3. ANALISA DAN DESAIN SISTEM

Bab ini akan membahas lebih lanjut tentang analisis dan desain sistem yang digunakan untuk membuat model klasifikasi *multi-label* yang dilatih dengan data *train* yang dihasilkan dari model generasi data *multi-label*. Bagian pertama bab ini akan menjelaskan teknik pengumpulan data serta analisis data pada *dataset* yang digunakan dalam proses penelitian. Bagian selanjutnya bab ini akan menjelaskan alur dan struktur sistem yang akan dijelaskan dalam bentuk *flowchart*. Bagian akhir bab ini, terdapat bagian desain *interface* yang menjelaskan dan menampilkan desain *interface* pada program.

3.1. Analisa Masalah

Seperti yang dapat dilihat pada sub bab 1.1 mengenai latar belakang, tantangan-tantangan yang muncul dalam klasifikasi gangguan mental dalam teks media sosial menggunakan metode *sampling single-to-multi-label* dan varian dari BERT. Pertama, kompleksitas klasifikasi *multi-label* menghadirkan tantangan dalam memahami dan mengelola informasi yang terkandung dalam teks yang dapat merujuk pada beberapa gangguan mental sekaligus.

Keterbatasan data *multi-label* yang relevan juga menjadi masalah signifikan karena kekurangan data yang mencakup setiap label yang mungkin dapat menghambat pembangunan model yang akurat dan umum. Selanjutnya, variasi dalam bahasa dan gaya ekspresi di media sosial menyulitkan pemrosesan teks secara konsisten dan efektif. Sementara itu, perluasan model BERT untuk menangani klasifikasi multi-label memerlukan adaptasi arsitektur yang sesuai dengan sifat multi-label dari masalah ini. Terakhir, evaluasi dan interpretasi model juga merupakan aspek penting yang harus diperhatikan untuk memastikan kesesuaian dari hasil klasifikasi gangguan mental dalam teks media sosial.

3.2. Analisa Data

Data yang digunakan dalam skripsi ini berasal dari sumber penelitian sebelumnya yang telah dilakukan *cleaning* dan *preprocessing* berbentuk *csv*. Data juga ditambah jumlahnya dengan metode *web scraping*, yang juga merupakan metode yang digunakan oleh penelitian sebelumnya untuk memperoleh *dataset* terkait. Data yang digunakan dalam bahasa inggris dan

berupa *text*. Untuk data berupa set *post* pengguna pada platform *reddit* yang menunjukan kondisi mental dari *user.*

Untuk mengambil data tambahan yang hanya menunjukan gangguan mental, dilakukan penyaringan dengan mencari *sub-reddit r/adhd*, *r/anxiety*, *r/bipolarreddit*, *r/bipolar*, *r/depression*, dan *r/ptsd*. Jumlah data ialah 22610 dengan distribusi data untuk setiap label *adhd*, *anxiety*, *bipolar*, *depression*, dan *ptsd* adalah 15871, 18738, 4522, 19043, 6214 secara berturut. Berikut beberapa contoh unggahan *text* dari pengguna *reddit* dapat dilihat pada Tabel 3.1.

