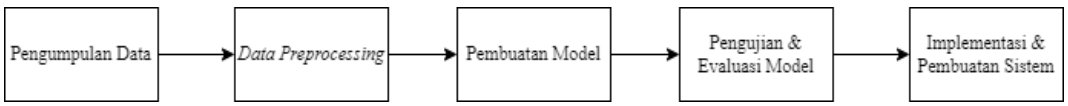
BAB III

ANALISIS DAN PERANCANGAN

Pada bab ini akan dijelaskan metode pengembangan aplikasi yang digunakan, meliputi analisis kebutuhan, perancangan proses dan tampilan antarmuka aplikasi, serta sumber daya yang dibutuhkan dalam pengembangan aplikasi.

3.1 Tahapan Penelitian

Tahapan penelitian adalah kerangka kerja yang digunakan dalam melakukan suatu penelitian Dalam penelitian ini, dirancang sebuah alur tahapan penelitian yang akan digunakan sebagai acuan dalam proses penelitian agar penelitian yang dilakukan dapat berjalan dengan baik, sistematis, dan menghasilkan output sesuai dengan yang diharapkan. Gambar 3.1 merupakan alur tahapan penelitian yang akan digunakan dalam penelitian ini.



Gambar 3.1 Visualisasi alur tahapan Penelitian

Penelitian ini dilakukan dengan tahapan-tahapan penelitian yang meliputi pengumpulan dan persiapan seluruh data yang dibutuhkan untuk proses pembuatan model, pengolahan data dengan melakukan *data profiling* untuk mengenali data yang akan digunakan, proses data preprocessing, pembagian data menjadi data *training*, validation, dan testing dengan metode *K-Fold Cross Validation*, pembuatan model dengan melakukan *training* pada pre-*trained* model BERT yang akan menghasilkan model yang sudah dilatih, lalu model akan dievaluasi dengan

menggunakan confusion matrix yang digunakan untuk menghitung performance metrics, dan langkah terakhir setelah model terbaik didapatkan adalah dengan mengimplementasikan sistem yang bisa dapat diakses untuk melakukan klasifikasi judul artikel berbahasa Inggris.

3.2 Analisis Kebutuhan

Pada penelitian ini penulis menggunakan perangkat lunak maupun perangkat keras komputer yang dapat membantu penulis dalam melakukan pengujian. Adapun perangkat yang akan digunakan dalam penelitian ini adalah sebagai berikut:

3.2.1 Perangkat Lunak

Perangkat lunak yang digunakan untuk penelitian ini berdasarkan analisis kebutuhan antara lain:

1. Windows 10
2. Google Chrome Web Browser
3. Google Colab Pro
4. *Library* Tensorflow
5. *Library* HuggingFace
6. *Library* SciKit Learn
7. Google Drive
8. Bahasa Pemrograman Python 3.9
9. Anaconda versi 4.10.3
10. Jupyter Notebook
11. Microsoft Excel

12. Framework Streamlit
13. Figma
14. GitHub

3.2.2 Perangkat Keras

Untuk menggunakan *environment* Anaconda pada sistem operasi Windows, dibutuhkan spesifikasi sistem seperti berikut:

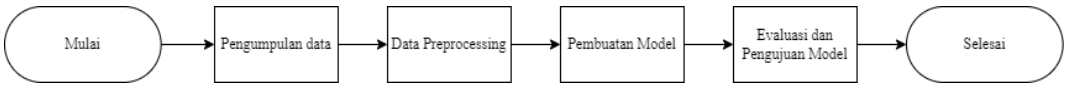
Sistem operasi	:	Windows 10
Arsitektur	:	x64 i.e 64 bit
Disk Space	:	5 GB atau lebih

Pada penelitian ini menggunakan *personal computer* yang memenuhi spesifikasi di atas. Spesifikasi *personal computer* penulis adalah sebagai berikut:

Manufaktur	:	Dell
Prosesor	:	Intel Core i7 8750H
GPU	:	NVIDIA GeForce GTX 1060 6 GB GDDR5 Max-Q
RAM	:	16 GB
Sistem Operasi	:	Windows 10
Arsitektur	:	x64 i.e 64 bit

3.3 Perancangan Proses Model Klasifikasi

Tahap perancangan sistem berisi penjelasan alur pengembangan dalam membuat rancangan dari model klasifikasi akan dibangun dan digunakan ke dalam bentuk aplikasi. Langkah-langkah yang dilakukan adalah dimulai dari pembagian data, *exploratory data analysis*, *preprocessing*, dan pelatihan model. langkah-langkah yang dilakukan dapat dilihat pada flowchart berikut.



Gambar 3.2 Langkah-langkah perancangan Proses Model Klasifikasi

3.3.1 Pengumpulan Data

Data yang akan digunakan dalam penelitian ini adalah data judul artikel berbahasa Inggris. Setelah melakukan riset dengan membaca dan memahami penelitian terkait yang pernah dilakukan sebelumnya, diputuskan bahwa dalam penelitian ini akan menggunakan *News Aggregator Data Set* (Dua and Graff, 2017). *Dataset News Aggregator* dipilih karena merupakan *dataset* artikel berbahasa Inggris dengan jumlah besar yang tersedia di publik dengan total jumlah 422.937 artikel. *Dataset* ini disediakan oleh *Artificial Intelligence Lab @ Faculty of Engineering, Roma Tre University – Italy.*

