BAB III

METODOLOGI PENELITIAN

3.1. Objek Penelitian

Objek penelitian ini adalah salah satu model berbasis *Machine Learning*, khususnya *Deep Learning* dalam topik NLP, yaitu BERT (*Bidirectional Encoder Representations from Transformers*) untuk mengklasifikasi secara biner teks di media sosial apakah mengandung konten kekerasan atau tidak. Kekerasan yang dimaksud mayoritas meliputi perundungan siber (*cyberbullying*) dan ujaran kebencian (*hate speech*), tetapi tidak menutup kemungkinan untuk kategori lainnya, seperti pelecehan seksual (*sexting*), pemfitnahan, penyindiran, dan lain-lain. Model ini kemudian digunakan untuk memprediksi data baru, yaitu data teks yang diekstrak dari objek gambar tangkapan layar (*screenshot*) yang mengandung konten kekerasan. Sumber data teks tidak dibatasi oleh *platform* media sosial tertentu saja. Hal ini ditujukan untuk menguji generalisasi dari model yang dibangun, agar model tidak hanya mampu menganalisis teks di kanal media sosial tunggal.

3.2. Metode Penelitian

3.2.1. Prapemrosesan Data

Untuk data Twitter, teks dari setiap *tweet* akan melalui tahap *case folding* terlebih dahulu, yaitu mengubah semua kata menjadi huruf kecil. Setelah itu tahap *data cleansing* dilakukan, meliputi penghapusan URL atau *link*, *hashtag*, nama pengguna, istilah “RT” (*retweet*), tanda baca, dan transformasi beberapa sintaks HTML, seperti misalnya “&amp” yang diubah menjadi “dan”. Setelah itu, setiap

kata akan dinormalisasi ke bentuk yang lebih baku menggunakan kamus kosakata baku dari dataset Ibrohim & Budi (2019) dengan tambahan kamus kosakata kata-kata gaul (*slang words*) eksternal lainnya[2].

Dataset Instagram memiliki tahap-tahap prapemrosesan yang hampir sama dengan data Twitter, seperti *case folding*, penghapusan penyebutan nama pengguna, *hashtag*, dan normalisasi teks. Untuk data Instagram ada beberapa kasus lain yang perlu dilakukan *data cleansing* lebih lanjut, seperti menghilangkan karakter *encoding*.

Setiap dataset terdiri dari data teks dan label dari masing-masing teks. Karena nama pelabelan dari masing-masing dataset beragam, maka akan disamaratakan menjadi "0" untuk teks yang tidak mengandung kekerasan dan "1" untuk teks yang mengandung konten kekerasan. Selain karena faktor keberagaman nama pelabelan, berdasarkan hasil pengamatan dataset, ada beberapa teks yang sebenarnya tidak hanya mengandung *cyberbullying* atau *hate speech*. Ada beberapa sampel teks yang lebih mengarah ke jenis kekerasan di media sosial lainnya, contohnya salah satunya adalah pelecehan seksual.

3.2.2. Pemformatan Data

Sebelum data teks siap diklasifikasi, data teks tersebut perlu melalui tahap pemrosesan lebih lanjut agar formatnya sesuai dengan arsitektur dari model BERT. Tahap-tahap tersebut di antaranya sebagai berikut:

[2] https://github.com/louisowen6/NLP_bahasa_resources

1) Penambahan token khusus, meliputi: (1) token `[SEP]` di setiap akhir kalimat sebagai penanda pemisah antar kalimat, dan (2) token `[CLS]` di setiap awal teks yang ada, sebagai penanda bahwa *hidden layer* dari arsitektur *Transformers* milik BERT akan melakukan pembelajaran klasifikasi teks, dimana token ini akan terus mengalami pembaharuan bobot dari setiap lapisan *Transformers*, sehingga hasil pembobotan akhir dari token ini yang akan menjadi *output* dari model (Devlin et al., 2019).
2) *Padding & Truncating*, yaitu tahap untuk menyamaratakan panjang token dari setiap teks dalam ukuran yang tetap. Ukuran standar dari panjang token milik model *pre-trained* BERT adalah 512 token, tetapi kita dapat mendefinisikan kembali berapa panjang token maksimal sesuai dengan data teks yang kita miliki. Misalnya, kalimat pada dataset pelatihan kita tidak terlalu panjang sehingga kita bisa mengurangi panjang maksimal dari token. Hal ini dapat mempengaruhi waktu proses pelatihan dan evaluasi.
3) *Attention Masking*, merupakan urutan *array* yang berisikan nilai 0 atau 1, 0 berarti token tersebut merupakan token dari hasil *padding & truncating*, sedangkan 1 berarti token tersebut merupakan token dari teks asli. Hal ini bertujuan agar *self-attention layer* dari BERT tidak memperdulikan token *padding* sehingga tidak mempengaruhi hasil dari model.