Tabel 3.1

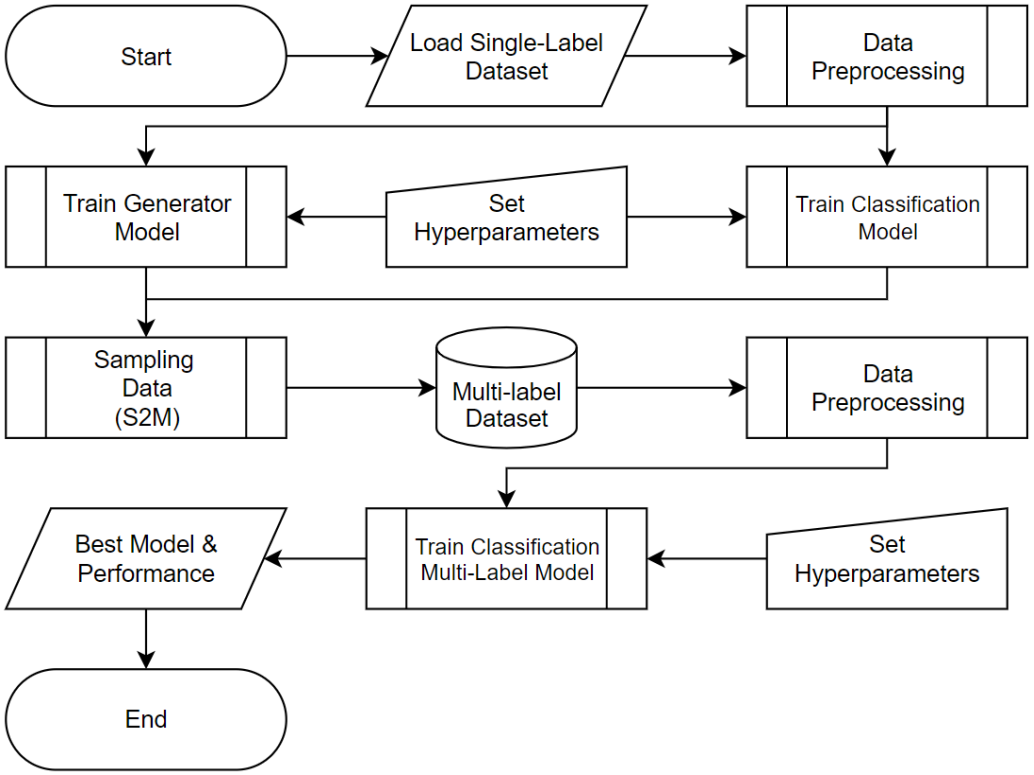
Beberapa Contoh Unggahan *Text Reddit*

label	text
adhd	looking like i'm going to fail out of university due to my poor time management so i'm super stressed
adhd	help! first 6 hours of concerta worse than without meds - hour 7 to 12 quite good, like what?! too low
anxiety	bad anxiety attack today let me start off this post by saying that i suffer from anxiety/depression.
anxiety	seeing a doctor about anxiety? so i have recently been having more acute issues with anxiety. i am
bipolar	how to tell a new significant other that you're bipolar ? i recently got into a new relationship, and i fe
bipolar	update, additional advice requested: bipolar gf attempted suicide last nigh, need advice an update t
depression	i have everything a person could ask for.... why am i so sad all the time? not sure what i'm looking fo
depression	i feel more and more depressed everyday. i hate summer. especially this one. my parents are forcin
ptsd	cyber molestation. i really don't feel like writing out my whole experience. it so confusing and went o
ptsd	tw why am i still thinking about it? tw so, my dad has (until recently) always been the type of parent

3.3. Desain Sistem

Pada desain sistem ini akan dijelaskan alur kerja dari sistem yang akan dibuat dimulai dari *input* data, proses, dan *output* yang diharapkan. Sistem mencakup *preprocessing* data teks, model untuk generasi dan klasifikasi data, dan sistem untuk *testing*.

Gambar 3.1 menunjukkan alur kerja pembuatan model untuk klasifikasi gangguan mental. Proses dimulai dari *load dataset* yang digunakan untuk melatih model generatif dan model *binary classification*. Pertama, data akan dilakukan *preprocessing* dan *tokenization* sebelum dipakai untuk melatih model. Setelah model di build dengan menyesuaikan *hyperparameter*-nya, kedua jenis model tersebut akan dilatih menggunakan data *single-label* set. Kemudian hasil kalimat yang dibuat model generatif akan dilakukan sampling dengan S2M untuk memastikan hasilnya hanya data *multi-label.* Proses menentukan data yang dipilih sesuai kriteria dibantu oleh model klasifikasi biner yang sudah dilatih. Setelah itu data akan disimpan untuk dijadikan sebagai data *train* dan *test* dari model klasifikasi *multi-label* gangguan mental.



Gambar 3.1 *Flowchart* Model

Data yang sudah dihasilkan model generatif akan dibagi menjadi data *training set* dan *testing set.* Data tersebut kemudian dilakukan *tokenization* untuk nanti digunakan melatih model. Setelah model di build dengan menyesuaikan parameter dan *hyperparameter*-nya, model tersebut di *train* menggunakan data *multi-label* yang sudah dibagi sebelumnya. Proses terakhir, untuk mengevaluasi performa model akan digunakan beberapa *metric score* seperti akurasi, *recall*, presisi, dan *f1-score*.