Dataset dikelompokkan ke dalam kelompok yang mewakili halaman yang membahas berita yang sama. Dalam *dataset* terdapat 422.937 halaman berita dan terbagi atas: 152.746 berita kategori bisnis, 108.465 berita kategori sains dan teknologi, 115.920 berita kategori bisnis, 45.615 berita kategori kesehatan, 2.076 klaster berita serupa untuk kategori hiburan, 1.789 klaster berita serupa untuk kategori sains dan teknologi, Kumpulan berita serupa 2.019 untuk kategori bisnis, 1.347 klaster berita serupa untuk kategori kesehatan. Data yang akan digunakan untuk proses pembuatan model sejumlah 20.000 judul artikel yang berisi 5.000 data per-kategori. *Dataset* mempunyai 8 atribut dalam *dataset* ini, yaitu:

Tabel 3.1 Daftar atribut dalam *dataset*

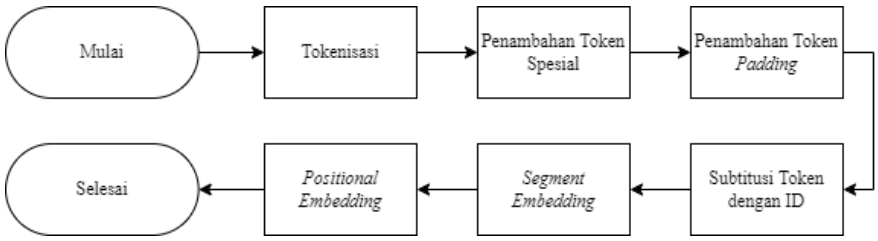
No.	Atribut	Keterangan
1.	ID	Pengidentifikasi unik artikel

No.	Atribut	Keterangan
2.	TITLE	Judul artikel
3.	URL	Alamat halaman artikel
4.	PUBLISHER	Nama penerbit
5.	CATEGORY	Kategori artikel berita
6.	STORY	Isi certia artikel
7.	HOSTNAME	Alamat halaman penertbit
8	TIMESTAMP	Waktu saat artikel diambil

Pada tahap pengumpulan data ini *dataset* akan dibagi menjadi data *training-validation* dan data *testing*. Pembagian data juga bisa disebut data *splitting*. *Dataset* yang akan digunakan adalah sejumlah 20.000 dari 422.937 judul artikel, dikarenakan keterbatasan perangkat keras yang digunakan google colab. Data ini harus dibagi menjadi data *training-validation* dan data *testing.* Pembagian ini dilakukan dengan rasio 90% data *training-validasi* dan 10% data *testing* agar didapatkan jumlah data yang optimal untuk proses *training.* Dengan menggunakan rasio tersebut dihasilkan data *training-validation* sejumlah 18.000 judul artikel dan data *testing* sejumlah 2.000 judul artikel. Pembagian data *training-validation* akan dilakukan pada tahap *cross validation.*

3.3.2 Data Preprocessing

Pada pembuataan model terdapat beberapa langkah yang sebelum melakukan pelatihan. Model *dataset* disesuaikan terlebih dahulu dengan representasi input yang dimiliki oleh BERT. Oleh karena itu, akan dilakukan tahap *word embedding* pada judul artikel yang akan dijadikan objek pembelajaran agar menghasilkan input yang bisa diproses oleh *pre-trained model*. Tahap *word embedding* ini dilakukan oleh BERT tokenizer. Gambar 3.3 menunjukkan *flowchart* untuk tahapan word *embedding.*



Gambar 3.3 Tahapan *Word Embedding*

Tahapan pertama dalam *word embedding* adalah tokenisasi. Tokenisasi bertujuan untuk memeriksa apakah semua kata pada input terdapat pada kamus BERT. Ada beberapa ketentuan yang dilakukan oleh BERT dalam melakukan tokenisasi yaitu: Jika kata terdapat pada kamus, BERT akan menggunakan seluruh kata pada input secara keseluruhan. Jika kata tidak terdapat pada kamus BERT maka tokenizer akan memecah kata tersebut menjadi sub kata yang disesuaikan dengan kamus BERT. Sub kata akan diawali dengan ##, kecuali untuk sub kata pertama. tokenisasi akan dilakukan menggunakan Tokenizer BERT base model (cased). Kamus tokenizer ini berisikan kata-kata *case-sensitive* yang berarti model ini peka huruf besar atau kecil contohnya kata mother dan Mother mempunyai token yang berbeda. Hal ini juga menyebabkan tidak perlu dilakukannya normalisasi pada *dataset*.

Tahap kedua dalam *word embedding* adalah penambahan token spesial, dalam Setiap input pada BERT akan diberikan dua token spesial, token [CLS] di awal kalimat dan [SEP] di akhir kalimat. Token [CLS] merupakan token yang menandakan awal kalimat serta digunakan untuk klasifikasi. Token [SEP] merupakan token yang digunakan sebagai penanda akhir kalimat ataupun pemisah antar kalimat.

