BAB I

PENDAHULUAN

1.1 Latar Belakang

Di era modern ini, media sosial berkembang sangat pesat di dunia maya. Media sosial merupakan sebuah wadah baru yang memungkinkan kita untuk menyuarakan ide-ide dan berbagi perspektif serta pemikiran baru tentang berbagai isu dan topik. Media sosial memungkinkan terjadinya interaksi tanpa batasan waktu dan ruang antara infrastruktur dan suprastruktur sistem politik. Penggunaan internet semakin meluas di Indonesia, tidak hanya untuk keperluan pribadi atau bisnis-komersial tetapi juga untuk urusan politik (Noorikhsan et al., 2023). Salah satu fungsi media sosial adalah untuk mendeteksi pro dan kontra yang timbul dari kebijakan pemerintah dan mengukur bagaimana tingkat penerimaan kebijakan tersebut terhadap kelangsungan hidup di Indonesia (Sukma et al., 2020). Sentimen publik yang diserap melalui media sosial biasanya untuk menjajaki cara menarik perhatian publik dan pola respon dari jejaring sosial dalam konteks pengembangan event mobilitas sehingga dapat menjadi acuan perbaikan kinerja pelayanan dan regulasi pemerintah (Ye et al., 2019).

Di era digital seperti saat ini banyak sekali aplikasi media sosial yang digunakan oleh masyarakat dalam kesehariannya di antaranya seperti YouTube, Instagram, Tiktok, Facebook, Twitter dan lain sebagainya. Media sosial merupakan media online yang membantu seseorang memperoleh dan mengkomunikasikan informasi (Kamhar & Lestari, 2019). Masyarakat bebas memberikan tanggapan dan pendapatnya di berbagai media sosial dan menjadikan media sosial tempat untuk mencurahkan segala hal (Fitriana et al., 2021).

Analisis sentimen merupakan bidang multidisiplin, meliputi psikologi, sosiologi, pemrosesan bahasa alami, dan pembelajaran mesin. Baru-baru ini, jumlah data dan daya komputasi yang tumbuh secara eksponensial memungkinkan bentuk analisis yang lebih canggih. Oleh karena itu, pembelajaran mesin menjadi alat yang dominan untuk analisis sentimen (Ligthart et al., 2021). Analisis sentimen dapat mengklasifikasikan kalimat opini berupa kalimat positif, negatif, dan netral. Sehingga dapat menghasilkan sebuah informasi bagi entitas baik itu perusahaan maupun instansi. Untuk mendukung analisis sentimen tersebut ada berbagai metode yang digunakan diantaranya Naive Bayes Classifier, Support Vector Machine, K-NN, RNN, C4.5, Lexicon Based, LDA Based Topic Modeling, dan beberapa algoritma lainnya (Permatasari et al., 2021).

Berdasarkan penelitian yang dilakukan oleh Helmi Imaduddin, dkk dengan judul “Sentiment Analysis in Indonesian Healthcare Applications using IndoBERT Approach” menggunakan Algoritma IndoBERT dengan menganalisis ulasan dari aplikasi layanan kesehatan di Google Play Store, termasuk Halodoc, Alodokter, dan klikdokter yang menghasilkan accuracy 96%, precision 95%, recall 96%, dan F1-score 95%. Pada penelitian lainnya yang dilakukan oleh Ali Alshahrani, dkk dengan judul “Identifying Optimism and Pessimism in Twitter Messages Using XLNet and Deep Consensus” menggunakan algoritma XLNet dan Deep Consensus dengan menganalisis pesan Optimisme dan Pesimisme di Twitter yang menghasilkan akurasi model XLNet sebesar 96,45%. Selanjutnya, penelitian oleh Thariq Iskandar Zulkarnain Maulana Putra, dkk

dengan judul “Classification Model Based on Multiclass Classification with a Combination of Indobert Embedding and Long Short-Term Memory for Indonesian-language Tweets” menggunakan algoritma Word2Vec – LSTM, IndoBERT, dan IndoBERT - LSTM, melakukan klasifikasi terhadap 10 kelas data dari crawling pada aplikasi twitter, yaitu beasiswa, bulutangkis, demokrasi, film, investasi, kecantikan, konser, pajak, sepakbola, dan wisata yang menghasilkan Model IndoBERT - LSTM berhasil mendapatkan F1-score sebesar 98,90% pada dataset yang tidak termodifikasi (peningkatan 0,70% dari model Word2Vec - LSTM dan 0,40% dari model IndoBERT) dan 92,83% pada dataset yang telah termodifikasi (peningkatan 4,51% dari model Word2Vec-LSTM dan 0,69% dari model fine-tuned IndoBERT).

IndoBERT merupakan adaptasi dari model BERT (Bidirectional Encoder Representations from Transformers) yang dirancang khusus untuk menangani teks berbahasa Indonesia. Dikembangkan untuk mengatasi kebutuhan pemrosesan bahasa alami Natural Language Processing (NLP) dalam konteks bahasa Indonesia, IndoBERT dilatih menggunakan korpus teks yang luas dan beragam dari sumber-sumber berbahasa Indonesia, termasuk berita, media sosial, dan dokumen bahasa Indonesia lainnya (Wilie et al., 2020). Sementara itu, XLNet memperkenalkan teknik pra-latihan baru bernama Permutation Language Modeling. Teknik ini menggabungkan manfaat dari model autoregresif dan bidirectional, dengan memungkinkan model belajar dari berbagai urutan kata dalam kalimat. Pendekatan ini memberikan fleksibilitas lebih besar dalam memahami konteks yang kompleks (Yang et al., 2019).

Dalam penelitian ini digunakan pendekatan *deep learning* berbasis arsitektur *transformer* dengan dua model utama, yaitu IndoBERT dan XLNet, yang terbukti efektif dalam berbagai tugas *Natural Language Processing* (NLP). Kedua model ini diharapkan mampu memberikan hasil analisis sentimen yang lebih akurat terhadap opini publik di media sosial. Berdasarkan uraian tersebut, penulis melakukan penelitian dengan judul “Analisis Sentimen terhadap Kinerja Presiden dan Wakil Presiden Menggunakan Model IndoBERT dan XLNet.” Penelitian ini bertujuan untuk menganalisis sentimen yang terdapat dalam komentar pengguna pada platform X (Twitter) dan YouTube terkait kinerja Presiden dan Wakil Presiden Indonesia. Melalui pendekatan ini, diharapkan dapat diketahui kecenderungan opini publik terhadap kinerja pemerintahan. Pemahaman terhadap kecenderungan tersebut menjadi penting karena dapat memberikan gambaran yang lebih objektif mengenai persepsi masyarakat, baik dalam bentuk dukungan, kritik, maupun respons netral yang muncul secara alami di ruang digital. Selain itu, penelitian ini juga diharapkan dapat memberikan kontribusi terhadap pengembangan teknologi *Natural Language Processing* (NLP) berbasis Bahasa Indonesia, khususnya dalam penerapan model *transformer* seperti IndoBERT dan XLNet untuk memahami persepsi masyarakat di ruang digital.

1.2 Teori

1.2.1 *Natural Language Processing* (NLP)

Natural Language Processing (NLP) adalah bidang ilmu komputer yang mempelajari bagaimana mesin memahami dan memproses bahasa manusia. Tujuan utama NLP

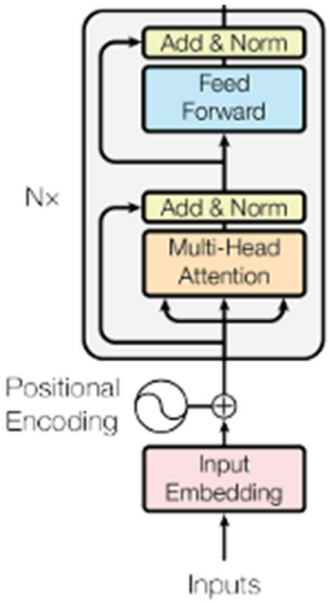
adalah memungkinkan komputer berinteraksi dengan manusia dalam bahasa alami dan memproses, menganalisis, dan menghasilkan teks atau ucapan bahasa manusia (Hasibuan & Heriyanto, 2022). Salah satu aplikasi dari NLP adalah analisis sentimen, yaitu proses mengidentifikasi apakah suatu teks berisi sentimen positif, negatif, atau netral (Togatorop et al., 2021).

Text Mining merupakah proses mengekstraksi suatu pola untuk diteliti yang datanya berasal dari sebuah teks. *Text Mining* merupakan disiplin ilmu yang didasarkan pada *information retrieval*, *data mining*, *machine learning*, *statistic* dan *linguistic* komputasi (Surya et al., 2022). Pada umumnya proses kerja dari *Text Mining* banyak mengadopsi dari penelitian data mining namun yang menjadi perbedaan adalah pola yang digunakan oleh *Text Mining* diambil dari sekumpulan bahasa alami yang tidak terstruktur sedangkan dalam data mining pola yang diambil dari *database* yang terstruktur (Ridwansyah, 2022).

Analisis sentimen merupakan tugas mendasar dalam *natural language processing* (NLP). Analisis ini penting untuk memahami teks yang dibuat pengguna dalam laporan berita, ulasan produk, atau diskusi sosial (Gao et al., 2019). Analisis sentimen mengklasifikasikan sentimen berdasarkan polaritas teks dalam sebuah frasa, sehingga dapat ditentukan sebagai sentimen positif, negatif, atau netral (Fatima Anees et al., 2020; Prabowo & Wiguna, 2021). Dengan menggunakan analisis semacam ini, berbagai perspektif publik terhadap kebijakan pemerintah dapat ditentukan. Analisis sentimen mengekstraksi opini yang diungkapkan oleh publik melalui media sosial, yang merupakan informasi yang mencerminkan perspektif masyarakat terhadap kebijakan tertentu (Yulita et al., 2023).

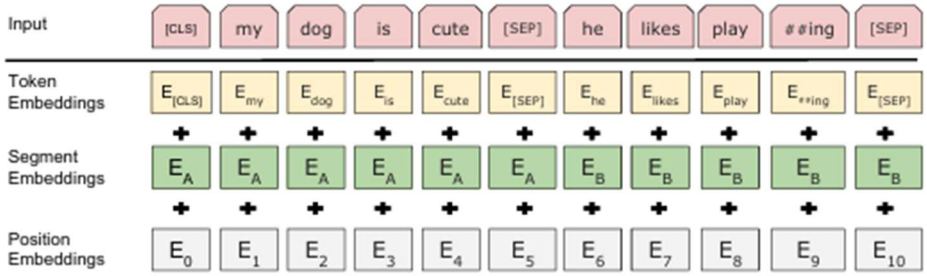
1.2.2 *BERT*

BERT (*Bidirectional Encoder Representations from Transformers*) merupakan *framework machine learning open-source* untuk pemrosesan bahasa alami (*Natural Language Processing*) yang dikembangkan oleh Google pada tahun 2018 (Devlin et al., 2019). BERT sesuai dengan namanya *bidirectional encoder representations from transformers* merupakan representasi *encoder* dari transformer, tujuan dari BERT adalah untuk membuat sebuah *language model*. Oleh karena itu, BERT hanya membutuhkan bagian *encoder* dari *transformer* dan tidak memerlukan bagian *decoder*. BERT dibangun berdasarkan arsitektur *transformer encoder*, yang terdiri dari beberapa lapisan. Setiap lapisan *encoder* memiliki dua sub-lapisan: mekanisme *multi-head attention* dan jaringan *feed forward* terhubung sepenuhnya. Dalam setiap lapisan, *input* melewati *self-attention* untuk memahami konteks secara lebih luas sebelum diteruskan ke jaringan *feed forward* untuk langkah berikutnya (Devlin et al., 2019; Vaswani et al., 2023). Model BERT memiliki dua versi yaitu *BERT-Base* dan *BERT-Large* yang memiliki perbedaan pada ukuran model, *BERT-Base* memiliki 12 *layer* atau encoder block yang disusun dan memiliki 110 juta *parameters* dan *BERT-Large* memiliki 24 *layer* atau *encoder block* yang disusun dan memiliki 340 juta *parameters*. Pada penilitian ini versi model BERT yang digunakan adalah versi *BERT-Base* dengan *12 layers*. Arsitektur *transformer encoder* dapat dilihat pada gambar 1 di bawah ini.