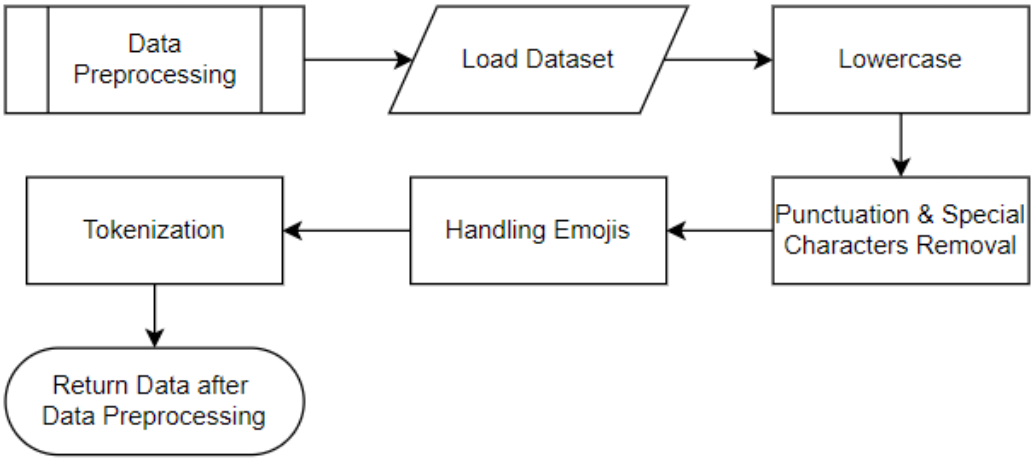
3.3.1. Data Preprocessing

Pemrosesan awal data adalah langkah penting dalam penelitian data yang bertujuan untuk menghilangkan permasalahan yang dapat mengganggu jalannya pemrosesan data. Selama fase ini, noise pada data dikurangi, penyelesaian data yang formatnya kurang konsisten, dan mengisi nilai yang hilang. Agar data siap digunakan dalam pemodelan klasifikasi data, dilakukan *preprocessing*. Selain itu, persiapan dan pembersihan data merupakan fase penting dalam penelitian data yang bertujuan untuk menghasilkan kumpulan data berkualitas tinggi sebelum melakukan analisis tambahan. Sejumlah prosedur terlibat dalam proses ini untuk membersihkan, mengatur, dan mengubah data mentah menjadi format yang lebih sesuai untuk

pemodelan. Hasil yang diharapkan adalah model yang tepat, dapat dipercaya, dan relevan dengan tujuan penelitian yang ingin dicapai. Alur data *preprocessing* dapat dilihat pada *flowchart* di Gambar 3.2.

Gambar 3.2 menjelaskan alur dari data *preprocessing*. Berikut adalah penjelasan lebih detail untuk setiap langkah pada alur tersebut.

- *Lowercase*. Pada tahap ini, semua huruf pada kolom 'text' akan diubah menjadi huruf kecil. Hal ini dilakukan agar model fokus pada makna dan tidak sensitif terhadap variasi penggunaan huruf besar.
- *Punctuation and special characters removal*. Pada tahap ini, semua tanda baca seperti titik, koma, tanda tanya, dan sejenisnya dihapus dari teks. Ini dilakukan untuk menyederhanakan data saat pelatihan model dengan berfokus pada makna kata inti.
- *Handling emojis*. Pada tahap ini, setiap karakter yang termasuk *emoji* diubah dalam representasi kata. Ini dilakukan untuk memahami keseluruhan makna kalimat, karena emoji yang disisipkan bisa saja menyampaikan sebuah pesan.
- *Tokenization*. Semua teks yang ada diubah menjadi token sesuai dengan *dictionary* yang ada. *Dictionary* didapat dari menggunakan model *pre-trained*, salah satunya BERT. Tahap ini juga membatasi panjang maksimal teks yang akan dipakai.



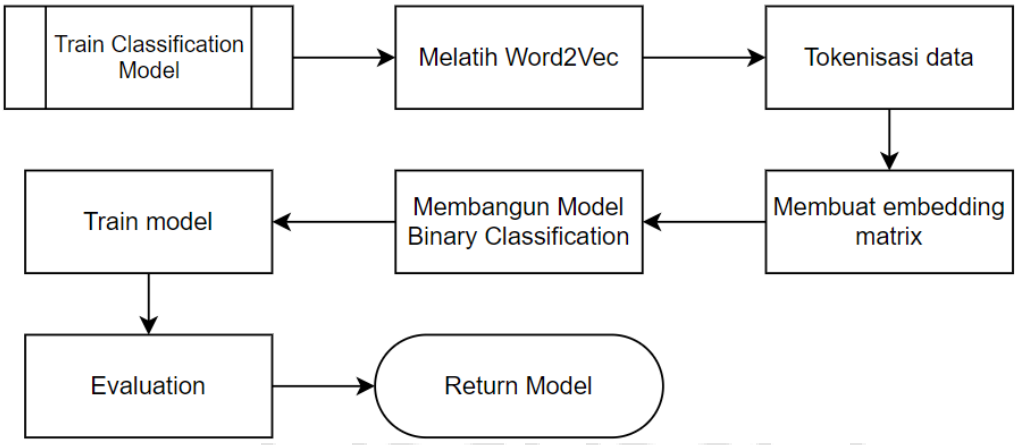
Gambar 3.2 Flowchart Data *Preprocessing*

3.3.2. Build dan Train Model Klasifikasi Biner

Model klasifikasi biner perlu untuk dilatih agar dapat menentukan sebuah data termasuk pada kelas tertentu dengan benar. Proses dimulai dengan menggunakan data yang telah melalui tahap *preprocessing*, memastikan data siap digunakan untuk pelatihan model.