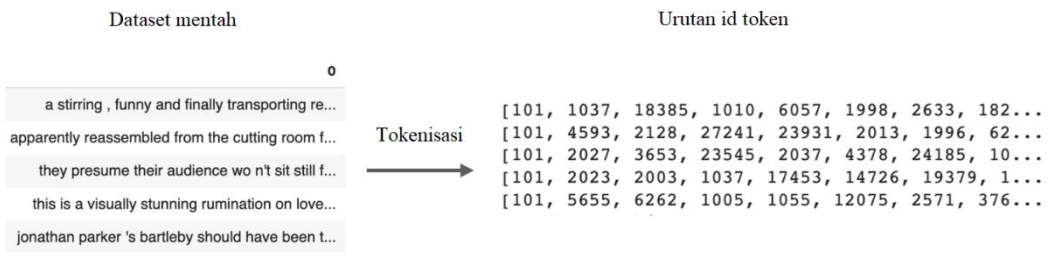
Input dalam BERT kemudian akan disesuaikan panjangnya dengan panjang maksimum yang telah ditentukan. Penyesuaian panjang kalimat ini dilakukan dengan melakukan penambahan token [PAD]. Tiap input tersebut kemudian akan disesuaikan dengan *ID* sesuai dengan kamus yang digunakan oleh BERT. *ID* ini diperoleh dari tahap *training* model BERT, penyusunan *ID* pada kamus didasari oleh tingkat kemunculannya. Subtitusi ini perlu dilakukan, karena BERT hanya memahami token dari *ID*.

Segment Embedding diberikan pada input untuk membedakkan antara kalimat pertama dan kedua dari input. Hal ini dilakukan dengan melihat posisi dari token [SEP] yang memiliki fungsi untuk memisahkan kalimat. Selain itu pada segment embedding token akan dibedakan antara input dan *padding*. Hal ini ditandai dengan pemberian angka pada input, serta untuk *padding*.

Positional Embedding ditambahkan input untuk mengetahui posisi tiap kata pada kalimat. Tahap ini dilakukan karena BERT tidak mengetahui atau mengenali posisi dari tiap kata. Misalkan terdapat dua token yang sama pada sebuah kalimat, akan tetapi kedua token ini memiliki makna yang berbeda pada kalimat tersebut. BERT akan memberikan perlakuan yang berbeda pada kedua token tersebut.

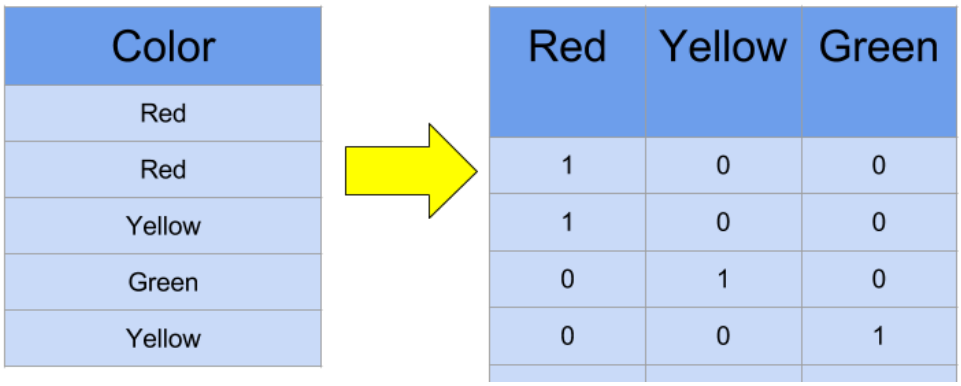
Semua proses tersebut akan mengembalikan hasil *input ids* yang berisi token ID yang di-input-kan, dan *attenion mask* yang berisi urutan *array* yang berisikan nilai 1 atau 0. Nilai 1 pada *array* berarti token tersebut merupakan token dari teks asli dan nilai 0 berarti token tersebut merupakan token dari hasil *padding*, sedangkan nilai 1 berarti token tersebut merupakan token dari teks asli. Hal ini bertujuan agar self-attention layer dari BERT tidak menghiraukan token *padding*

sehingga tidak akan mempengaruhi hasil dari model yang di-*training*. Gambar 3.4 adalah contoh hasil proses tokenisasi.



Gambar 3.4 Contoh hasil tokenisasi

Setelah melakukan *word embedding* pada judul artikel *dataset* akan dilakukan *one-hot encoding*. *One-Hot encoding* adalah salah satu metode *encoding*. *Encoding* ini dilakukan agar model *pre-train* dapat membaca kategori setiap data latih. Metode ini merepresentasikan data bertipe kategori sebagai vektor biner yang bernilai integer, 0 dan 1, dimana semua elemen akan bernilai 0 kecuali satu elemen yang bernilai 1, yaitu elemen yang memiliki nilai kategori tersebut. Metode ini digunakan pada semua kategori yang akan dipakai. Gambar 3.4 adalah contoh hasil proses *one-hot encoding*.