Gambar 1 Arsitektur *Encoder* pada*Transformer* (Vaswani et al., 2023)

Setiap *layers* atau *encoder* pada model BERT terdiri dari 2 bagian utama yaitu *multi-head attention* dan *feed forward*. Sebelum data input masuk dan diproses di setiap *layer* data *input* akan melalui proses *input embedding* yang dapat dilihat pada gambar 1. *Input embedding* pada BERT merupakan kombinasi dari tiga jenis *embedding* yang berbeda yaitu *token embeddings*, *segment embeddings*, dan *positional embeddings* (Devlin et al., 2019).



Gambar 2 Representasi *Input BERT* (Devlin et al., 2019)

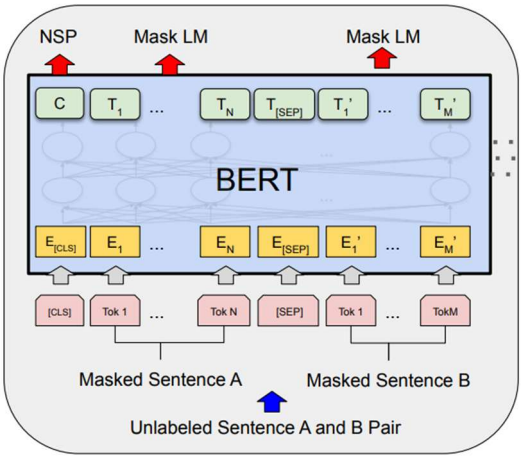
Setiap kalimat dalam *BERT* diawali dengan token khusus [CLS] untuk klasifikasi, dan diakhiri dengan token [SEP] untuk pemisahan antar kalimat. Dapat dilihat pada gambar 2 *token embeddings input token* diubah menjadi vektor yang menyimpan data semantik, *token embeddings* pada BERT menggunakan *pre-trained embeddings* yaitu *WordPieces* yang memiliki 30K *vocabulary tokens*. Pada *segment embeddings input token* diberi kode untuk membedakan kalimat satu dengan kalimat lainnya berdasarkan *token* [SEP] dan dapat dilihat pada gambar 2 *token* dengan kode A dan *token* dengan kode B dipisahkan oleh token [SEP]. Terakhir, *position embeddings token* diberi kode untuk mengetahui posisi *token* pada *input* data. Dengan menggabungkan tiga jenis

embeddings akan menghasilkan *input embedding* yang akan masuk ke *layer* atau *encoder* pada model BERT.

Pada *layer encoder*, *input embedding* pertama melalui mekanisme *multi-head attention* dimana setiap *token* dibuatkan representasi *attention vectors* yang memiliki informasi relasi kontekstual setiap kata pada kata yang lain, setelah melewati *multi-head attention* dilakukan proses *Add & Norm* dimana proses ini merujuk pada dua langkah proses yang dikenal sebagai *residual connection* dan *layer normalization*, pada *residual connection output* dari *sub-layer* dalam hal ini adalah *output* dari *multi-head attention* ditambahkan kembali dengan *input* asli dengan tujuan untuk menghindari masalah yang disebut *vanishing gradient*, setelah penambahan tersebut normalisasi dilakukan dengan menyesuaikan skala dan translasi dari *output* agar memiliki *mean* 0 dan *varians* 1, hal ini untuk membantu memastikan bahwa dari setiap *sub-layer* memiliki rentang nilai yang konsisten, yang membantu model belajar dengan lebih stabil dan efektif.

Setelah proses *multi-head attention output* yang berupa *attention vectors* melalui *sub-layer feed forward* dimana pada proses ini *attention vectors* diubah menjadi bentuk yang lebih mudah untuk di proses pada *layer* selanjutnya. Pada model *BERT-Base* yang memiliki 12 *layer* proses ini akan dilakukan sebanyak 12 kali sesuai dengan ukuran dari model.

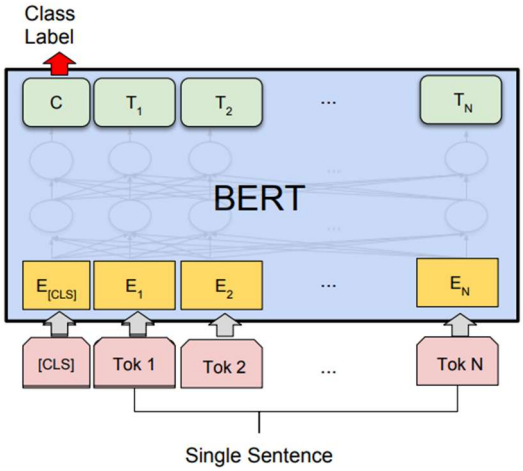
Pada tahap pertama yaitu tahap *pre-training* BERT kita memerlukan agar model BERT dapat memahami bahasa dengan baik, untuk mencapai tujuan tersebut model BERT dilatih dengan menggunakan *unlabeled datasets* yang besar untuk menyelesaikan dua tugas yaitu *Masked Language Model* (MLM) dan *Next Sentence Prediction* (NSP). Dimana pada tugas MLM *input token* secara acak diberi *mask* untuk menyembunyikan nilai asli dari token tersebut dan kemudian model dilatih untuk memprediksi kata yang benar pada token yang diberi mask, tugas ini dilakukan agar model dapat memahami bahasa dalam level kata dengan lebih baik. Kemudian pada tugas kedua yaitu NSP *input embeddings* terdiri dari dua kalimat yaitu kalimat A dan kalimat B dan model akan dilatih untuk memprediksi apakah kalimat B merupakan lanjutan dari kalimat A, tugas ini dilakukan agar model dapat memahami bahasa pada level kalimat dengan lebih baik.



Gambar 3 Arsitektur *Pre-training* BERT (Devlin et al., 2019)

Arsitektur *pre-training* BERT dapat dilihat pada gambar 3 dimana *input token* terdiri dari dua kalimat yang dipisahkan oleh *token* [SEP] dan salah satu *token* diberi *mask* yang yang ditandai dengan nama *TokM*, *token* kemudian di *embeddings* dan diumpan kedalam BERT *neural network* yang terdiri dari *layer encoder transformer*. Pada *output layer* untuk *token* yang disamarkan BERT akan memprediksi nilai asli dari *token* tersebut, ini adalah hasil dari tugas MLM. Untuk tugas NSP, *output* dari token [CLS] digunakan untuk menjadi representasi vektor yang menentukan apakah kalimat B merupakan lanjutan dari kalimat A. Dengan menjalankan dua tugas tersebut secara bersamaan BERT dapat mempelajari dan memahami bahasa dengan konteks lebih baik.

Pada tahap kedua, dilakukan *fine-tuning* pada *pre-training* model dengan menambahkan *output layer* berupa *classification layer* untuk mengklasifikasi kelas sentiment dan dengan melatih model menggunakan dataset yang telah dilabel sebelumnya dengan kelas positif, negatif, dan netral.



Gambar 4 *Arsitektur fine-tuning BERT* (Devlin et al., 2019)

Dapat dilihat pada gambar 4 merupakan arsitektur fine-tuning BERT untuk tugas sentiment analisis, pada kasus sentiment analisis data input yang dimasukkan dalam bentuk satu kalimat dan dipecah setiap kata menjadi *token* kemudian *token* di *embeddings* dan diumpan ke BERT *Neural Network* dan pada *output layer* BERT *node C* merupakan representasi vektor dari konteks data yang di *input* yang kemudian akan dimasukkan kedalam *classification layer* untuk menentukan kelas dari data tersebut (Devlin et al., 2019).

1.2.3 *IndoBERT* (Indonesian Bidirectional Encoder Representations from Transformers)

IndoBERT merupakan adaptasi dari model BERT (*Bidirectional Encoder Representations from Transformers*) yang dirancang khusus untuk menangani teks berbahasa Indonesia. Dikembangkan untuk mengatasi kebutuhan pemrosesan bahasa alami (*Natural Language Processing*) dalam konteks bahasa Indonesia, IndoBERT dilatih menggunakan korpus teks yang luas dan beragam dari sumber-sumber berbahasa Indonesia, termasuk berita, media sosial, dan dokumen bahasa Indonesia

lainnya (Wilie et al., 2020). IndoBERT dilatih dengan menggunakan lebih dari 220 juta kata dari tiga sumber utama, yaitu Wikipedia bahasa Indonesia (74 juta kata), artikel berita dari Kompas, Tempo, dan Liputan6 (55 juta kata), dan Korpus Web Indonesia (90 juta kata) (Koto et al., 2020). Model ini adalah suatu *pre-trained model monolingual* dalam bahasa Indonesia yang mengikuti konfigurasi dari BERT-Base sehingga struktur dari IndoBERT sama dengan BERT. Konfigurasi standar untuk BERT-Base yang memiliki 12 *hidden layer*, *12 attention head*, dan *feed-forward hidden layer*. IndoBERT menggunakan *transformer encoder* untuk mempelajari hubungan antar kalimat (Koto et al., 2020).

1.2.4 *XLNet*

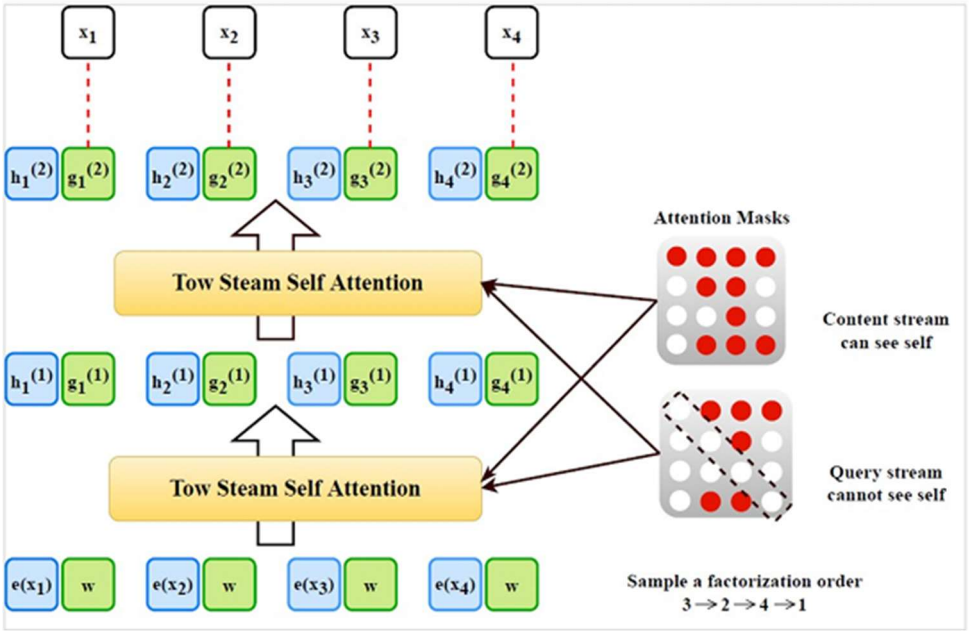
XLNet adalah model bahasa yang telah dilatih sebelumnya dan menggabungkan keunggulan pependekatan autoregressive seperti GPT dan pemodelan konteks dua arah seperti BERT. Model ini dikenal karena kemampuannya untuk menangkap konteks dua arah secara efektit (Yang et al., 2019). *XLNet* juga merupakan model bahasa berbasis transformasi, tetapi memperkenalkan pendekatan pelatihan berbasis permutasi yang memeriksa kata-kata dalam urutan yang berbeda dan dilatih dengan kumpulan data yang lebih besar. Model *XLNet-base-cased* terdiri dari 12 lapisan *transformator* dan 12 *attention head* dengan 768 unit tersembunyi yang menghasilkan 110 juta parameter. Pendekatan ini memungkinkan *XLNet* untuk memahami teks dari kedua arah dengan lebih efisien, sehingga menangkap pola kompleks dalam data teks (Chatzimina et al., 2024).