Setelah teks melalui tahap pemrosesan di atas, maka teks dapat langsung ditokenisasi dengan model *pre-trained* yang sudah ditentukan. Output dari proses tokenisasi ini adalah himpunan (*array*) tensor yang meliputi hasil *embedding* dari setiap kata berdasarkan pengenalan kosakata dari model *pre-trained* yang dimiliki.

Token yang berlebih akan diwakilkan dengan angka 0. Misalnya, dalam teks kita memiliki 30 kata, dan kita tentukan panjang token maksimal 32, maka dua token terakhir hasil *embedding* akan bernilai 0.

3.2.3. Pembagian Dataset

Selanjutnya, dataset akan dibagi menjadi data *training* dan *testing* dengan proporsi masing-masing 80% dan 20% dari keseluruhan jumlah data. Pembagian ini mengikuti penelitian terakhir mengenai deteksi kekerasan pada teks media sosial berbahasa Indonesia yang juga menerapkan model BERT (Salminen et al., 2020). Setelah itu data *training* akan dibagi lagi menjadi data *training* itu sendiri dan data validasi dengan proporsi masing-masing 70% dan 30% dari jumlah data *training*.

3.2.4. Pemodelan

Di tahap ini, akan dilakukan *fine-tuning* dari empat model *pre-trained* berbasis BERT yang berbeda, meliputi:

1) *Multilingual* BERT

Peneliti asli BERT mengembangkan arsitektur model BERT yang awalnya hanya melatih model dengan kumpulan korpus dalam bahasa Inggris, kemudian membuat model multibahasa[3] dengan melatih arsitektur yang sama menggunakan dataset korpus dalam 104 bahasa berbeda. 104 bahasa yang dipilih berdasarkan pada jumlah arsip Wikipedia terbesar, dan bahasa Indonesia merupakan salah satunya. Arsitektur yang digunakan pada penelitian ini adalah arsitektur versi

[3] https://github.com/google-research/bert/blob/master/multilingual.md

standar dengan 12 *attention layers*, 768 *hidden layers*, 12 *transformer heads*, dan menghasilkan ~178 juta parameter (Devlin et al., 2019).

2) *Multilingual* DistilBERT

Model ini dilatih menggunakan dataset yang sama seperti *Multilingual* BERT, tetapi dengan arsitektur BERT yang berbeda. Teknik distilasi (Hinton et al., 2015) digunakan untuk mengurangi kompleksitas dari model BERT yang asli sehingga menghasilkan model yang lebih ringkas, hemat sumber daya komputasi, tetapi tidak mengorbankan performa model terlalu besar.

Arsitektur versi standar dari *Multilingual* DistilBERT[4] ini memangkas sekitar 40% dari ukuran arsitektur asli BERT, di mana lapisan *Transformers* yang digunakan hanya separuh dari versi standar BERT, yaitu 6 lapisan. Parameter yang dihasilkan berkisar 135 juta parameter. Modelnya 60% lebih cepat dan mempertahankan 97% dari performa arsitektur asli BERT. (Sanh et al., 2019)

3) IndoBERT *Base*

IndoBERT (Wilie et al., 2020)[5] merupakan pengembangan dari arsitektur BERT yang khusus dilatih menggunakan data korpus bahasa Indonesia. Dataset bahasa Indonesia yang digunakan untuk pelatihan mencapai 4 milyar kata, baik dalam bahasa formal maupun bahasa sehari-hari hasil penggabungan 12 dataset korpus bahasa Indonesia. Dataset ini kemudian dilatih dengan arsitektur versi

[4] https://github.com/huggingface/transformers/tree/master/examples/distillation
[5] https://github.com/indobenchmark/indonlu

standar dari BERT, yaitu dengan 12 *Transformer layers*, 768 *hidden layers*, dan 12 *attention heads* dan menghasilkan 124,5 juta parameter.