Gambar 3.5 Contoh penggunaan proses *one-hot encoding*

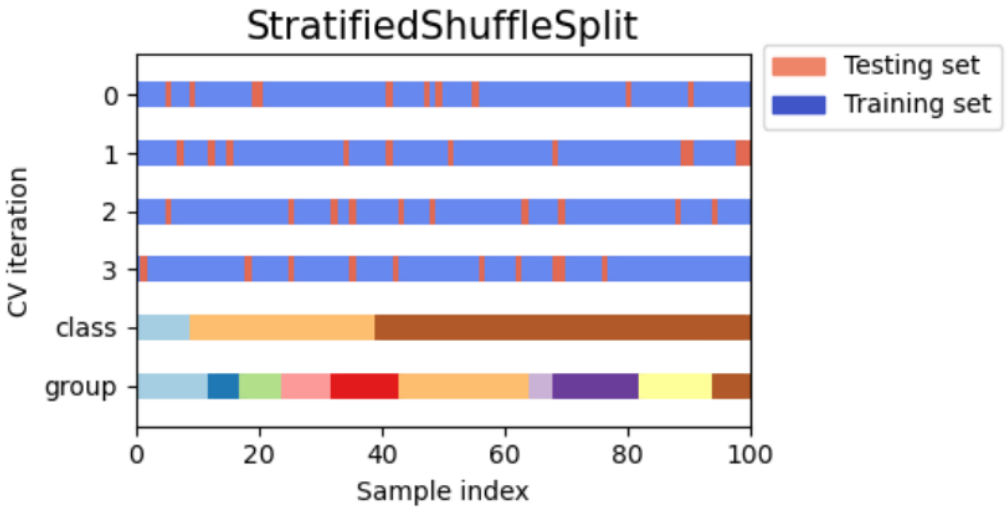
3.3.3 Pembuatan *Model*

Setelah dilakukannya data *preprocessing dataset*, harus dibuatkan sebuah data *pipeline.* berguna agar membantu menghemat penggunaan memori, mempercepat proses pengambilan data, meningkatkan kecepatan *training* model. Data *pipeline* berperan untuk membuat *source dataset* dari data input dan mengiterasi *dataset* tersebut. *Source dataset* yang digunakan akan dibagi menjadi dua yaitu *dataset training* dan *dataset* validasi. Tahap ini akan digunakan pada tahap *cross-validation.*

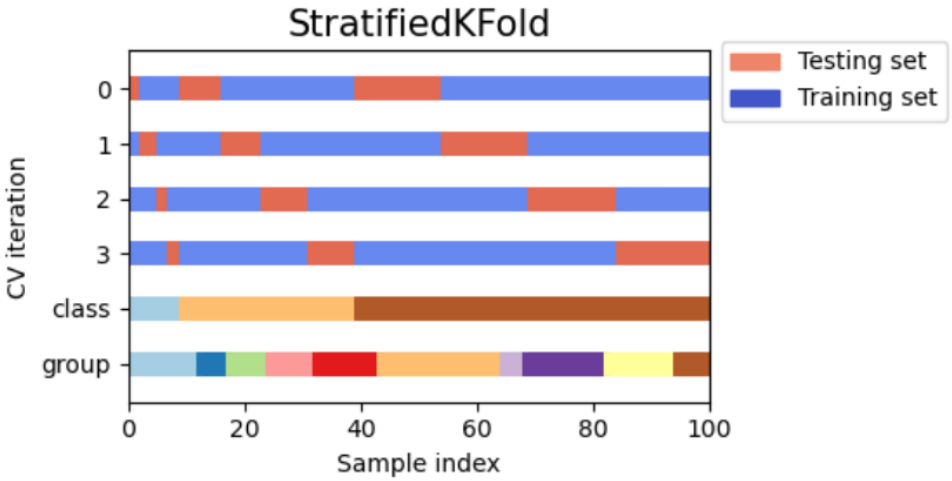
Cross validation merupakan salah satu cara yang dapat dilakukan untuk mengurangi terjadinya *overfitting* pada model. *Overfitting* dapat menyebabkan kegagalan model memprediksi dengan tepat jika diberikan *dataset* yang berbeda dari yang digunakan ketika model melakukan *training. Cross validation* mengatasi permasalahan tersebut dengan melakukan *training* beberapa kali sesuai dengan jumlah *folds* yang dideklarasikan dengan komposisi data *training* dan data *testing* yang berbeda untuk setiap *fold*-nya. Pada penelitian ini, jenis *cross validation* yang diaplikasikan adalah *Startified Shuffle Split Cross Validation*.

Startified Shuffle Split Cross Validation sedikit berbeda dengan *K-Folds Cross Validation* yang biasa digunakan. *StratifiedShuffleSplit* adalah kombinasi dari *Shuffle Split* dan *Stratified K-Fold*. Menggunakan *StratifiedShuffleSplit* rasio distribusi label kelas hampir merata antara data *train* dan data validasi, distribusi inilah mengapa dinamakan *stratified*. Perbedaan utama antara *Stratified Shuffle Split* dan *Stratified K-Fold* dengan *shuffle* adalah bahwa di *Stratified K-Fold*, kumpulan data dikocok hanya sekali di awal dan rasio data *training* dan data

validasi didasarkan oleh jumlah lipatan yang sudah ditentukan, dimana *Stratified Shuffle Split* data dikocok setiap lipatan dan jumlah rasio antara data *training* dan data validasi dapat ditentukan sendiri. Visualisasi pembagian data dengan *Stratified K-Fold* dan *Stratified Shuffle Split* dapat dilihat pada Gambar 3.6 dan Gambar 3.7.



Gambar 3.6 Visualisasi Pembagian *Stratified Shuffle Split* (der Walt *et al.*, 2014)



Gambar 3.7 Visualisasi *Stratified K-Fold* (der Walt *et al.*, 2014)

Setelah data sudah dibagi pembuatan model sudah bisa dilakukan. Langkah pertama yang dilakukan adalah memuat model *pretrain* BERT, model yang akan dimuat adalah versi $\text{BERT}_{\text{BASE}}$. Model yang sudah dimuat akan dicoba untuk

prediksi contoh judul artikel tanpa melakukan *training* model dan hanya mengandalkan model pre-*trained* dari $BERT_{BASE}$. Hasil yang didapat akan dibandingkan dengan hasil prediksi judul artikel dengan *trained* model.