Arsitektur XLNet menggabungkan konteks dua arah dan pengodean posisional sambil memanfaatkan mekanisme perhatian baru yang dikenal sebagai Two-Stream Self-Attention. Seperti BERT, XLNet juga mengikuti proses dua langkah: pra-pelatihan dan penyempurnaan, tetapi XLNet memperkenalkan inovasi utama—Permutation Language Modeling (PLM)—yang mempertimbangkan semua kemungkinan permutasi dari urutan input selama pra-pelatihan. Pendekatan unik ini memungkinkan XLNet untuk menangkap lebih banyak informasi konstekstual dan mengatasi keterbatasan BERT terletak pada penggunaan token [MASK] yang tidak muncul saat fine-tuning, bukan pada arah konteksnya (Apu et al., 2025).

Pre-Training XLNet: Dalam fase pra-pelatihan, XLNet dilatih pada korpus besar teks tak berlabel menggunakan Permutation Language Modeling (PLM), yang berbeda dari Masked Language Modeling BERT. PLM mengubah urutan faktorisasi prediksi token, bukan urutan token itu sendiri, dan XLNet belajar memprediksi token berikutnya dalam urutan tersebut berdasarkan permutasi ini. Gambar 5 menunjukkan strategi kerja model bahasa Permutasi (PLM). Pendekatan ini memungkinkan XLNet untuk memanfaatkan semua konteks yang mungkin dalam urutan tersebut, sehingga menawarkan pemahaman yang lebih fleksibel dan kuat terhadap teks tersebut. Tidak seperti BERT, yang bergantung pada arah kiri-ke-kanan atau kanan-ke-kiri yang tetap, model berbasis permutasi XLNet meningkatkan kinerjanya pada tugas-tugas hilir dengan menangkap hubungan antar-token dengan lebih baik (Apu et al., 2025).

Fine-Tuning XLNet: Setelah prapelatihan, XLNet disempurnakan pada kumpulan data berlabel tertentu untuk tugas-tugas hilir, termasuk klasifikasi teks, tanya jawab, dan analisis sentimen. Fase ini mengadaptasi pemahaman bahasa umum yang dicapai

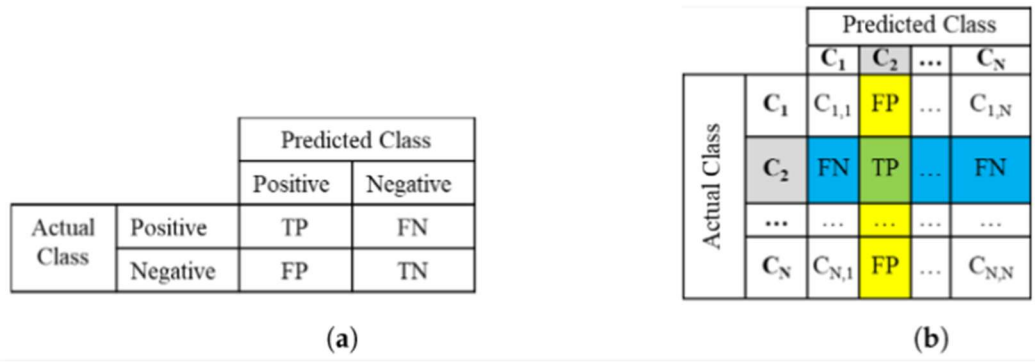
dalam prapelatihan untuk melakukan tugas-tugas yang memerlukan pengetahuan dan interpretasi teks yang lebih terarah (Apu et al., 2025).



Gambar 5 Arsitektur XLnet (Apu et al., 2025)

1.2.5 *Confusion Matrix*

Confusion Matrix digunakan untuk menentukan efektivitas pemodelan klasifikasi, yang terdiri dari baris dan kolom yang membentuk tabel yang berisi dari data uji prediksi (Salih & Abdulazeez, 2021). *Multiclass confusion matrix* adalah alat evaluasi yang digunakan dalam pemodelan dan pembelajaran mesin ketika memiliki lebih dari dua kelas pada klasifikasi. Gambar 5 menampilkan tabel *confusion matrix binary classification* dan *multiclass classification*.



Gambar 6 Confusion Matrix (a) Binary Classification (b) Multiclass Classification (Markoulidakis et al., 2021)

Confusion Matrix yang terdiri dari *True Positive* (TPos), *True Neutral* (TNet), dan *True Negative* (TNeg) merupakan prediksi yang benar berdasarkan data aktual. *False positive* (Fpos) merupakan error dimana data aktual yang berlabel *negative* atau *neutral*

diprediksi *positive*. *False neutral* (Fnet) merupakan error dimana data aktual yang berlabel *positive* atau *negative* diprediksi *neutral*. *False negative* (Fneg) merupakan error dimana data aktual yang berlabel *positive* atau *neutral* diprediksi *negative* (Markoulidakis et al., 2021).

Tabel 1 *Confusion Matrix Multiclass*

		Prediksi		
Aktual		Positif	Netral	Negatif
	Positif	TPos	FPosNet	FPosNeg
	Netral	FNetPos	TNet	FNetNeg
	Negatif	FNegPos	FNegNet	TNeg

Tabel 1 adalah contoh dari *confusion matrix multiclass* terhadap nilai aktual dan prediksi, tabel tersebut digunakan untuk menampilkan nilai *True positive* (TP), *False positive* (FP), *True negative* (TN), dan *False Negative* (FN)(Pahrul, 2023) (Amaliyah Muzakkir, 2023).

1. TPos (*True Positive*): merupakan nilai prediksi dan aktualnya yang bersifat positif. Variabel ini mewakili jumlah prediksi di mana sentimen positif diklasifikasikan dengan benar [sebagai sentimen positif]. Ini juga merupakan *True Positive* untuk kelas positif.
2. FPosNet (*False Positive Neutral*): merupakan nilai prediksi bersifat netral sedangkan nilai aktualnya bersifat positif. Variabel ini mewakili jumlah prediksi di mana sentimen positif salah diklasifikasikan sebagai sentimen netral.
3. FPosNeg (*False Positive Negative*): merupakan nilai prediksi bersifat negatif sedangkan nilai aktualnya yang bersifat positif. Variabel ini mewakili jumlah prediksi di mana sentimen positif salah diklasifikasikan sebagai sentimen negatif.
4. FNetPos (*False Neutral Positive*): merupakan nilai prediksi bersifat positif sedangkan nilai aktualnya yang bersifat netral. Variabel ini mewakili jumlah prediksi di mana sentimen netral salah diklasifikasikan sebagai sentimen positif.
5. TNet (*True Neutral*): merupakan nilai prediksi dan aktualnya yang bersifat netral. Variabel ini mewakili jumlah prediksi di mana sentimen netral diklasifikasikan dengan benar [sebagai sentimen Netral]. Ini juga merupakan *True Positive* untuk kelas *neutral*.
6. FNetNeg (*False Neutral Negative*): merupakan nilai prediksi bersifat negatif sedangkan nilai aktualnya yang bersifat netral. Variabel ini mewakili jumlah prediksi di mana sentimen netral salah diklasifikasikan sebagai sentimen negatif.
7. FNegPos (*False Negative Positive*): merupakan nilai prediksi bersifat positif sedangkan nilai aktualnya yang bersifat negatif. Variabel ini mewakili jumlah prediksi di mana sentimen negatif salah diklasifikasikan sebagai sentimen positif.
8. FNegNet (*False Negative Neutral*): merupakan nilai prediksi bersifat netral sedangkan nilai aktualnya yang bersifat negatif. Variabel ini mewakili jumlah prediksi di mana sentimen negatif salah diklasifikasikan sebagai sentimen netral.
9. TNeg (*True Negative*): merupakan nilai prediksi dan aktualnya yang bersifat negatif. Variabel ini mewakili jumlah prediksi di mana sentimen negatif diklasifikasikan

dengan benar [sebagai sentimen negatif]. Ini juga merupakan *True Positive* untuk kelas negatif.

Pada *Confusion Matrix*, ada 4 istilah yang digunakan untuk merepresentasikan hasil klasifikasi. Keempat istilah tersebut adalah:

1) *True Positve* (TP), merupakan data yang bernilai positif dan hasilnya diprediksi positif (Pahrul, 2023). Berikut adalah nilai TP dari masing-masing kelas:

   Tabel 2 Nilai *True Positive*

		Prediksi		
Aktual		Positif	Netral	Negatif
	Positif	TPos	FPosNet	FPosNeg
	Netral	FNetPos	TNet	FNetNeg
	Negatif	FNegPos	FNegNet	TNeg

   Nilai *True Positive*:
   a. TPpositif = TPos
   b. TPnetral = TNet
   c. TPnegatif = TNeg

2) *False Positive* (FP), merupakan data yang bernilai negatif, namun hasilnya diprediksi sebagai data yang bernilai positif (Pahrul, 2023). Berikut adalah nilai FP pada masing-masing kelas:

   Tabel 3 Nilai *False Positive*

		Prediksi		
Aktual		Positif	Netral	Negatif
	Positif	TPos	FPosNet	FPosNeg
	Netral	FNetPos	TNet	FNetNeg
	Negatif	FNegPos	FNegNet	TNeg

   Nilai *False Positive*:
   a. FPpositif = FNetPos + FNegPos
   b. FPnetral = FPosNet + FNegNet
   c. FPnegatif = FPosNeg + FNetNeg

3) *True Negative* (TN), merupakan data yang bernilai negatif dan hasilnya diprediksi negatif (Pahrul, 2023). Berikut adalah nilai TN pada masing-masing kelas :

   Tabel 4 Nilai *True Negative*

		Prediksi		
Aktual		Positif	Netral	Negatif
	Positif	TPos	FPosNet	FPosNeg
	Netral	FNetPos	TNet	FNetNeg
	Negatif	FNegPos	FNegNet	TNeg

True Negative untuk kelas positif adalah:
a. TNpositif = TNet + FNetNeg + FNegNet + TNeg
True Negative untuk kelas netral adalah:
a. TNnetral = TPos + FPosNeg + FNegPos + TNeg
True Negative untuk kelas negatif adalah:
a. TNnegatif = TPos + FPosNet + FNetPos + Tnet

4) *False Negative* (FN), merupakan data yang bernilai positif, namun hasilnya diprediksi sebagai data yang bernilai negatif (Pahrul, 2023). Berikut adalah nilai FN dari masing-masing kelas:

Tabel 5 Nilai *False Negative*

		Prediksi		
Aktual		Positif	Netral	Negatif
	Positif	TPos	FPosNet	FPosNeg
	Netral	FNetPos	TNet	FNetNeg
	Negatif	FNegPos	FNegNet	TNeg

Nilai *False Negative*:
a. FNpositif = FPosNet + FPosNeg
b. FNnetral = FNetPos + FNetNeg
c. FNnegatif = FNegPos + FNegNet

Untuk mengevaluasi kinerja model menggunakan empat aspek penilaian, yaitu *accuracy*, *precision*, *recall*, dan *F1 score*.