4) IndoBERT *Lite*

Arsitektur model IndoBERT versi *lite* memanfaatkan arsitektur model *pre-trained* yang berbeda dari model BERT asli, yaitu **ALBERT (A Lite BERT)**[6]. ALBERT masih memiliki arsitektur model yang sama dengan BERT, tetapi menyumbangkan desain pelatihan baru yang menghasilkan ukuran model yang lebih kecil tetapi dengan performa yang lebih baik jika dibandingkan model BERT aslinya.

Versi standar dari ALBERT memiliki jumlah *transformer layers*, *hidden layers*, dan *attention heads* yang sama dengan versi standar dari BERT, yaitu masing-masing 12, 768, dan 12. Perbedaan keduanya adalah ukuran vektor dari *word embedding*, dimana model BERT asli berukuran 768 sedangkan ALBERT hanya berukuran 128 berkat teknik *factorized embedding parameterization* yang digunakan. Selain itu, ALBERT juga melakukan *Cross-layer Parameter Sharing* yang memangkas jumlah parameter menjadi hanya 12 juta parameter dari 110 juta parameter. Arsitektur ini yang kemudian digunakan oleh IndoBERT *lite* untuk melatih dataset korpus bahasa Indonesia yang sudah dimiliki. (Lan et al., 2019)

Proses *fine-tuning* dilakukan dengan memanfaatkan *library* "transformers" dari HuggingFace[7], dan proses pemrograman akan dilakukan di atas *framework* PyTorch menggunakan bahasa pemrograman Python. *Library* "transformers" juga

[6] https://github.com/google-research/albert
[7] https://github.com/huggingface/transformers

sudah menyediakan keempat model *pre-trained* yang akan digunakan penelitian ini pada repositorinya.

3.2.5. *Hyperparameter Tuning*

Menurut penelitian dari BERT, ada tiga parameter yang dapat disesuaikan untuk mengoptimalkan performa pada saat *fine-tuning*, di antaranya: *batch size*, *learning rate*, dan jumlah *epochs* (Devlin et al., 2019). Maka, berdasarkan rekomendasi tersebut akan dilakukan pencarian nilai terbaik dari ketiga parameter tersebut.

Dikarenakan kombinasi yang cukup banyak dan membutuhkan proses komputasi yang lama, proses ini dibantu dengan memanfaatkan modul *Trainer*[8] yang juga tersedia di *library* ”transformers” dari HuggingFace (HuggingFace, 2020). Pada modul ini terdapat fungsi `hyperparameter_search()` yang mengautomatisasi pelatihan model berdasarkan setiap kombinasi parameter yang diinginkan. Fungsi ini akan memberikan output berupa nilai parameter terbaik.

Berdasarkan hasil pencarian *hyperparameter* ini, nilai-nilai parameter tersebut yang akan diambil dan digunakan untuk tahap pelatihan model. Adapun daftar alternatif nilai parameter yang digunakan (Devlin et al., 2019) adalah sebagai berikut:

- ***Batch size***: 16, 32
- ***Learning rate***: 5e-5, 3e-5, 23-5
- ***Epochs***: 2, 3, 4

[8] Trainer — transformers 3.5.0 documentation (huggingface.co)

3.2.6. Pelatihan Model

Adam dipilih sebagai *optimizer* yang digunakan, sesuai dengan pengaturan *default* dari model BERT. Parameter yang disetel sesuai dengan hasil dari pencarian *hyperparameter* paling optimal di tahap sebelumnya. Selama proses pelatihan, akan dilakukan pemantauan terhadap *loss* dan akurasi dari data *training* dan data validasi pada setiap *epoch*. Proses pelatihan akan dijalankan menggunakan Google Colab, IDE berbasis *Jupyter Notebook* untuk pemrograman Python yang menggunakan sumber daya komputasi yang disediakan oleh Google Cloud dengan spesifikasi: RAM sebesar 26 GB, GPU NVIDIA Tesla P100 dengan VRAM sebesar 16 GB.

3.2.7. Evaluasi Model

Evaluasi model klasifikasi teks akan dilakukan pada dataset pengujian (*testing*) dengan metrik akurasi, *precision*, *recall*, dan skor F1. Keempat perhitungan tersebut dapat dilihat pada Rumus 3.1 s.d. Rumus 3.4.