Selanjutnya adalah proses *training* model sudah bisa dimulai, dengan melakukan *fine-tuning*. *Fine-tuning* adalah mengkonfigurasi model *pretrained* untuk membuatnya dapat melakukan tugas yang spesifik dengan baik. Tujuan dilakukan *fine-tuning* adalah agar hasil model yang telah dilatih dapat menghasilkan *output* prediksi klasifikasi yang optimal. Disini akan dilakukan proses *fine-tuning* menggunakan varian *pre-trained model* $BERT_{BASE}$ *uncased* dengan memanfaatkan *library* Transformers.

Untuk mendapatkan model yang baik, model yang telah dimuat harus dilatih dengan konfigurasi *hyperparameter* yang optimal. Dengan konfigurasi hyperparamter yang optimal bisa didapatkan hasil dengan kesalahan atau error yang rendah. *Hyperparameter* yang dapat diubah antara lain: *batch size, epoch, optimizer, activation function, learning-rate,* dan lainnya. Untuk menentukan nilai *hyperparameter* yang terbaik, perlu mempertimbangkan antara kemampuan perangkat yang digunakan, waktu yang dapat diluangkan untuk melakukan proses pelatihan, dan hasil yang didapatkan. Maka dari itu, pada penelitian ini akan dilakukan pencarian nilai *hyperparameter* terbaik.

3.3.4 Evaluasi

Tahap evaluasi adalah tahap dimana model dari tahap pembuatan diuji dan dinilai. Ditahap ini akan didapatkan informasi mengenai seberapa mampu model yang sudah dibuat. Model yang telah dibuat akan diuji dengan data *testing* dan

dievaluasi dengan hasil dari data *confusion matrix*, yang dimana data bisa diolah menjadi *perfomence metrics.*

Tahap *testing* ini akan dilakukan pada *dataset* pengujian atau *testing* menggunakan *dataset* dengan rasio 10% dari keseluruhan jumlah data yamg digunakan dengan jumlah 2000 data, dengan *performance metrics* yang dihitung yaitu *accuracy*, *precision*, *recall*, dan *f1-score*. Langkah evaluasi model yang dilakukan adalah dengan menggunakan *confusion matrix* yang digunakan untuk menghitung *performance metrics.*

Setelah hasil performance metrics didapatkan, selanjutnya adalah membandingkan hasil *performance metrics* yang didapatkan pada semua percobaan dengan *fine-tuning Hyperparameter* yang sudah ditentukan untuk mendapatkan *trained* model yang paling optimal. Berdasarkan hasil evaluasi yang telah dibandingkan, maka *trained* model dengan hasil yang terbaik akan dipilih untuk disimpan dan digunakan pada tahap implementasi sistem.

3.4 Perancangan *Sistem*

Tahap implementasi sistem merupakan tahap terakhir yang dilakukan dalam pengembangan sistem klasifikasi judul berita berbahasa Inggris. Yang akan dilakukan pada tahap ini adalah melakukan pengembangan sistem berbasis *website* dan melakukan pengujian sistem yang telah dibuat ke dalam bentuk *website*.

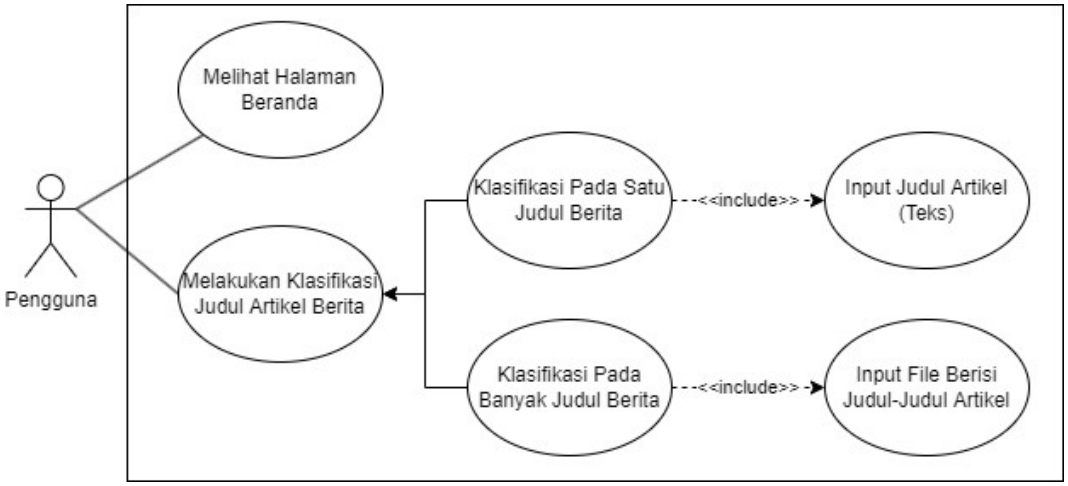
3.4.1 Perancangan UML

Setelah pembuatan model dan evaluasi sudah dijalankan dan didapatkan model yang terbaik, tahapan selanjutnya adalah untuk membangun sebuah aplikasi

berbasis web. Sebelum sistem ini dibuat, tahap pertama yang dilakukan adalah untuk merancang UML sebagai dasar pengembangan sistem informasi.