1. Akurasi
   Akurasi adalah rasio sampel sentimen yang diklasifikasikan dengan benar terhadap jumlah total sampel (Alzoman & Alenazi, 2021).
   $$Accuracy = \frac{TP+TN}{TP+TN+FP+F} \quad (1)$$
   Berikut adalah rumus untuk menghitung *accuracy* pada *confusion matrix multiclass* (Amaliyah Muzakkir, 2023)
   $$\frac{\text{TPos+TNet+TNe}}{\text{TPos+FPosNet+FPosNeg+FNetPos+TNet+FNetNeg+FNegPos+FNegNet+TNeg}} \quad (2)$$
   a. TPos = *True Positive*
   b. TNet = *True Neutral*
   c. TNeg = *True Negative*

2. Presisi
   Presisi adalah ukuran rasio positif, kelas sentimen yang diprediksi dengan benar terhadap jumlah total prediksi klasifikasi positif (Alzoman & Alenazi, 2021).
   $$Precision = \frac{TP}{TP+FP} \quad (3)$$
   Berikut adalah rumus untuk menghitung presisi positif, netral, dan negatif pada *confusion matrix multiclass* (Amaliyah Muzakkir, 2023)
   $$Precision\ (Positive\ (0)) = \frac{\text{TPos}}{\text{TPos+FNetPos+FNegPo}} \quad (4)$$

$$Precision\ (Neutral\ (1)) = \frac{\text{TNet}}{\text{TNet+FPosNet+FNegNet}} \quad (5)$$

$$Precision\ (Negative\ (2)) = \frac{\text{TNeg}}{\text{TNeg+FPosNeg+FNetNeg}} \quad (6)$$

Setiap kelas akan dihitung menggunakan rumus diatas lalu dihitung rata-ratanya menggunakan rumus:

$$\frac{\text{Precision label 0+Precision label 1+Precision label 2}}{3}\ x\ 100\% \quad (7)$$

3. Recall

Recall mengukur rasio positif aktual, kelas sentimen yang diprediksi dengan benar (Alzoman & Alenazi, 2021).

$$Recall = \frac{TP}{TP+FN} \quad (8)$$

Berikut adalah rumus untuk menghitung *recall* positif, netral, dan negatif pada *confusion matrix multiclass* (Amaliyah Muzakkir, 2023)

$$Recall\ (Positive\ (0)) = \frac{\text{TPos}}{\text{TPos+FPosNet+FPosNeg}} \quad (9)$$

$$Recall\ (Netral\ (1)) = \frac{\text{TNet}}{\text{TNet+FNetPos+FNetNeg}} \quad (10)$$

$$Recall\ (Negative\ (2)) = \frac{\text{TNeg}}{\text{TNeg+FNegPos+FNegNet}} \quad (11)$$

Setiap kelas akan dihitung menggunakan rumus diatas lalu dihitung rata-ratanya menggunakan rumus:

$$\frac{\text{Recall label 0+Recall label 1+Recall label 2}}{3}\ x\ 100\% \quad (12)$$

4. F1 Score

F1 Score mengukur presisi rata-rata dan memori (Alzoman & Alenazi, 2021).

$$F1\ Score = \frac{2\ x\ Precision\ x\ Recall}{Precision+Re} \quad (13)$$

Berikut adalah rumus untuk menghitung *F1-Score* positif, netral, dan negatif pada *confusion matrix multiclass* (Amaliyah Muzakkir, 2023)

$$F1-score\ \left(Positive\ (0)\right) = 2\ X\ \frac{\text{RecallPos x PrecisionPos}}{\text{RecallPos+PrecisionPos}} \quad (14)$$

$$F1-score\ \left(Netral\ (1)\right) = 2\ X\ \frac{\text{RecallNet x PrecisionNet}}{\text{RecallNet+PrecisionNet}} \quad (15)$$

$$F1-score\ \left(Negative\ (2)\right) = 2\ X\ \frac{\text{RecallNeg x PrecisionNeg}}{\text{RecallNeg+PrecisionNeg}} \quad (16)$$

Setiap kelas akan dihitung menggunakan rumus diatas lalu dihitung rata-ratanya menggunakan rumus:

$$\frac{\text{F1−score label 0+F1−score label 1+F1−score label 2}}{3}\ x\ 100\% \quad (17)$$

Perbedaan antara *macro average* dan *weight average* adalah *macro average* menghitung matriks secara bebas untuk setiap kelas kemudian mengambil rata-ratanya. Sedangkan *weight average* menghitung rata-rata dengan memperhitungkan bobot pada setiap datanya (Amaliyah Muzakkir, 2023). Adapun rumus untuk mencari *weight average* f1-score sebagai berikut:

$$\frac{\text{F1−score (0)x W0+F1−scor (1) x W1+F1−score (2) } x\ W2}{\text{(W0+W1+W2)}}\ x\ 100\% \quad (18)$$

Adapun w0, w1, dan w2 merupakan bobot dari masing-masing label positif, netral, dan negatif.

1.3 Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dijelaskan di atas maka rumusan masalah dalam penelitian ini adalah:

1. Bagaimana cara mengimplementasikan model *IndoBERT* dan *XLNet* dalam analisis sentimen terhadap kinerja presiden dan wakil presiden?
2. Bagaimana Tingkat performa model *IndoBERT* dan *XLNet* dalam analisis sentimen terhadap kinerja presiden dan wakil presiden?

1.4 Tujuan Penelitian

Penelitian ini bertujuan untuk:

1. Mengetahui cara mengimplementasikan model *IndoBERT* dan *XLNet* dalam analisis sentimen terhadap kinerja presiden dan wakil presiden.
2. Mengetahui Tingkat performa model IndoBERT dan XLNet dalam analisis sentimen terhadap kinerja presiden dan wakil presiden.

1.5 Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat sebagai berikut:

1. Penelitian ini memberikan wawasan praktis dalam penerapan model *IndoBERT* dan *XLNet* untuk analisis sentimen terhadap kinerja presiden dan wakil presiden.
2. Penelitian ini meningkatkan kesadaran tentang kinerja presiden dan wakil presiden.

1.6 Ruang Lingkup

Penelitian ini memiliki beberapa batasan, yaitu:

1. Kumpulan data yang digunakan bersumber dari media sosial X (Twitter) dan komentar dari beberapa video Youtube yang membahas kinerja presiden dan wakil presiden.
2. Data yang dianalisis berbahasa Indonesia.
3. Penelitian ini akan menganalisis sentimen ke dalam tiga kategori, yaitu netral, positif, dan negatif.

BAB II

METODE PENELITIAN

2.1 Lokasi Penelitian

Penelitian ini dilakukan sejak bulan November 2024 di Laboratorium *Ubiquitous Computing and Networking* Departemen Teknik Informatika Universitas Hasanuddin.

2.2 Instrumen Penelitian

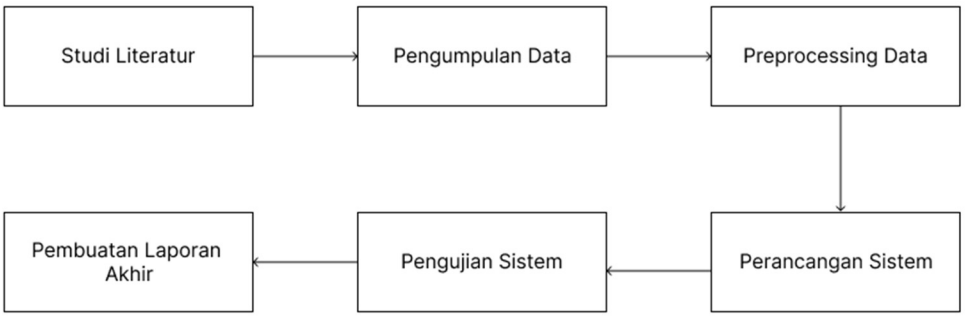
Instrumen yang digunakan dalam penelitian ini adalah sebagai berikut:
Tabel 6 Spesifikasi *software*

Komponen	Spesifikasi
OS	Windows 11
Bahasa	Python
Framework	Flask, Bootstrap
Tools	Google Collaboratory (GPU T4), Microsoft Excel, Google Drive, Browser, Visual Studio Code

Tabel 7 Spesifikasi *hardware*

Komponen	Spesifikasi
Jenis	Asus TUF Gaming FX505GD
Processor	IntelCore i7
RAM	8GB

2.3 Tahapan Penelitian



Gambar 7 Tahapan penelitian

2.3.1 Studi literatur

Langkah pertama dalam penelitian ini adalah melakukan studi literatur untuk mengumpulkan informasi, data, dan rujukan dari berbagai sumber seperti buku, artikel, dan jurnal yang berkaitan dengan objek penelitian. Objek penelitian tersebut meliputi analisis sentimen, kinerja pemerintahan, dan dua algoritma yang akan digunakan, yaitu

IndoBERT dan XLNet. Melalui kegiatan studi literatur, diharapkan dapat memperoleh pemahaman yang lebih mendalam mengenai objek penelitian serta mengumpulkan berbagai referensi yang nantinya dapat digunakan sebagai bahan acuan dalam melakukan penelitian lebih lanjut.

2.3.2 Pengumpulan Data (Data Collection)

Pada tahap ini dilakukan proses pengumpulan data, data dikumpulkan melalui proses scrape data menggunakan bahasa pemrograman python melalui platform Google Collaboratory. Dalam proses scrape data, dilakukan pengambilan data secara otomatis dari sumber yang telah ditentukan yaitu Twitter dan Youtube. Data-data yang berhasil diambil kemudian disimpan dalam format yang sesuai dan dapat digunakan pada penelitian selanjutnya. Melalui tahap pengumpulan data ini, diharapkan dapat memperoleh data yang diperlukan untuk melakukan penelitian secara lebih terperinci dan akurat.

2.3.3 *Preprocessing* Data (Data *Praproses*)

Tahap selanjutnya setelah pengumpulan data adalah *Preprocessing* data, yang bertujuan untuk memastikan bahwa data yang akan digunakan untuk membangun model adalah data yang bersih dan siap untuk dianalsis. Proses *preprocessing* meliputi beberapa tahapan yaitu, *case folding*, *remove punctuation*, *stopword removal*, *normalization*, *stemming*, dan *tokenization*. Dengan melakukan preprocessing data secara benar, maka akan memungkinkan penggunaan data yang lebih efektif dan efisien dalam membangun model yang berkualitas.

2.3.4 Perancangan Sistem

Setelah memastikan bahwa data yang akan digunakan telah melalui preprocessing data yang sesuai, langkah selanjutnya yang akan diambil adalah melakukan perancangan sistem. Dalam hal ini, proses perancangan sistem akan dilakukan dengan mengacu pada rancangan sistem yang telah dibuat.