Berdasarkan hasil evaluasi tersebut, maka model *pre-trained* terbaik akan dipilih untuk dievaluasi lebih lanjut. Sejauh ini belum ada penelitian-penelitian terdahulu pada topik klasifikasi teks untuk deteksi kekerasan di media sosial dalam bahasa Indonesia yang melakukan evaluasi lebih lanjut terhadap model, khususnya dalam hal pengawasan model selama masa pembelajarannya. Hal ini krusial karena ke depannya model akan dimanfaatkan untuk menangani data baru. Pengawasan model selama pembelajarannya memungkinkan kita untuk mengevaluasi seberapa baik model dalam belajar dari pengalaman yang dimilikinya, dalam hal ini adalah dataset pelatihan.

$$Akurasi = \frac{Label\ Benar}{Total\ Label}$$

Rumus 3.1. Rumus Perhitungan Akurasi

$$Precision = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

Rumus 3.2. Rumus Perhitungan *Precision*

$$Recall = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

Rumus 3.3. Rumus Perhitungan *Recall*

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

Rumus 3.4. Rumus Perhitungan F1

Metode yang paling sering digunakan untuk hal di atas adalah dengan menggunakan Kurva Pembelajaran atau *Learning Curve*. *Learning Curve* (LC) merupakan kurva yang dianggap sebagai alat yang efektif untuk memantau performa model yang akan menangani tugas baru ke depannya, di mana LC menyajikan representasi matematis dari proses pembelajaran yang terjadi selama iterasi tugas sebelumnya (Anzanello & Fogliatto, 2011). Komponen kurva ini terdiri dari dua, yaitu kurva pembelajaran pelatihan dan kurva pembelajaran validasi. Kurva pembelajaran pelatihan diukur dari dataset pelatihan (*training set*) untuk memberikan gambaran seberapa baik model dalam belajar. Kurva pembelajaran validasi diukur dari dataset validasi (*validation set*) yang terpisah dari dataset pelatihan. Di sini layaknya ujian pada manusia, model akan diuji seberapa

baik dalam menggunakan pengalaman yang sebelumnya dipelajari untuk memecahkan kasus baru, yang berarti hal ini akan menguji seberapa baik generalisasi dari model.

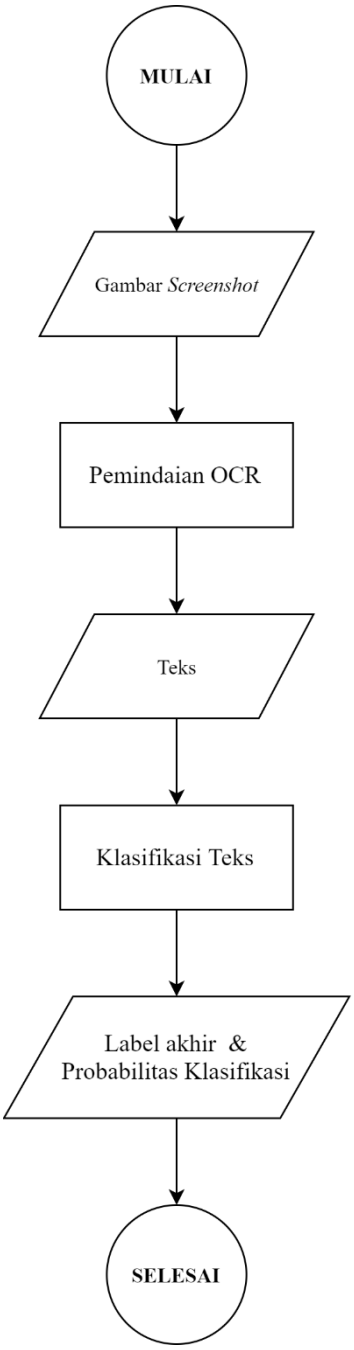
Kurva yang paling umum digunakan adalah kurva *loss* dan kurva akurasi. Kurva *loss* digunakan sebagai metrik untuk mengukur seberapa optimal model dalam menggunakan parameter yang dimiliki untuk belajar, sedangkan kurva akurasi digunakan sebagai metrik untuk mengevaluasi performa yang dihasilkan model. Berdasarkan evaluasi tersebut, maka model akan dipersiapkan dengan lebih maksimal sebelum masuk ke dalam tahap implementasi atau *deployment*.