A. Use Case Diagram

Gambar 3.8 menggambarkan *use case diagram* dari sistem yang akan dikembangkan. Pada diagram tersebut terdapat satu aktor yang bernama User yang bertindak sebagai pengguna aplikasi. User dapat melakukan dua aksi di dalam aplikasi, yaitu melihat halaman beranda, melakukan klasifikasi judul berita berbahasa Inggris.



Gambar 3.8 *Use Case Diagram* Sistem

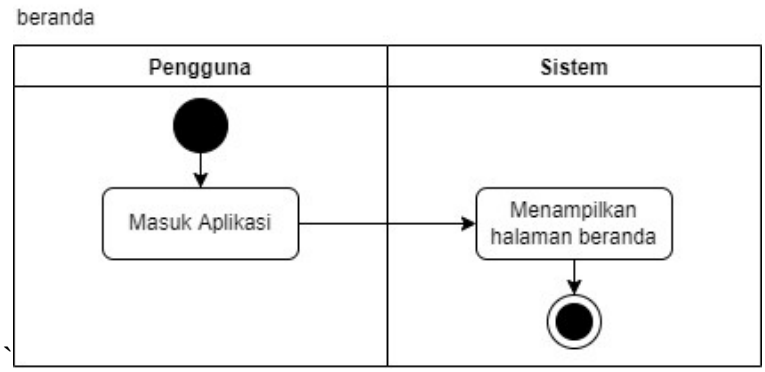
Saat memasuki aplikasi, pertama user akan melihat halaman beranda yang berisikan informasi mengenai aplikasi serta cara penggunaannya. Selanjutnya user dapat masuk ke halam klasifikasi berbahasa Inggris dan menginput data yang akan dianalisis. User dapat memasukkan satu judul data atau banyak judul data yang berbentuk file *excel*. *Use case* memiliki relasi *include* terhadap input data klasifikasi dikarenakan user harus menginputkan data terlebih dahulu sebelum mendapatkan hasil klasifikasi.

B. Activity Diagram

Activity diagram digunakan untuk menggambarkan rangkaian kegiatan pada setiap use case secara detail dari awal hingga akhir. Berikut activity diagram pada aplikasi yang akan dibangun.

1. Activity Diagram Beranda

Gambar 3.9 menggambarkan *activity diagram* pada *use case* melihat halaman beranda. Diagram dimulai dengan user masuk ke aplikasi yang akan menampilkan halaman beranda. Pada halaman tersebut user dapat melihat informasi mengenai aplikasi. Informasi yang akan disuguhkan kepada user adalah seperti tujuan pembuatan aplikasi, pengertian klasifikasi berita, metode BERT, *hypermarameter* yang digunakan pada model yang digunakan, serta fungsi dari setiap menu. Selain itu user juga dapat melihat bagaimana cara menggunakan aplikasi atau cara melakukan klasifikasi dengan menggunakan aplikasi.

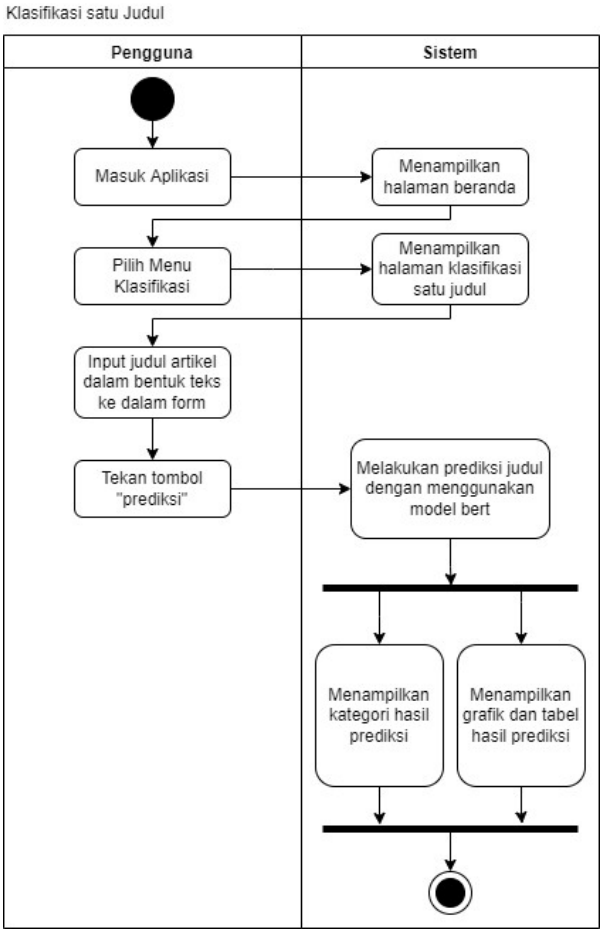


Gambar 3.9 *Activity Diagram* Halaman Beranda

2. Activity Diagram Input Satu Judul

Gambar 3.10 menunjukkan *activity diagram* pada *use case* input satu judul. Setelah masuk ke aplikasi dan melihat halaman beranda, untuk menginput data, user dapat memilih menu “Klasifikasi”. Setelah diarahkan ke halaman input data, user

dapat memilih antara klasifikasi satu judul atau klasifikasi banyak judul. Jika user memilih untuk klasifikasi satu judul, user dapat mengisi judul di dalam *text-box*. Setelah *text-box* diisi user tekan tombol prediksi. klasifikasi akan dapat dijalankan dan jika klasifikasi sudah selesai akan muncul hasil klasifikasi berisi hasil kategori prediksi dan tabel beserta grafik hasil prediksi.