2.3.5 Pengujian Sistem

Pengujian sistem bertujuan untuk melakukan evaluasi terhadap kinerja sistem yang telah dibangun, sehingga dapat memberikan gambaran yang jelas mengenai seberapa baik sistem tersebut berjalan sesuai dengan yang diharapkan. Pengujian sistem dilakukan dengan menggunakan metrik evaluasi yaitu dengan menghitung *accuracy*, *recall*, *precision*, dan *f1-score*.

2.3.6 Pembuatan Laporan Akhir

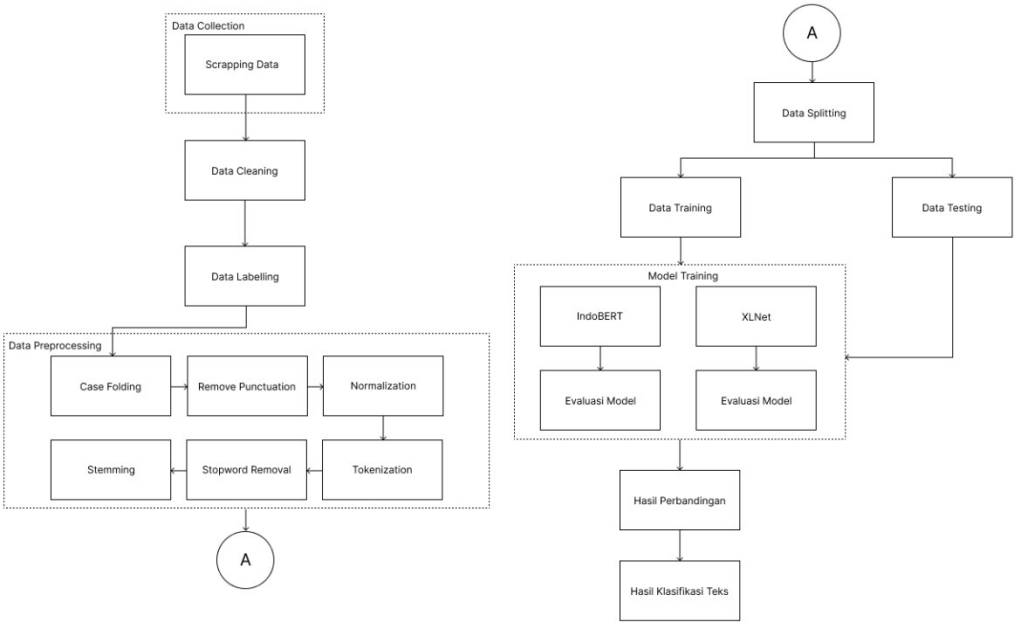
Pada tahap akhir penelitian, dilakukan proses pembuatan dokumentasi dalam bentuk sebuah laporan yang berisi hasil penelitian, temuan yang diperoleh, kesimpulan, serta rekomendasi untuk penelitian selanjutnya. Proses pembuatan laporan dirancang secara sistematis dan mudah dipahami dengan mengumpulkan data dan informasi dari seluruh tahapan penelitian, disertai analisis data serta pembahasan terhadap temuan yang diperoleh selama proses penelitian.

2.4 Teknik Pengumpulan Data

Pada tahap pengumpulan data, dilakukan proses scrapping data menggunakan library Python Bernama Tweet Harvest / snscrape dan YouTube Data API v3. Library tersebut menyediakan sebuah API yang sangat memudahkan dalam melakukan crawling data pada Twitter dan Youtube tanpa memerlukan external dependencies lainnya. Proses scrapping data dilakukan pada data-data yang terdapat pada ulasan atau komentar dari pengguna aplikasi Twitter dan Youtube.

2.5 Rancangan Sistem

Rancangan sistem dapat dilihat pada diagram alur berikut:



Gambar 8 Rancangan sistem

Gambar 8 menampilkan rancangan sistem penelitian. Penelitian ini akan membangun sebuah sistem klasifikasi menggunakan dua model klasifikasi berbeda, yaitu IndoBERT dan XLNet. Sistem ini akan digunakan untuk melakukan klasifikasi sentimen opini publik pada aplikasi x (twitter) dan youtube.

Selanjutnya, penelitian ini akan melakukan evaluasi model untuk membandingkan hasil sentimen dari komposisi data latih dan data uji yang sama. Setelah mendapatkan hasil klasifikasi dari 2 model, peneliti akan membandingkan kinerja masing-masing model dan model mana yang lebih unggul untuk melakukan klasifikasi sentimen opini publik pada aplikasi x (twitter) dan youtube. Setelah model terbaik terpilih, sistem ini akan digunakan untuk memberikan prediksi sentimen. Dengan menggunakan model klasifikasi yang akurat, diharapkan prediksi sentimen yang dihasilkan juga akurat.

2.6 Data *Scrapping* (Pengumpulan Data)

Pada tahap ini dilakukan proses pengumpulan data, data dikumpulkan melalui proses scrape data menggunakan bahasa pemrograman python melalui platform Google Collaboratory. Proses scrapping data menggunakan library Python Bernama Tweet Harvest / snscrape dan Youtube Data API v3. Library tersebut menyediakan sebuah API yang sangat memudahkan dalam melakukan crawling data pada Twitter dan Youtube tanpa memerlukan external dependencies lainnya. Proses scrapping data dilakukan pada data-data yang terdapat pada ulasan atau komentar dari pengguna aplikasi Twitter dan Youtube. Data-data yang berhasil diambil kemudian disimpan dalam format yang sesuai dan dapat digunakan pada penelitian selanjutnya. Melalui tahap pengumpulan data ini, diharapkan dapat memperoleh data yang diperlukan untuk melakukan penelitian secara lebih terperinci dan akurat.

2.7 Data *Cleaning* (Pembersihan Data)

Setelah data terkumpul selanjutnya adalah pembersihan data. Pada proses ini data yang akan dibersihkan seperti tidak akurat, tidak lengkap, atau tidak konsisten dalam dataset. Tujuan utamanya adalah untuk memastikan bahwa data yang akan digunakan untuk analisis atau pelatihan model adalah berkualitas tinggi dan dapat diandalkan.

2.8 Data *Labelling* (Pelabelan Data)

Setelah melakukan pembersihan data, setiap ulasan akan diberi label secara manual untuk membagi kelas menjadi kelas positif, netral, dan negatif. Kelas positif akan diberi label 0, kelas netral akan diberi label 1, dan kelas negatif akan diberi label 2. Penggunaan label sentimen pada setiap ulasan penting dilakukan agar pemodelan analisis sentimen dapat memberikan hasil yang akurat dan sesuai dengan harapan. Namun, pembuatan label sentimen pada setiap ulasan tidak boleh dilakukan secara sembarangan. Pelabelan data yang tidak konsisten dapat menjadi penyebab buruknya hasil pemodelan, bahkan bisa menghasilkan tingkat akurasi yang rendah. Oleh karena itu, kualitas label data menjadi salah satu faktor penentu agar sebuah sentimen menghasilkan tingkat akurasi yang tinggi.

2.9 Data *Preprocessing*

Setelah melakukan pengumpulan, pembersihan, dan pelabelan data, maka proses selanjutnya adalah data *preprocessing*. Data *preprocessing* adalah suatu proses untuk membersihkan, mengorganisasi, dan menyiapkan data mentah agar dapat digunakan secara efektif dalam proses analisis atau pemodelan. Dengan melakukan data *preprocessing* data, data teks dapat menjadi terstruktur dan menghilangkan informasi yang tidak relevan agar dapat mempermudah proses analisis sentimen dan kinerja model dalam memahami dan mengklasifikasikan sentimen teks. Berikut adalah beberapa tahapan *preprocessing* yang digunakan pada penelitian ini.

1. *Case folding*
   Case folding dilakukan untuk merubah semua huruf kapital (*uppercase*) menjadi huruf kecil (*lowercase*) yang bertujuan untuk membuat semua karakter dataset

memiliki format yang sama, yaitu menggunakan huruf kecil. Berikut adalah kode yang digunakan pada proses *case folding*.

```
def case_folding(text):
  if isinstance(text, str):
    lowercase_text = text.lower()
    return lowercase_text
  else:
    return text

df['case_folding'] = df['text'].apply(case_folding)
```

Sebagai contoh terdapat sentimen “Langkah cepat Pemerintahan @prabowo @gibran dlm merespon Krisis Patut diapresiasi... Salut! #IndonesiaMaju!!”. Setelah melalui proses case folding maka kalimat akan menjadi “langkah cepat pemerintahan @prabowo @gibran dlm merespon krisis patut diapresiasi... salut! #indonesiamaju!!”.

2. *Remove punctuation*

Tujuan dilakukan remove punctuation adalah penghapusan tanda baca dari teks untuk memastikan bahwa simbol-simbol non-alfabet tidak mempengaruhi analisis lebih lanjut. Selain itu, tahap ini juga mencakup pemrosesan kode ASCII, di mana karakter-karakter khusus yang tidak relevan dengan makna teks, seperti simbol atau tanda khusus, diidentifikasi dan dihapus guna menghasilkan teks yang lebih bersih dan siap untuk diproses oleh model NLP atau algoritma pembelajaran mesin. Berikut adalah kode yang digunakan pada proses *remove punctuation*.

```
def remove_punctuation(text):
    if not isinstance(text, str):
        return ''

    # Hapus URL
    text = re.sub(r'https?://\S+', '', text)
    # Hapus tag HTML
    text = re.sub(r'<.*?>', '', text)
    # Hapus entitas HTML seperti &amp; atau &#123;
    text = re.sub(r'(&#\d+;|&[a-zA-Z]+;)', '', text)
    # Hapus mention
    text = re.sub(r'@\w+', '', text)
    # Hapus hashtag
    text = re.sub(r'#\w+', '', text)
    # Hapus emoji dan karakter non-ASCII
    text = re.sub(r'[^\x00-\x7F]+', ' ', text)
    # Hapus angka
    text = re.sub(r'\d+', '', text)
    # Hapus tanda baca
    text = text.translate(str.maketrans('', '',
string.punctuation))
```

```
    # Hapus spasi berlebih
    text = re.sub(r'\s+', ' ', text).strip()

    return text
```

Sebagai contoh terdapat sentimen “langkah cepat pemerintahan @prabowo @gibran dlm merespon krisis patut diapresiasi... salut! #indonesiamaju!!”. Setelah melalui proses remove punctuation maka kalimat akan menjadi “langkah cepat pemerintahan dlm merespon krisis patut diapresiasi salut”.

3. *Normalization*

Normalization adalah mengubah kata-kata pendek, serta kata-kata yang mengandung kesalahan pengetikan, menjadi kata-kata standar yang sesuai. *Normalization* ini bertujuan untuk meningkatkan akurasi pemahaman model terhadap teks, sekaligus memastikan bahwa kata-kata yang memiliki makna sama dianggap sebagai satu entitas yang seragam dalam analisis lebih lanjut. Pada penelitian ini digunakan kamus slang yang berisikan kata-kata normalisasi untuk memperbaiki kata. Berikut adalah contoh kata pada kamus slang normalisasi kata.