Setiap data kemudian digunakan untuk membangun model klasifikasi biner yang bertugas mendeteksi keberadaan kelas tertentu. Untuk setiap kelas, model klasifikasi biner dibuat dan dilatih secara independen, menghasilkan sejumlah model yang sesuai dengan jumlah kelas yang ada.

Tahapan ini mencakup pemisahan data menjadi set pelatihan dan pengujian, pelatihan model menggunakan algoritma yang dipilih, dan evaluasi performa model menggunakan metrik akurasi. Gambar 3.3 memberikan visualisasi yang jelas tentang langkah-langkah yang diambil dari awal hingga akhir proses pelatihan model klasifikasi biner, memudahkan pemahaman mengenai keseluruhan alur kerja yang diterapkan dalam penelitian ini.



Gambar 3.3 Alur Model Klasifikasi Biner

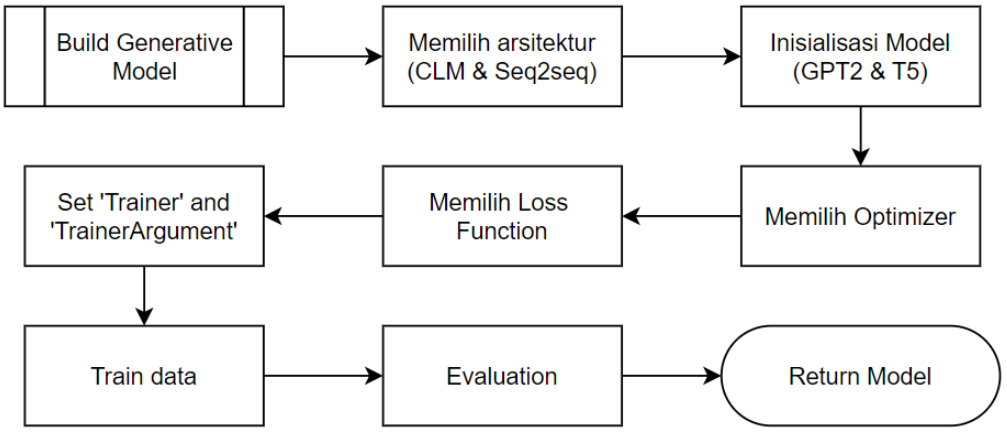
Gambar 3.3 menjelaskan alur dari model klasifikasi biner. Berikut adalah penjelasan lebih detail untuk setiap langkah pada alur tersebut.

- Melatih *Word2Vec*. Tahap ini dilakukan karena *Word2Vec* mengubah kata-kata menjadi vektor bilangan, memungkinkan program untuk memahami hubungan makna antar kata. *Word2Vec* juga memungkinkan analisis data teks yang lebih mendalam, seperti menemukan topik tersembunyi, mengidentifikasi pola dalam penggunaan kata, dan membangun model bahasa yang lebih canggih.
- Tokenisasi data. Tahap ini memecah teks menjadi unit-unit yang lebih kecil seperti kata, kalimat, atau token lainnya, yang memudahkan pemrosesan dan analisis data teks. Dengan tokenisasi, program dapat memanipulasi dan menganalisis teks dengan lebih mudah, seperti menghitung frekuensi kata, mengidentifikasi pola, dan membangun model bahasa.