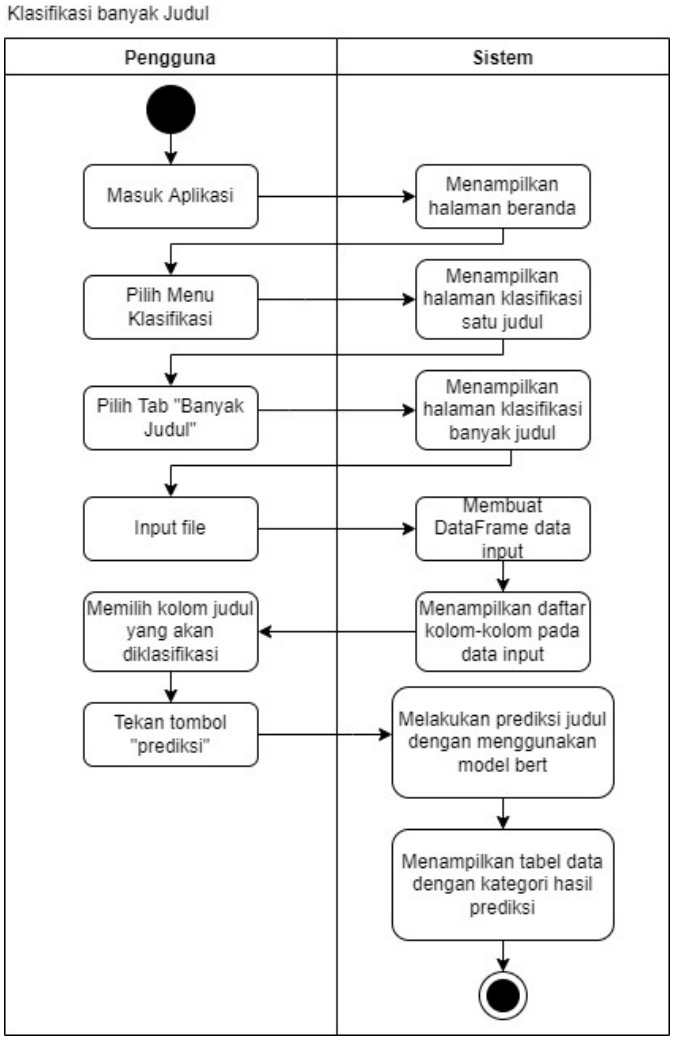


Gambar 3.10 *Activity Diagram* Input Data satu judul

3. Activity Diagram Input Banyak Judul

Gambar 3.11 menunjukkan *activity diagram* pada *use case* input satu judul. Setelah masuk ke aplikasi dan melihat halaman beranda, untuk menginput data, user dapat memilih menu “Klasifikasi”. Setelah diarahkan ke halaman input data, user

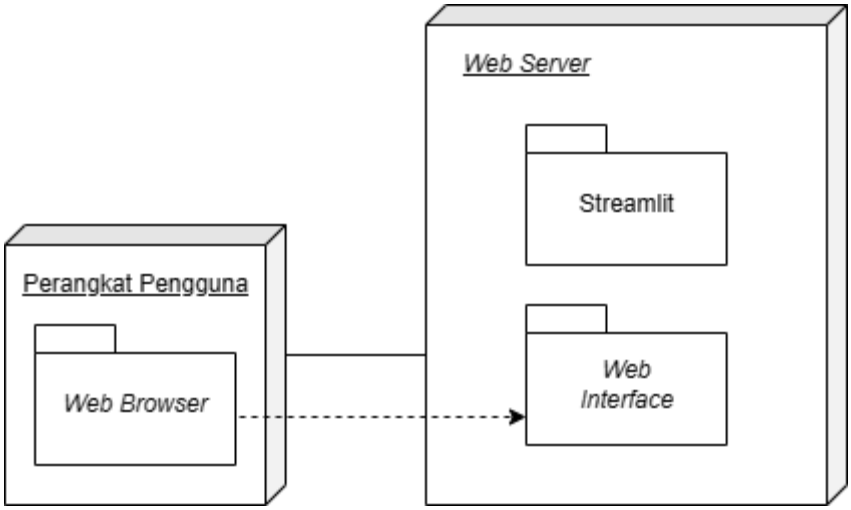
dapat memilih antara klasifikasi satu judul atau klasifikasi banyak judul. Jika user memilih untuk klasifikasi banyak judul, user dapat memasukkan file berbentuk *excel*. Setelah data dimasukkan sistem akan membuat *dataframe* dari data input yang akan diprediksi dan menampilkan daftar kolom pada data input. Setelah itu user harus memilih kolom man ayang berisikan kolom judul. Jika sudah memilih, user dapat menekan tombol klasifikasi dan sistem dapat menjalankan klasifikasi dan jika klasifikasi sudah selesai akan muncul hasil klasifikasi berisi hasil kategori prediksi dan tabel beserta grafik hasil prediksi. User juga dapat mengunduh data hasil prediksi.



Gambar 3.11 *Activity Diagram* Input Data satu judul

C. Package Diagram

Gambar 3.12 memperlihatkan deployment diagram pada sistem aplikasi yang dibangun. Deployment diagram memberikan gambaran sistem dalam bentuk streamlit. Sistem direpresentasikan oleh sekumpulan node, masing-masing dari node tersebut direpresentasikan dengan bentuk kubus. Garis yang menghubungkan antar node mengambarkan hubungan diantara kedua node tersebut. streamlit menjadi framework untuk keseluruhan fungsi sistem dengan web interface sebagai tampilannya dan dapat diakses oleh pengguna melalui browser pada perangkat masing-masing.