Tabel 8 Contoh Kamus Normalisasi Kata

Slang	Formal
trus	terus
udah	sudah
sdkit	sedikit
sempet	sempat

Menampilkan contoh kata-kata yang terdapat pada kamus normalisasi. Berikut adalah kode yang digunakan pada proses normalisasi.

```
# 1. Membaca file kamus alay (dengan kolom 'slang' dan
'formal')
kamus_df = pd.read_csv('drive/MyDrive/Data/kamus_alay.csv',
encoding='utf-8')  # pastikan path sesuai

# 2. Ubah jadi dictionary: {'slang': 'formal'}
kamus_normalisasi = dict(zip(kamus_df['slang'],
kamus_df['formal']))

# 3. Fungsi normalisasi teks
def normalize_text(text):
    if pd.isna(text):
        return ''

    text = str(text).lower()
    words = text.split()

    normalized_words = [kamus_normalisasi.get(word, word) for
word in words]
```

```
    normalized_text = ' '.join(normalized_words)
    normalized_text = re.sub(r'\s+', ' ',
normalized_text).strip()

    return normalized_text

def text_normalization(text):
  normalized_text = text
  return normalized_text

df['normalized_text'] =
df['remove_punctuation'].apply(text_normalization)
```

Sebagai contoh terdapat sentimen “langkah cepat pemerintahan dlm merespon krisis patut diapresiasi salut”. Setelah melalui proses *normalization* maka kalimat akan menjadi “langkah cepat pemerintahan dalam merespon krisis patut diapresiasi salut”.

4. *Tokenization*

Tokenization dilakukan untuk mengubah kalimat menjadi kata-kata (atau unit-unit lebih kecil) yang disebut token. Tokenisasi bertujuan untuk membuat representasi teks yang lebih mudah dipahami oleh komputer, di mana setiap kata dianggap sebagai satu unit yang dapat dianalisis secara independent atau dalam konteks dengan kata-kata lain di sekitarnya.

```
def tokenize(text):
    tokens = text.split()
    return tokens
```

Sebagai contoh terdapat sentimen “langkah cepat pemerintahan dalam merespon krisis patut diapresiasi salut”. Setelah melalui proses tokenization maka kalimat akan menjadi “['langkah', 'cepat', 'pemerintahan', 'dalam', 'merespon', 'krisis', 'patut', 'diapresiasi', 'salut']”.

5. *Stopword removal*

Tujuan dilakukan *stopword removal* adalah untuk menghilangkan angka serta kata-kata pendek yang tidak relevan. Penghapusan ini bertujuan untuk mengurangi *noise* dalam data dan memastikan analisis lebih fokus pada konten yang penting, terutrama saat memproses teks untuk model NLP atau algoritma pembelajaran mesin. Penelitian ini menggunakan kamus yang berisikan stop-word seperti kata “bisa”, “lagi”, “yang” dan masih banyak lagi. Berikut adalah kode yang digunakan pada proses *stopword removal*.

```
import nltk
import pandas as pd
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

!pip install Sastrawi
```

```
from Sastrawi.StopWordRemover.StopWordRemoverFactory import
StopWordRemoverFactory

# Unduh resource NLTK
nltk.download('stopwords')
nltk.download('punkt')

# 1. Ambil stopword dari NLTK (Bahasa Indonesia)
stop_words = set(stopwords.words('indonesian'))

# 2. Membaca stopword dari file stopword_removal.txt
with open('drive/MyDrive/Data/stopword_removal.txt', 'r',
encoding='utf-8') as f:
    list_stopwords = set(f.read().splitlines())

# 3. Tambahkan stopword tambahan (opsional)
tambahan_stopwords = {
    'yg', 'dg', 'rt', 'dgn', 'ny', 'd', 'klo', 'kalo', 'amp',
'biar', 'bikin', 'bilang',
    'gak', 'ga', 'krn', 'nya', 'nih', 'sih', 'si', 'tau',
'tdk', 'tuh', 'utk', 'ya',
    'jd', 'jgn', 'sdh', 'aja', 'n', 't', 'nyg', 'hehe',
'pen', 'u', 'nan', 'loh', 'yah'
}

# 4. Gabungkan semua sumber stopword (file, tambahan, dan
NLTK)
combined_stopwords =
list_stopwords.union(tambahan_stopwords).union(stop_words)

# 5. Hapus kata penting agar tidak dianggap stopword
combined_stopwords.discard('pemerintah')
combined_stopwords.discard('pemerintahan')

# 6. Fungsi untuk menghapus stopword (hasil tetap list)
def remove_stopwords(token_list):
    return [word for word in token_list if word not in
combined_stopwords]

# 7. Terapkan ke kolom 'tokenize'
df['stopword_removal'] =
df['tokenize'].apply(remove_stopwords)

# 8. Lihat hasil
df.head()
```

Sebagai contoh terdapat sentimen “['langkah', 'cepat', 'pemerintahan', 'dalam', 'merespon', 'krisis', 'patut', 'diapresiasi', 'salut']”. Setelah melalui proses stopword removal maka kalimat akan menjadi “['langkah', 'cepat', 'pemerintahan', 'merespon', 'krisis', 'patut', 'diapresiasi', 'salut']”.

6. *Stemming*

Stemming adalah merubah kata-kata yang mengandung imbuhan menjadi kata dasar dengan menghilangkan imbuhan yang terdapat di awal, tengah, akhir, dan kombinasi imbuhan dalam sebuah kata yang bertujuan untuk mengelompokkan kata-kata lain yang memiliki kata dasar dan makna serupa, namun memiliki bentuk yang berbeda karena mendapatkan imbuhan yang berbeda. Berikut adalah kode yang digunakan pada proses *stemming*.

```
# Inisialisasi stemmer
factory = StemmerFactory()
stemmer = factory.create_stemmer()

# Fungsi stemming
def stemming_text(token_list):
    return [stemmer.stem(word) for word in token_list]
```

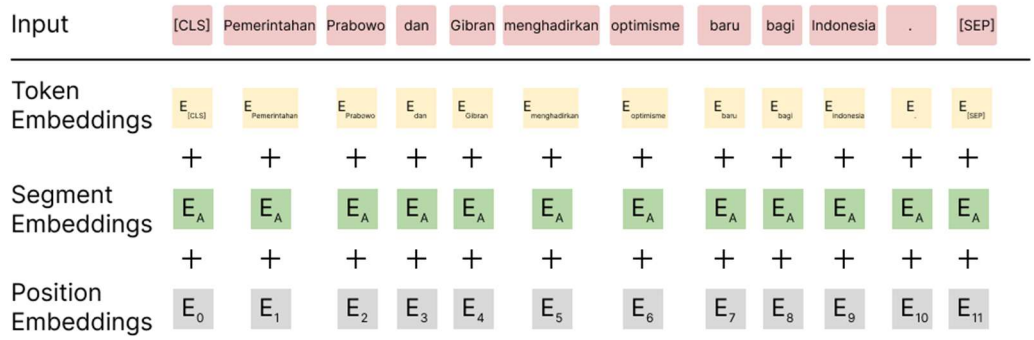
Sebagai contoh terdapat sentimen “['langkah', 'cepat', 'pemerintahan', 'merespon', 'krisis', 'patut', 'diapresiasi', 'salut']”. Setelah melalui proses stemming maka kalimat akan menjadi “['langkah', 'cepat', 'pemerintahan', 'merespon', 'krisis', 'patut', 'apresiasi', 'salut']”.

2.10 Arsitektur IndoBERT

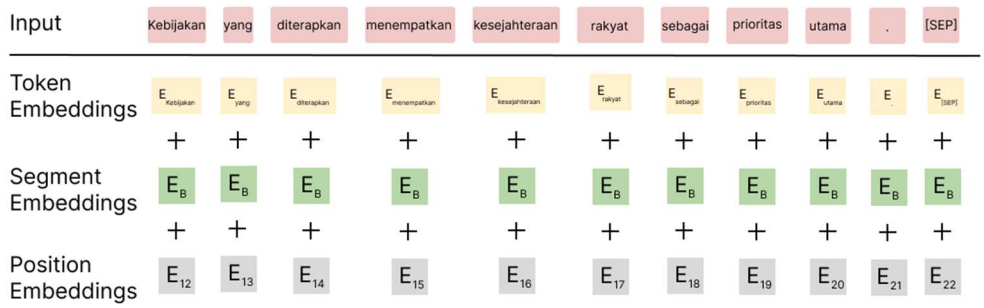
Di bawah ini adalah penjelasan terkait arsitektur indobert yang terdiri dari Representasi *Input* BERT, Arsitektur *Encoder* pada *Transformer*, dan *Pre-training* dan *Fine-Tuning*. Berikut penjelasannya:

1. Representasi *Input* BERT

Kalimat yang digunakan pada proses ini: Pemerintahan Prabowo dan Gibran menghadirkan optimisme baru bagi Indonesia. Kebijakan yang diterapkan menempatkan kesejahteraan rakyat sebagai prioritas utama.



Gambar 9 Representasi *Input* BERT kalimat pertama



Gambar 10 Representasi *Input* BERT kalimat kedua

Gambar 9 dan 10 di atas merupakan representasi *input* BERT. Proses ini mengubah teks yang bisa dibaca manusia menjadi format numerik yang kaya informasi untuk dipahami oleh model. Prosesnya terdiri dari tokenisasi dan 3 tahap utama (*embedding*) yang kemudian digabungkan:

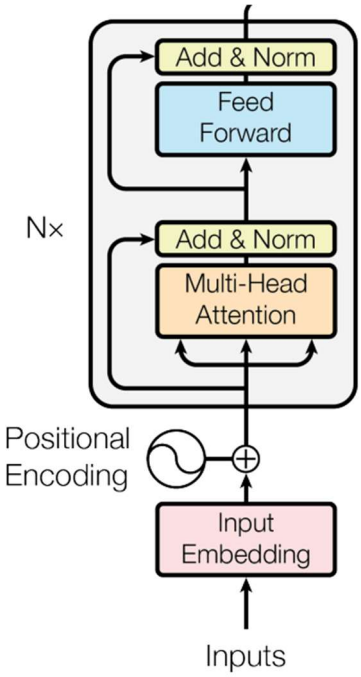
1. Input dan Tokenisasi (Warna merah)
   Pertama, kalimat Anda dipecah menjadi unit-unit yang lebih kecil yang disebut *token*. Biasanya, ini adalah kata atau tanda baca. Kemudian, dua *token special* ditambahkan:
   a. [CLS] (*Classification*): Diletakkan di awal setiap urutan. Representasi dari token ini sering digunakan untuk tugas klasifikasi kalimat secara keseluruhan.
   b. [SEP] (*Separator*): Diletakkan di akhir kalimat untuk menandakan batas akhir sebuah segmen atau kalimat.

   Jadi, *input* dari teks yang digunakan: [CLS] Pemerintahan Prabowo dan Gibran menghadirkan optimisme baru bagi Indonesia. [SEP] Kebijakan yang diterapkan menempatkan kesejahteraan rakyat sebagai prioritas utama. [SEP]
2. *Token Embeddings* (Warna Kuning)
   Setiap token di dalam kalimat dipetakan ke sebuah vektor numerik. Vektor ini merespresentasikan makna dari token tersebut. Misalnya, token Pemerintahan akan memiliki vektor $E_{Pemerintahan}$ yang berisi angka-angka yang menangkap arti dari kata “pemerintahan”.
3. *Segment Embeddings* (Warna Hijau)
   Embedding ini berfungsi untuk membedakan antara kalimat satu dengan kalimat lainnya. Kalimat terdiri dari dua kalimat (dua segmen), maka token di dalamnya diberi tanda, yaitu E_A untuk kalimat satu dan E_b untuk kalimat dua.
4. *Position Embeddings* (Warna Abu-abu)
   Model *Transformer* seperti BERT tidak memiliki pemahaman inheren tentang urutan kata. Oleh karena itu, *Position Embedding* ditambahkan untuk memberikan informasi posisi atau urutan setiap token di dalam kalimat. Token pertama mendapatkan vektor posisi E_0, yang kedua E_1, dan seterusnya hingga token terakhir E_{22}.