- Membuat *Embedding Matrix*. Tahap ini dilakukan karena *embedding matrix* mengubah kata-kata menjadi vektor bilangan yang dapat diproses oleh model deep learning dengan lebih mudah dan efisien. *Embedding matrix* memungkinkan model untuk memahami hubungan makna antar kata, sehingga meningkatkan akurasi dan performa model dalam tugas-tugas NLP.
- Membangun model klasifikasi biner. Tahap ini menyusun arsitektur model dengan layer-layer yang sesuai, seperti embedding layer, hidden layer, dan output layer. Proses ini dimulai dengan memberikan data latih ke model dan mengoptimalkan parameternya untuk menghasilkan prediksi yang akurat pada data baru. Dalam penelitian ini, arsitektur yang digunakan merupakan kombinasi dari CNN, LSTM, dan Bi-LSTM.
- Train Model. Pada tahap ini, model akan belajar dari data yang diberikan untuk memprediksi data baru dengan lebih akurat. Proses ini melibatkan menyesuaikan parameter model berdasarkan data latih, sehingga model dapat menemukan pola dan membangun hubungan antara data. Parameter yang disesuaikan diantaranya:
    - *Epoch*. Mengatur jumlah iterasi pelatihan dari data yang ada
    - Rasio validasi. Mengatur jumlah data *train* dan data *validation* dari data yang diberikan ke model.
    - *Batch size*. Mengatur jumlah sampel data yang diproses model secara bersamaan dalam setiap iterasi saat *training*.
- *Evaluation*. Tahap ini melakukan pengujian pada data yang belum pernah dilihat model sebelumnya untuk melihat bagaimana model tersebut berkinerja dalam situasi nyata. Hal ini dilakukan dengan menghitung metrik akurasi untuk mengukur kemampuan model dalam memprediksi dengan benar.

3.3.3. Build dan Train Model Generatif

Pada model generatif harus melalui tahap *build* dan *training* sebelum model dilakukan untuk proses menghasilkan teks. Dapat dilihat pada Gambar 3.4, model yang di-*build* menggunakan model *pretrained* yang akan konfigurasinya perlu diatur ulang. Begitu juga beberapa *hyperparameter* untuk *training* akan disesuaikan dengan data yang disediakan. Setelah melakukan proses *training*, model akan bisa melakukan generasi data sesuai dengan target label yang ditentukan.



Gambar 3.4 Alur Model Generatif

Gambar 3.4 menjelaskan alur dari model generatif. Berikut adalah penjelasan lebih detail untuk setiap langkah pada alur tersebut.

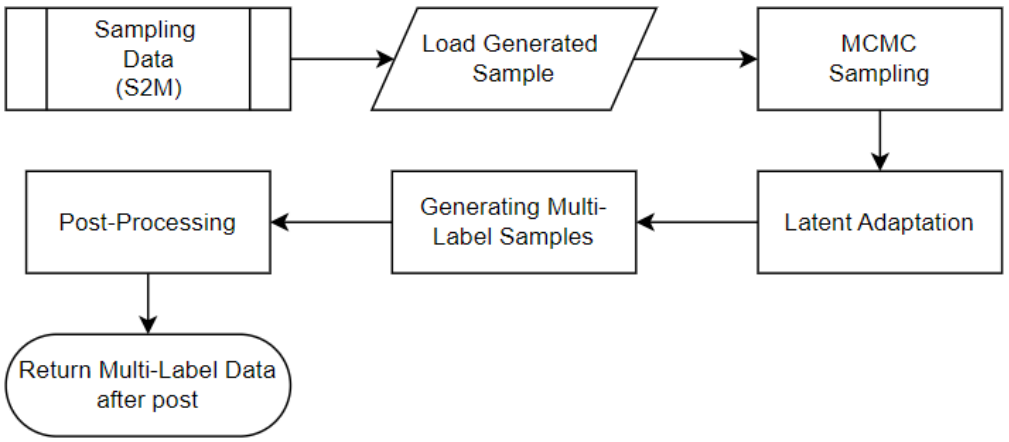
- Memilih arsitektur. Tahap ini menentukan arsitektur yang tepat karena ini dapat menentukan kemampuan model dalam menghasilkan teks yang koheren, kreatif, dan informatif. Dalam penelitian ini, arsitektur *Causal Language Model* dan *Conditional Generation Model*.
- Inisialisasi Model. Pada tahap ini, parameter awal model generatif ditentukan. Parameter diinisialisasi dengan nilai dari model *pretrained* yang memiliki tugas serupa, model GPT2 untuk arsitektur *Causal Language Model* dan model T5 untuk *Conditional Generation Model*. Pendekatan ini berfungsi untuk memanfaatkan pengetahuan yang telah diperoleh model sebelumnya, sehingga model baru dapat belajar lebih cepat dan efisien.
- Memilih *Optimizer*. Bagian ini diperlukan untuk memilih algoritma optimizer yang akan digunakan untuk memperbarui bobot model selama pelatihan. Optimizer menentukan bagaimana model belajar dari data dan bagaimana bobot model diubah untuk meminimalkan kesalahan prediksi. Parameter lain seperti learning rate diatur juga saat memilih optimizer. *Learning rate* menentukan seberapa besar bobot model diperbarui setiap kali. Pada penelitian ini digunakan optimizer *Adam* dengan *learning rate* $2x10^{-5}$.
- Memilih *Loss Function*. Tahap ini mengatur fungsi *loss* yang akan digunakan untuk mengukur kesalahan prediksi model selama pelatihan. Fungsi *loss* ini akan digunakan untuk mengarahkan proses optimasi dan membantu model belajar menghasilkan teks yang lebih akurat dan relevan. Hal ini dilakukan dengan menggunakan *loss function* seperti *Cross Entropy Loss*. *Loss* yang dihitung memberikan ukuran seberapa baik