3.2.8. Implementasi Model

Model yang sudah dipilih dan disetel sedemikian rupa untuk memberikan hasil yang maksimal selanjutnya masuk ke dalam tahap implementasi atau *deployment*. Model akan disematkan pada sistem web sederhana, dimana *user* akan mengunggah sebuah gambar tangkapan layar (*screenshot*) berisikan teks yang dianggap mencela atau mengusik pengguna. Kemudian, API dari Tesseract OCR akan mengekstrak teks dari gambar tersebut sehingga akan didapatkan teks sebagai input dari model klasifikasi yang sebelumnya sudah dibangun. Hasil output dari model adalah penentuan apakah teks tersebut mengandung konten kekerasan atau tidak beserta probabilitas klasifikasinya.

Adapun implementasi model ke dalam purwarupa sistem web menggunakan *framework* Flask untuk pengembangan *back-end* web menggunakan bahasa pemrograman Python. Sedangkan untuk pengembangan dari sisi *front-end*,

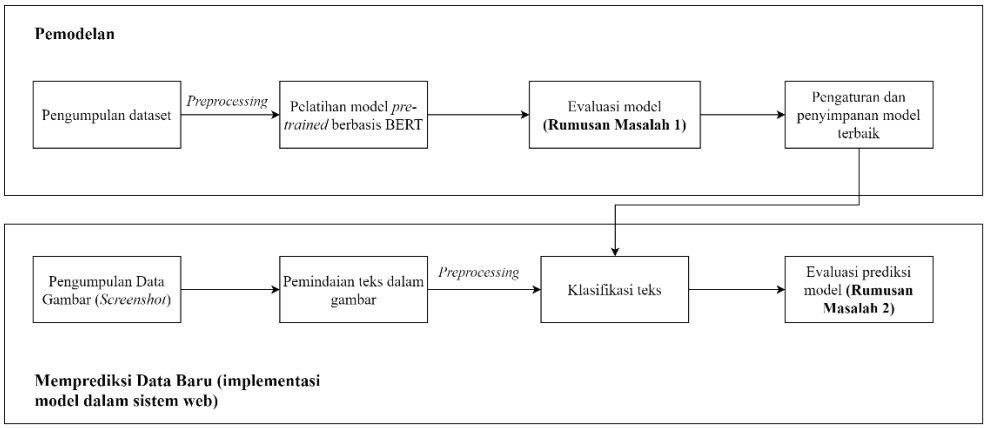
menggunakan bahasa yang sudah umum digunakan, yaitu HTML, CSS dan Javascript dibantu dengan *framework* dari Bootstrap 4. Alur implementasi model diilustrasikan pada Gambar 3.1.



Gambar 3.1. *Flowchart* dari Sistem Implementasi Model

3.3. Kerangka Teori

Kerangka kerja pada penelitian ini disajikan pada Gambar 3.2 di bawah ini.



Gambar 3.2. Kerangka Teori

3.3.1. Rumusan Masalah 1

Pertama-tama, dataset pelatihan yang sudah dikumpulkan akan masuk dalam tahap prapemrosesan yang kemudian menjadi input untuk proses *fine-tuning* dari keempat jenis model *pre-trained* berbasis BERT yang selanjutnya akan dibandingkan performanya. Data akan dilatih untuk tugas spesifik (*downstream task*) dari NLP, yaitu klasifikasi teks. Model yang dihasilkan kemudian akan dievaluasi menggunakan data validasi dan pengujian dari dataset. Model dengan performa terbaik kemudian akan dievaluasi dan dilatih lebih lanjut untuk mendapatkan performa yang maksimal. Kemudian model baru disimpan untuk digunakan pada proses *deployment* atau produksi.

3.3.2. Rumusan Masalah 2

Penelitian ini mengimplementasikan model ke dalam sistem berbasis web. Hal ini bertujuan untuk menguji kemampuan model dalam memprediksi data baru yang diperoleh berdasarkan masukan dari pengguna sistem, sehingga ke depannya sistem ini dapat dikembangkan untuk digunakan oleh khalayak umum.

Data baru yang dikumpulkan terdiri dari 50 sampel yang dikumpulkan secara *purposive sampling*. Sampel diambil secara mandiri, meliputi 25 sampel gambar yang didalamnya terkandung teks bersifat kekerasan (label positif) dan 25 sampel gambar bersifat non-kekerasan (label negatif).