2. Arsitektur *Transformer*

Setelah kalimat "Pemerintahan Prabowo dan Gibran menghadirkan optimisme baru bagi Indonesia. Kebijakan yang diterapkan menempatkan kesejahteraan rakyat sebagai prioritas utama." berhasil diubah menjadi serangkaian vektor *input* (hasil penjumlahan Token, *Segment*, dan *Position Embedding*), vektor-vektor ini siap dimasukkan ke "mesin" utama IndoBERT: Arsitektur *Encoder* pada *Transformer*. IndoBERT menggunakan *encoder* saja dari arsitektur *Transformer* yang terdapat pada gambar 11. *Encoder* bertugas mengolah deretan vektor *input* agar model dapat memahami konteks kalimat secara mendalam dan bidirectional. Sama seperti *BERT-base*, IndoBERT memiliki:

a. 12 *Encoder Layers*
b. Setiap layer memiliki:
    1. *Multi-Head Self-Attention*
    2. *Add & Norm*
    3. *Feed Forward Network*
    4. *Add & Norm*



Gambar 11 Arsitektur *Encoder* pada *Transformer* (Vaswani et al., 2023)

Berikut adalah penjelasan dari *layer-layer* yang terdapat pada Arsitektur *Encoder* pada *Transformer*:

1. *Multi-Head Self-Attention*

    Self-attention memungkinkan model memperhatikan setiap kata berdasarkan seluruh konteks kalimat. Contoh ketika memproses token "kesejahteraan" pada kalimat kedua:

Model mempelajari keterkaitannya dengan:

a. “Kebijakan yang diterapkan” → subjek dan frasa penentu
b. “menempatkan” → hubungan sebagai objek tujuan
c. “prioritas utama” → indikator tingkat kepentingan
d. “rakyat” → penjelas objek kesejahteraan

Pada kalimat pertama:

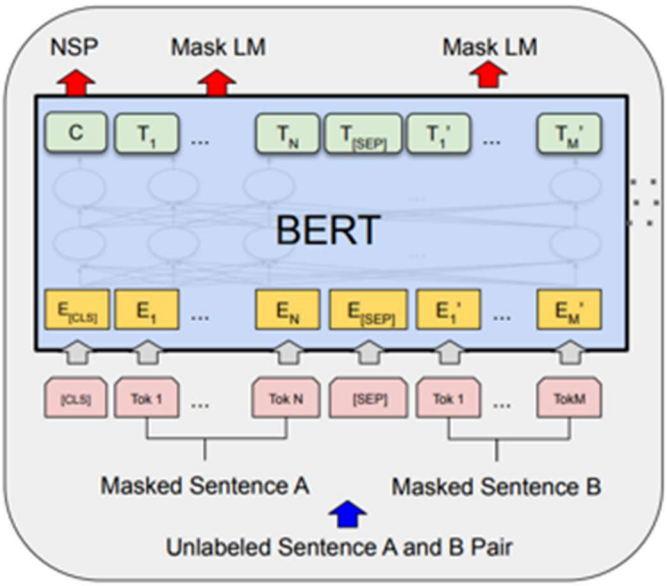
Kata “optimisme” memperhatikan:

a. “Pemerintahan Prabowo dan Gibran” → penyebab
b. “menghadirkan” → tindakan
c. “Indonesia” → objek manfaat

Proses ini terjadi secara paralel untuk semua token dan diproses melalui banyak “*attention heads*” untuk mempelajari berbagai jenis hubungan (sintaksis, semantik, gramatikal).

2. *Add & Norm* (*Residual Connection* + *Layer Normalization*)
   Setelah *self-attention*, *output* ditambahkan kembali ke *input* aslinya (*residual connection*) untuk menjaga stabilitas pembelajaran, lalu dinormalisasi agar berada dalam skala yang seimbang (*layer normalization*).
3. *Feed Forward Network* (FFN)
   FFN mengubah representasi vektor menjadi bentuk yang lebih abstrak dan informatif. Operasi ini dilakukan untuk setiap token secara individual namun tetap mempertahankan konteks global.
4. Proses Berulang 12 Kali
   Layer-layer encoder diproses bertumpuk sebanyak 12 kali. Pada *layer* terakhir, setiap token mengandung pemahaman konteks yang kaya, misalnya:
   a. “rakyat” kini memiliki makna terkait “kesejahteraan”, “prioritas utama”, dan kebijakan pemerintah.
   b. “optimisme” memiliki konteks hubungan dengan “menghadirkan” dan aktor “Prabowo dan Gibran”.

3. *Pre-training* dan *Fine-Tuning*
   Sebelum dapat memahami Bahasa Indonesia dengan baik, IndoBERT harus dilatih melalui dua tahap besar:
   1. *Pre-training*
      Pre-training dilakukan pada jutaan kalimat Bahasa Indonesia dengan dua tugas:

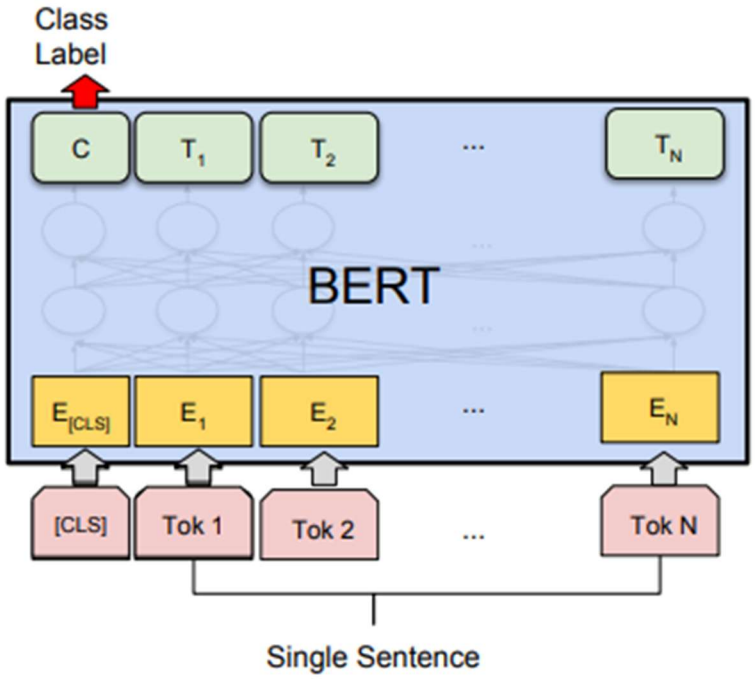


Gambar 12 Arsitektur *Pre-training* BERT (Devlin et al., 2019)

A. *Masked Language Model* (MLM)
   Beberapa token disembunyikan secara acak (sekitar 15%):
   Contoh:
   Pemerintahan Prabowo dan Gibran menghadirkan [MASK] baru bagi Indonesia.
   Model harus memprediksi token asli “optimisme”.
B. *Next Sentence Prediction* (NSP)
   Model dilatih untuk menentukan apakah kalimat B benar-benar lanjutan dari kalimat A.
   Contoh:
   A: “Pemerintahan Prabowo dan Gibran menghadirkan optimisme baru bagi Indonesia.”
   B: “Kebijakan yang diterapkan menempatkan kesejahteraan rakyat sebagai prioritas utama.”
   → Label: IsNext
   Dengan dua tugas ini, model mempelajari:
   - hubungan antarkata
   - hubungan antar-kalimat
   - pola semantik
   - struktur sintaksis bahasa Indonesia

Gambar 12 di atas adalah Arsitektur *Pre-training* BERT.

2. *Fine-tuning*
   Setelah *pre-training*, IndoBERT dapat disesuaikan untuk tugas spesifik. Gambar 13 di bawah ini adalah Arsitektur *fine-tuning* BERT.



Gambar 13 Arsitektur *fine-tuning* BERT (Devlin et al., 2019)

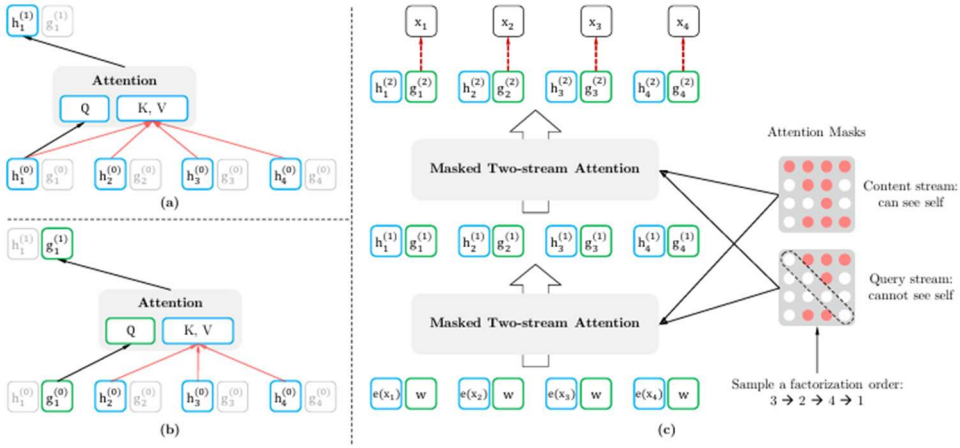
Contoh:

1. Kalimat dimasukkan ke IndoBERT.
2. Representasi akhir dari token [CLS] digunakan sebagai representasi seluruh teks.
3. Vektor [CLS] diteruskan ke classification layer.
4. Model memprediksi kelas, misalnya:
    a. Positif
    b. Negatif
    c. Netral

Dengan ribuan contoh berlabel, IndoBERT dapat menyelesaikan berbagai tugas seperti analisis sentimen, klasifikasi berita, NER, dan lainnya.

2.11 Arsitektur XLNet

Berikut adalah arsitektur XLNet:



Gambar 14 (a): *Content stream attention*, yang sama dengan perhatian diri standar. (b): *Query stream attention*, yang tidak memiliki akses informasi tentang konten x_{zt}. (c): Gambaran umum pelatihan permutation language modeling dengan two-stream attention (Yang et al., 2019)

Kalimat yang digunakan pada proses ini: Pemerintahan Prabowo dan Gibran menghadirkan optimisme baru bagi Indonesia. Kebijakan yang diterapkan menempatkan kesejahteraan rakyat sebagai prioritas utama.

Arsitektur ini menggunakan mekanisme canggih yang disebut *Permutation Language Modeling* (PLM) dan *Two-Stream Self-Attention* untuk memahami konteks kalimat secara dua arah (*bidirectional*).

1. Alur Pemrosesan XLNet dari *Input* sampai *Output*
   Input Teks → Tokenisasi
   Teks dipecah menjadi token, misalnya:
   Teks:
   1) [Pemerintahan]
   2) [Prabowo]
   3) [dan]
   4) [Gibran]
   5) [menghadirkan]
   6) [optimisme]
   7) [baru]
   8) [bagi]
   9) [Indonesia]
   10) [Kebijakan]
   11) [yang]
   12) [diterapkan]
   13) [menempatkan]
   14) [kesejahteraan]

15) [rakyat]
16) [sebagai]
17) [prioritas]
18) [utama]

Setiap token diubah menjadi *embedding* (kata, posisi, segmen).