performa model dan memandu proses pengoptimalan selama pelatihan dengan memastikan model dilatih dengan optimal dan menghasilkan teks yang berkualitas tinggi.

- Mengatur *'Trainer' and 'TrainerArgument'*. Tahap ini menggabungkan semua *hyperparameter* yang diatur untuk proses pelatihan model. Modul yang disediakan Hugging Face ini dipakai karena kepraktisannya dalam mengatur dan melatih model yang diinginkan.
- *Training Loop.* Tahap ini mengatur proses pelatihan model selama beberapa perulangan dan kumpulan data. Dalam *loop* ini, setiap kumpulan data yang diberi token diproses secara berurutan, melewati model untuk menghasilkan prediksi dan menghitung nilai *loss*. Proses pelatihan ini diulangi hingga model mencapai performa yang optimal sehingga menyempurnakan kemampuan model untuk menghasilkan teks yang berhubungan.
- *Evaluation*. Tahap ini menilai performa model dengan *test dataset*. Proses ini melibatkan pengumpulan data, analisis data, dan interpretasi hasil untuk mencapai kesimpulan. Metrik yang dipakai dalam perhitungan performa model ini adalah *perplexity*.

3.3.4. *Single-to-Multi-Label (S2M) Sampling*

Setelah model CLM mampu menghasilkan data dengan input label yang diinginkan, data yang dihasilkan akan dilakukan proses *sampling*. Proses ini bertujuan untuk mengambil data yang termasuk *multi-label* dengan target label yang diinginkan.



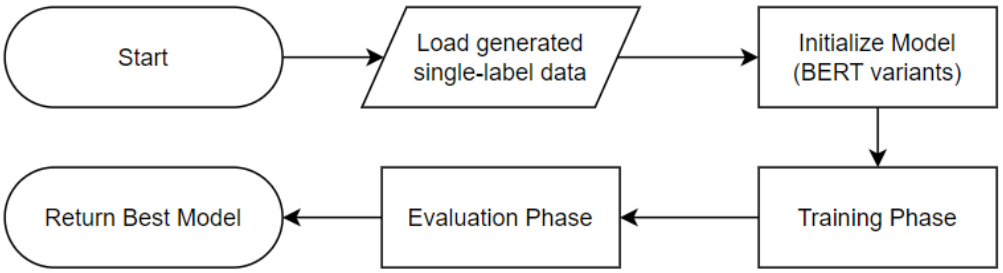
Gambar 3.5 Alur Metode Sampling *single-to-multi-label*

Gambar 3.4 menjelaskan alur dari metode sampling *single-to-multi-label* . Berikut adalah penjelasan lebih detail untuk setiap langkah pada alur tersebut.

- *Markov Chain Monte Carlo* (MCMC) *Sampling*. Tahap ini digunakan untuk menjelajahi ruang kombinasi *multi-label* dengan mengambil sampel kombinasi label baru secara iteratif. Ini menggunakan algoritma *Metropolis-Hastings* (MH) untuk mengusulkan dan menerima atau menolak sampel berdasarkan kecocokan sampel terhadap distribusi target kelas gabungan. Pada dasarnya, MCMC sampling membantu menghasilkan sampel multi-label yang beragam dengan cara menjelajahi dan menyempurnakan ruang kombinasi label yang mungkin
- *Latent Adaptation*. Tahap ini berfungsi untuk mencari tahu bagaimana berbagai karakteristik terhubung dalam latar belakang dari label positif tunggal yang dimulai pertama. Ini dilakukan dengan menganalisis pola tersembunyi dalam data yang dihasilkan oleh model.
- Menghasilkan Sampel *Multi-label*. Tahap ini menggabungkan sampel awal yang dihasilkan oleh model generasi dengan distribusi *latent* yang diestimasi. Dengan cara ini, metode ini mengambil sampel *multi-label* yang beragam yang mewakili berbagai kombinasi atribut terkait dengan label positif tunggal.
- *Post-processing*. Tahap ini menyempurnakan data yang dihasilkan dengan menerapkan teknik tambahan seperti augmentasi data, atau penyisipan noise. Teknik-teknik ini bertujuan untuk meningkatkan keakuratan dan variasi dari sampel *multi-label*, memastikan bahwa mereka lebih baik merepresentasikan karakteristik yang diinginkan dari distribusi data target.

