [23]. Hal ini yang membedakan model BERT dengan model-model yang sudah ditemukan sebelumnya. Dengan adanya MLM dan NSP, model BERT dapat menghasilkan analisis sentimen yang lebih baik dari banyak model NLP lainnya. Model BERT dapat menyelesaikan berbagai macam tugas dengan baik dan mengungguli banyak sistem permodelan lainnya [12].

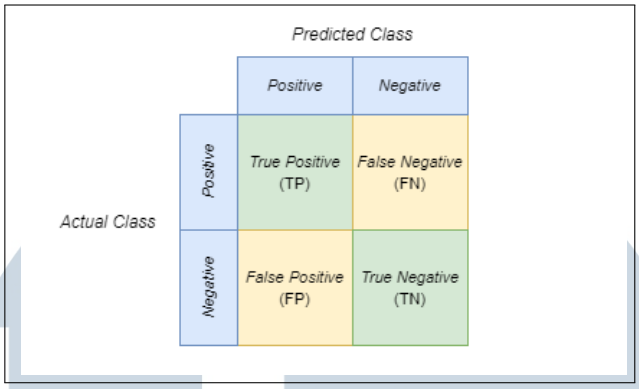
B. Tahap Fine-tuning

Pada tahap *fine-tuning*, model BERT yang sudah di-*pre-train* disesuaikan lebih lanjut untuk tugas-tugas yang lebih spesifik (*downstream task*). Model BERT dapat dengan mudah melakukan *fine-tuning* karena mekanisme *self-attention* yang dimiliki BERT.

Pada tahap ini, *input* dan *output* spesifik dimasukkan ke model BERT, kemudian semua parameter pada model disesuaikan satu per satu, sehingga model dapat menghasilkan *output* yang diharapkan. Untuk menghasilkan *output*, representasi token-token *input* digunakan untuk menjalankan fungsi-fungsi yang dikerjakan pada level token, seperti fungsi tanya-jawab. Adapun representasi token [CLS] secara khusus digunakan untuk menjalankan fungsi-fungsi klasifikasi, seperti analisis sentimen. Dengan demikian, model BERT pun dapat menjalankan analisis sentimen [12].

2.5 Metrik Evaluasi

Metrik evaluasi atau metrik performa, adalah metrik yang dapat digunakan sebagai indikator performa analisis sentimen [8]. Metrik evaluasi ini didasarkan dari hasil *confusion matrix*. *Confusion matrix* adalah matriks yang menunjukkan perbandingan nilai aktual dan nilai hasil prediksi dari suatu permodelan analisis sentimen.



Gambar 2.3. *Confusion matrix*

Dari *confusion matrix*, dapat diketahui nilai *true positive* (TP), *true negative* (TN), *false positive* (FP), *false positive* (FN). Nilai *true positive* (TP) menyatakan jumlah data yang diprediksi positif dan sesungguhnya positif, sedangkan *false positive* (FP) menyatakan jumlah data yang diprediksi positif tetapi sesungguhnya negatif. Nilai *true negative* (TN) menyatakan jumlah data yang diprediksi negatif dan sesungguhnya negatif, sedangkan *false negative* (FN) menyatakan jumlah data yang diprediksi negatif tetapi sesungguhnya positif. Dari nilai-nilai tersebut *accuracy*, *precision*, *recall*, *f1-score* dari suatu permodelan dapat diketahui [24].

2.5.1 Accuracy

Accuracy adalah indikator yang menunjukan keakuratan model untuk memprediksi label secara keseluruhan. *Accuracy* dihitung dengan membandingkan elemen yang diprediksi benar dengan keseluruhan elemen yang diprediksi. Bila dilihat dari *confusion matrix*, nilai akurasi dapat dihitung dari penjumlahan semua elemen yang terdapat pada diagonal utama, kemudian dibagi dengan jumlah semua elemen lainnya [24]. Berikut merupakan rumus dari *accuracy*.

$$Accuracy = \frac{TN + TP}{TP + FP + TN + FN} \tag{2.1}$$

2.5.2 Precision

Precision adalah indikator yang menunjukan keberhasilan model dalam memprediksi nilai elemen yang positif, seberapa besar model dipercaya dalam memprediksi nilai positif. *Precision* dihitung dengan membandingkan elemen yang diprediksi positif dengan keseluruhan elemen yang diprediksi positif [24]. Berikut

merupakan rumus dari *precision*.

$$Precision = \frac{TP}{TP + FP} \tag{2.2}$$

2.5.3 Recall

Recall adalah indikator yang menunjukan keakuratan model untuk memprediksi nilai elemen yang positif. *Recall* dihitung dengan membandingkan elemen yang diprediksi positif dengan keseluruhan elemen yang aktualnya bernilai positif [24]. Berikut merupakan rumus dari *recall*.

$$Recall = \frac{TP}{TP + FN} \tag{2.3}$$

2.5.4 F1-Score

F1-score adalah indikator yang menunjukan performa model, dengan menghitung rata-rata harmonik *precision* dan *recall*. Dengan demikian, didapatkan perbandingan merata antara *precision* dan *recall* [24]. Berikut merupakan rumus dari *f1-score*.

$$F1 - Score = 2 \cdot \left(\frac{precision \cdot recall}{precision + recall} \right) \tag{2.4}$$