2. Dua Aliran: *Content Stream* (h) & *Query Stream* (g)
   1) Content Stream – h
      a. Dilabeli h_i pada gambar.
      b. Menyimpan informasi "isi" token.
      c. Boleh melihat semua token lain dalam urutan faktorisasi.

      Contoh:

      h untuk token "optimisme" dapat melihat konteks lengkap seperti:
      a. Pemerintahan
      b. Prabowo dan Gibran
      c. menghadirkan
      d. optimisme
      e. baru

      Ini menghasilkan representasi semantik mendalam.
   2) Query Stream – g
      a. Dilabeli g_i.
      b. Dipakai untuk memprediksi token pada posisi tersebut.
      c. Tidak boleh melihat token itu sendiri (*masked*).

      Contoh:

      g untuk token "kesejahteraan" tidak boleh melihat kata "kesejahteraan". Ia hanya boleh melihat:
      a. Kebijakan
      b. yang
      c. diterapkan
      d. menempatkan

      Aliran ini membuat XLNet tetap autoregressive saat belajar.

3. Masked Two-Stream Attention

Ini bagian inti XLNet:

1) *Content stream* (h):
   a. Boleh melihat semua token yang berada sebelum posisinya dalam faktorisasi
   b. Dapat menggunakan informasi penuh untuk memahami konteks.
2) *Query stream* (g):
   a. Tidak boleh melihat diri sendiri
   b. Hanya melihat h dari token lain
   c. Digunakan untuk memprediksi token tanpa kebocoran informasi

Faktorisasi acak

XLNet menggunakan beberapa urutan faktorisasi — contoh (dari gambar 14):

3 → 2 → 4 → 1

Saat memprediksi “optimisme”, model bisa jadi memakai urutan berbeda setiap kali sehingga konteks yang dipelajari lebih kaya.

4. Proses untuk Teks
   Langkah 1 — Tokenisasi
   Setiap kata dari “Pemerintahan” sampai “utama” menjadi token & embedding.
   Langkah 2 — Pemrosesan oleh *Two-Stream Attention*
   1) *Content stream (h)* mengumpulkan makna penuh dari kalimat.
      Misal, representasi untuk “prioritas” memahami bahwa:
      a. kebijakan tersebut menempatkan
      b. kesejahteraan rakyat
      c. sebagai prioritas utama
   2) *Query stream (g)* mempelajari prediksi setiap token.
      Misal, g pada posisi “Indonesia” belajar memprediksi kata itu tanpa melihat “Indonesia” secara langsung.
   Langkah 3 — Representasi Akhir
   Setelah melalui beberapa lapisan:

   XLNet memahami bahwa konteks teks Anda adalah:

   a. Pemerintahan Prabowo–Gibran → membawa optimisme

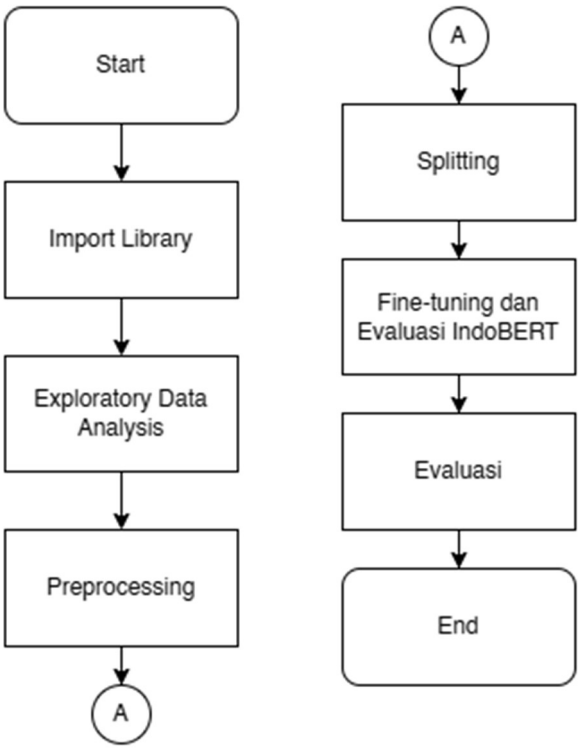
   b. Fokus utama → kesejahteraan rakyat

   c. Arah kebijakan → pro-rakyat / prioritas kesejahteraan

Representasi inilah yang dipakai untuk tugas selanjutnya (prediksi, klasifikasi, dll.)

2.12 Model Analisis Sentimen dengan IndoBERT

Langkah selanjutnya adalah membaginya menjadi data latih dan data uji. Kemudian, klasifikasi akan dilakukan menggunakan metode IndoBERT. Berikut merupakan diagram alur dari algoritma IndoBERT dalam menentukan prediksi kelas klasifikasi sentimen.



Gambar 15 Flowchart IndoBERT

Pada gambar 15 dijelaskan alur proses pengembangan model klasifikasi teks menggunakan IndoBERT. Proses dimulai dari tahap awal yaitu *start*, dilanjutkan dengan *import library* yang merupakan proses pemanggilan berbagai pustaka penting seperti numpy, pandas, dan hugging face transformers. Tahap selanjutnya adalah *exploratory data analysis (EDA)* yang bertujuan untuk memahami karakteristik awal data yang digunakan, seperti distribusi kelas dan pola teks. Setelah EDA, dilakukan tahap *preprocessing* yang mencakup proses *case folding, remove punctuation, normalization, tokenization, stopword removal*, dan *stemming* guna membersihkan dan menyiapkan data agar optimal saat digunakan oleh model.

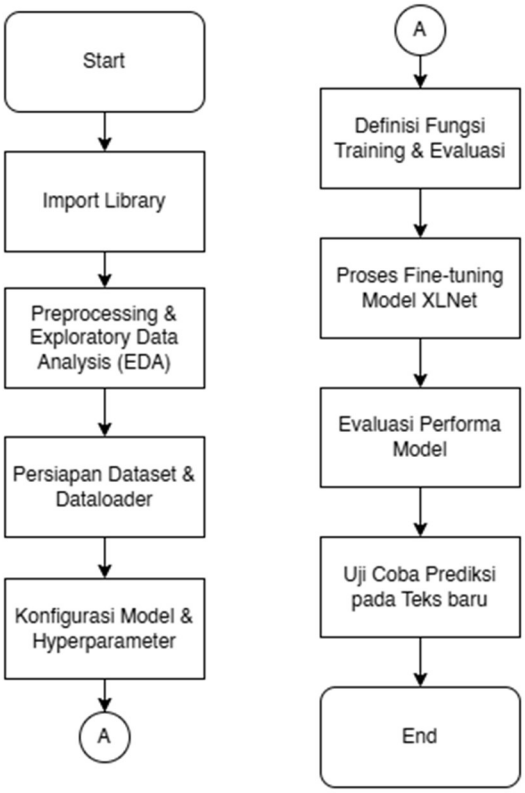
Langkah berikutnya adalahs*Splitting*, yaitu proses pembagian data menjadi data latih dan data uji. Tahap ini penting agar model dapat belajar dari sebagian data dan diuji performanya menggunakan data yang tidak pernah dilihat sebelumnya. Setelah data terbagi, proses dilanjutkan ke *fine-tuning dan* evaluasi IndoBERT di mana model IndoBERT dilatih ulang menggunakan data latih yang sudah diproses. Model juga dievaluasi menggunakan data uji untuk mengetahui seberapa baik hasil prediksi yang dihasilkan terhadap data baru.

Tahapan terakhir adalah *evaluasi*, di mana hasil prediksi model dibandingkan dengan label sebenarnya untuk menghitung metrik performa seperti akurasi, presisi, *recall*, dan F1-score. Proses ini memberikan gambaran akhir mengenai kualitas model sebelum memasuki tahap akhir yaitu *end* sebagai penutup alur kerja. Flowchart ini menggambarkan tahapan end-to-end yang sistematis dalam membangun model

klasifikasi teks berbasis IndoBERT, mulai dari persiapan data hingga evaluasi akhir terhadap performa model.

2.13 Model Analisis Sentimen dengan XLNet

Dataset juga akan diuji dengan menggunakan model XLNet. Berikut merupakan diagram alur dari algoritma XLNet dalam menentukan prediksi kelas klasifikasi sentimen.



Gambar 16 Flowchart XLNet

Pada gambar 16 dijelaskan alur kerja pembangunan model klasifikasi teks menggunakan XLNet dari huggingface. Proses dimulai dari tahap *start* dengan mengimpor pustaka atau *library* penting yang dibutuhkan dalam pemrosesan data dan pelatihan model. Selanjutnya dilakukan tahap *preprocessing & exploratory data analysis (EDA)*, di mana data mentah dibersihkan dan dianalisis secara awal untuk memahami distribusi data dan karakteristiknya. Setelah itu, dilakukan persiapan *dataset & dataloader* yang mencakup pembuatan kelas dataset khusus dan pemuatan data dengan *dataloader* agar proses pelatihan berjalan lebih efisien.

Langkah berikutnya adalah melakukan konfigurasi model *& hyperparameter*, termasuk menentukan parameter penting seperti *learning rate*, jumlah *epoch*, dan *batch size*. Setelah konfigurasi selesai, alur berpindah ke sisi kanan flowchart yang dimulai dengan definisi fungsi *training & evaluasi*, yaitu penulisan fungsi yang digunakan untuk melatih model dan mengevaluasi performanya. Kemudian, dilakukan proses *fine-tuning*

model *XLNet*, yaitu pelatihan lanjutan dari model *pre-trained* menggunakan data spesifik untuk tugas klasifikasi teks yang sedang dikerjakan.

Tahapan berikutnya adalah evaluasi performa model, di mana model yang telah dilatih diukur performanya terhadap data uji untuk mengetahui tingkat akurasi dan efektivitasnya. Setelah evaluasi, dilakukan uji coba prediksi pada teks baru, yang merupakan penerapan model terhadap data yang belum pernah dilihat sebelumnya untuk menguji kemampuan prediktif model secara praktis. Seluruh proses ini ditutup dengan tahap *end*, yang menandai selesainya pembangunan dan penerapan model klasifikasi teks berbasis XLNet secara *end-to-end*.

2.14 Implementasi Sistem

Dalam penelitian ini, sistem yang dikembangkan merupakan sistem pendukung Keputusan yang memungkinkan pengguna untuk menginput teks secara langsung atau menggunggah file .csv. Sistem ini akan mengklasifikasikan setiap data teks ke dalam tiga kategori sentimen, yaitu netral, positif, dan negatif. Implementasi sistem ini meliputi beberapa tahapan utama, yaitu:

1. *Website*: Sebagai antarmuka visual yang menampilkan hasil klasifikasi sentimen. Alat yang digunakan untuk pembuatan website adalah Visual Studio Code sebagai editor teks, HTML untuk front-end, dan Flask untuk back-end.
2. Metode: Metode yang diimplementasikan pada sistem ini adalah IndoBERT. Pemilihan IndoBERT didasarkan pada hasil evaluasi performa model, di mana IndoBERT menunjukkan akurasi yang lebih tinggi dibandingkan XLNet, sehingga dianggap paling optimal untuk tugas klasifikasi dalam penelitian ini.

2.15 Mengevaluasi Sistem

Pada penelitian ini, pengujian akan dilakukan dengan menggunakan metode *black box* untuk menguji dan memastikan bahwa program yang telah dibuat berfungsi dengan baik.