3.3.5. Build dan Train Model BERT variants

Untuk membuat model klasifikasi, diperlukan tahap build dan training dari model yang akan dipakai. Dapat dilihat pada Gambar 3.6, model yang akan diuji adalah beberapa model varian BERT seperti BERT, RoBERTA, dan DistilBERT untuk dilakukan pengecekan model yang memiliki kemampuan klasifikasi terbaik. Model yang dipakai berupa model *pretrained* agar meminimalisir biaya training dari model bahasa yang memerlukan data *train* yang sangat besar jika dimulai dari awal.



Gambar 3.6 Alur Model Klasifikasi dari Varian BERT

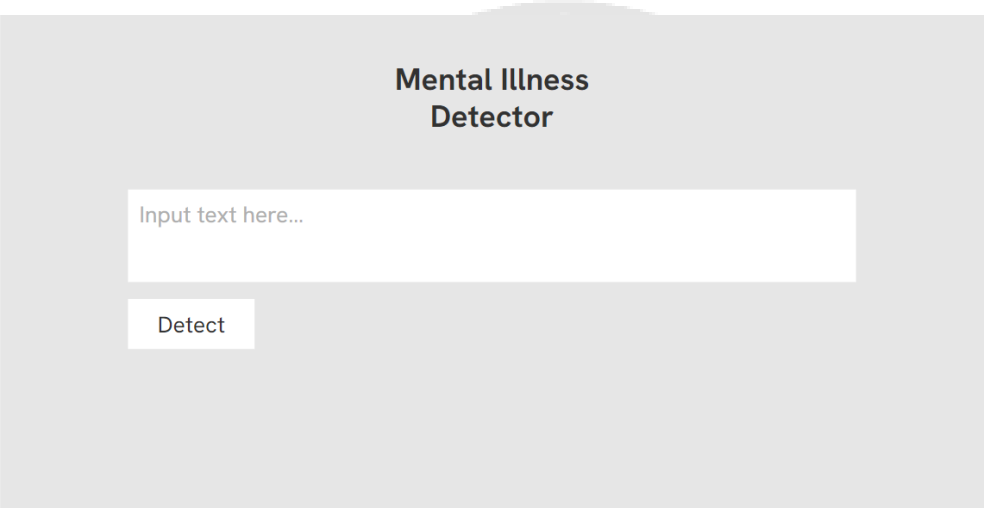
Gambar 3.6 menjelaskan alur pembuatan model klasifikasi dari varian BERT. Berikut adalah penjelasan lebih detail untuk setiap langkah pada alur tersebut.

- Inisialisasi model. Tahap ini dimulai dengan menginisialisasi model varian BERT (BERT, RoBERTA, atau DistilBERT) dari Hugging Face. Kemudian dilakukan modifikasi pada *output layer* berupa *feed forward neural network* dengan aktivasi *sigmoid*. Layer ini dipakai agar menghasilkan nilai biner 0 atau 1 untuk setiap label.
- Tahap *training*. Pada tahap ini, parameter model varian BERT yang disesuaikan diperbarui secara berulang menggunakan *backpropagation* dan *gradient descent*. Setiap iterasi, *batch* sampel teks dan label yang sesuai dimasukkan ke dalam model untuk menghasilkan prediksi, yang kemudian dibandingkan dengan label sebenarnya untuk menghitung kerugian. Parameter model disesuaikan untuk meminimalkan kerugian ini, mengoptimalkan kemampuannya untuk memprediksi label secara akurat untuk tugas klasifikasi multi-label. Proses dalam training ini diantaranya:
  - Generasi *batch*: Sampel teks dan labelnya dikelompokkan ke dalam beberapa *batch* untuk *train* yang efisien.
  - *Forward Pass*: Setiap *batch* dilewatkan melalui model varian BERT yang telah disesuaikan.
  - Perhitungan *loss*: Fungsi *BCEWithLogitsLoss* menghitung *loss* antara label prediksi dan label *ground truth*.
  - *Backward Pass*: *Gradient* dihitung dan digunakan untuk memperbarui parameter model melalui *backpropagation.*
  - *Perulangan*: Proses pelatihan diulangi dalam beberapa periode.
- Tahap evaluasi. Dalam tahap ini dilakukan penilaian kinerja model yang telah dilatih pada *dataset* terpisah. Ini menghitung berbagai metrik evaluasi, seperti akurasi, presisi, *recall*, dan *F1-score*, dengan membandingkan prediksi model dengan label sebenarnya. Metrik-metrik ini memberikan wawasan tentang seberapa baik model

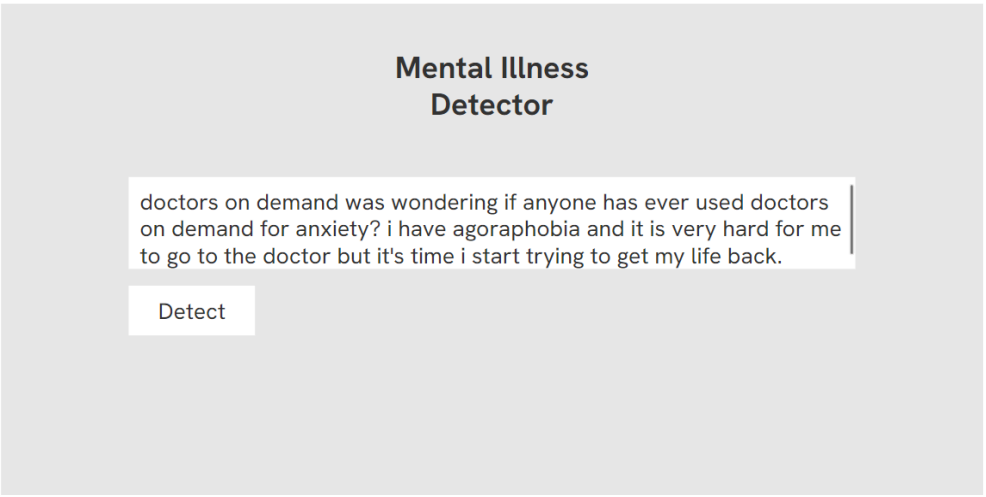
menggeneralisasi data yang belum pernah dilihat sebelumnya dan dapat memandu perbaikan lebih lanjut.

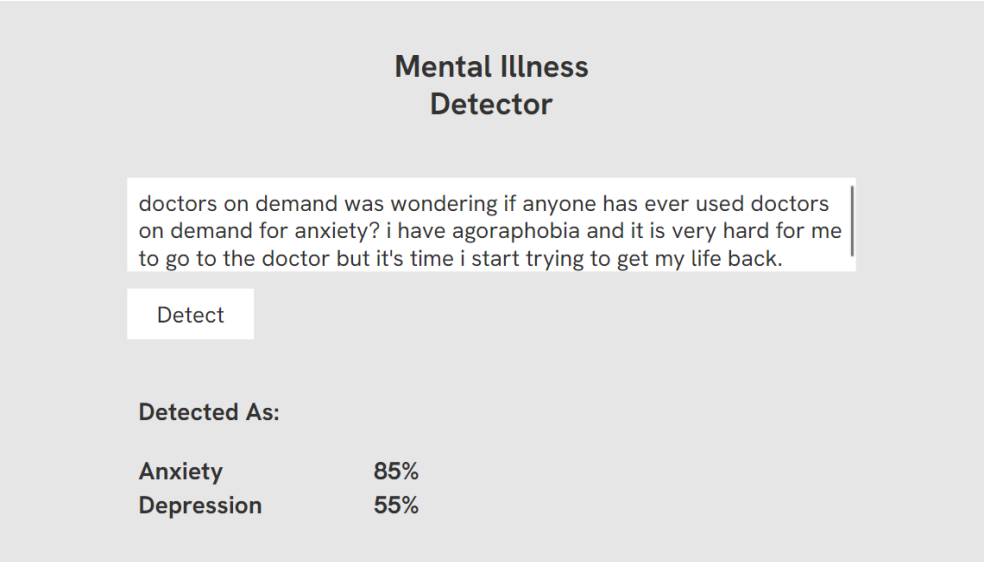
3.4. Desain Interface

Tampilan dibuat untuk menampilkan hasil klasifikasi teks terdeteksi gangguan mental apa saja berdasarkan teks yang dimasukkan pada aplikasi. Pada Gambar …. Terdapat beberapa bagian halaman utama seperti sebuah *text field* sebagai bagian untuk memasukkan kata yang akan dicek. Setelah itu ada tombol "*CHECK*" untuk menjalankan program. Ketika program dijalankan, *output* yang ditampilkan adalah label dari beberapa gangguan mental yang terdeteksi model.



Gambar 3.7 Tampilan awal program pendeteksi penyakit mental





Gambar 3.8 Tampilan ketika program dijalankan