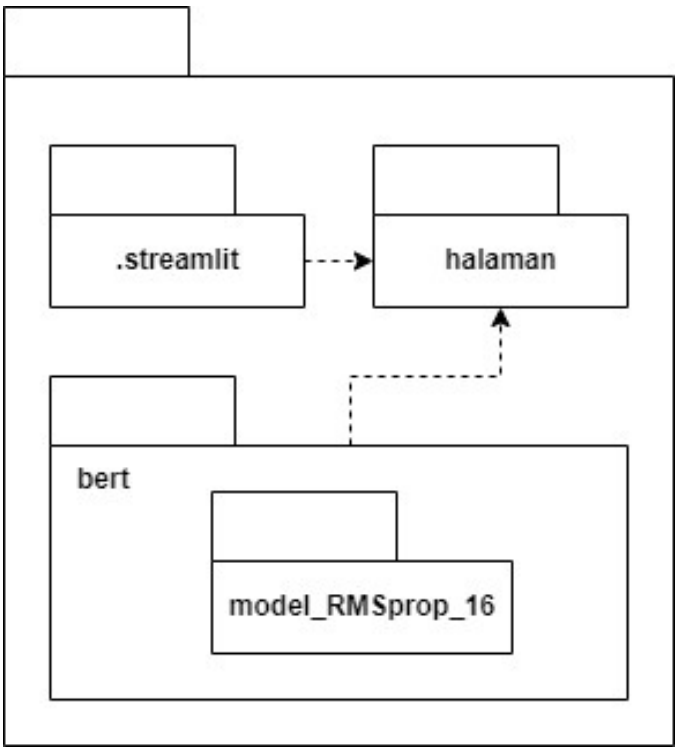
Gambar 3.12 *Deployment Diagram* Sistem Klasifikasi Judul Artikel Bahasa Inggris

D. Package Diagram

Gambar 3.12 menggambarkan *package diagram* dari sistem informasi yang akan dibangun. Dapat dilihat terdapat paket application yang didalamnya berisi paket-paket lainnya, yaitu _pycache_, .streamlit, halaman, dan bert. Paket halaman berisikan file-file python yang mengatur isi setiap halaman menu..Paket .streamlit

berfungsi untuk mengatur warna tema pada aplikasi. Paket bert berisikan model BERT dan kode untuk menjalankan prediksi.

Pada paket utama terdapat file utama untuk menjalankan aplikasi. Selain itu terdapat pula file requirement.txt yang berisikan semua *package* yang digunakan dalam pengembangan aplikasi. File requirement.txt ini digunakan untuk keperluan deploy aplikasi.



Gambar 3.13 *Package Diagram* Sistem

3.4.2 Perancangan Halaman Antar Muka Aplikasi

Sebelum pengembangan aplikasi dilakukan, dibuat perancangan halaman antar muka aplikasi terlebih dahulu sebagai panduan. Rancangan ini dibuat pada seluruh halaman yang akan terdapat pada aplikasi. Perancangan halaman antar muka aplikasi dilakukan dengan menggunakan *software* Figma.

A. Rancangan Halaman Beranda

Gambar 3.13 merupakan desain tampilan halaman beranda yang merupakan halaman pertama yang akan ditemui oleh pengguna saat mengakses aplikasi. Pada bagiansamping kiri halaman ini terdapat dua pilihan menu yang dapat dipilih untuk mengganti halaman. Konten utama pada halaman ini adalah berupa informasi mengenai aplikasi yang berbentuk teks. Halaman beranda dirancang dengan tujuan untuk memberikan informasi selengkap-lengkapnya mengenai aplikasi agar pengguna aplikasi dapat mengerti fungsi aplikasi dan cara menggunakannya.

Gambar 3.14 Rancangan Halaman Beranda

B. Rancangan Halaman Input Satu Judul

Pada halaman input data sama seperti halaman beranda, dibagian samping kiri halaman terdapat pilihan menu dengan warna tombol menu input data berbeda

untuk menandakan bahwa pengguna sedang membuka halaman tersebut. Dan dibagian atas ada dua pilihan menu klasifikasi satu judul atau banyak judul dengan warna yang berbeda untuk menandakan pengguna sedang membuka halaman tersebut Konten utama pada halaman ini merupakan tempat user untuk melakukan klasifikasi satu judul. Di halaman ini terdapat kotak dimana pengguna dapat mengisi teks judul yang akan diprediksi dan hasilnya. Gambar 3.14 adalah rancangan halaman input satu judul.

Gambar 3.15 Rancangan Halaman Input Data

C. Rancangan Halaman Input Satu Judul Data

Pada halaman input data sama seperti halaman beranda, dibagian samping kiri halaman terdapat pilihan menu dengan warna tombol menu input data berbeda

untuk menandakan bahwa pengguna sedang membuka halaman tersebut. Dan dibagian atas ada dua pilihan menu klasifikasi satu judul atau banyak judul dengan warna yang berbeda untuk menandakan pengguna sedang membuka halaman tersebut Konten utama pada halaman ini merupakan tempat user untuk melakukan klasifikasi satu judul. Di halaman ini terdapat kotak dimana pengguna dapat mengimpor *file* berisi judul yang akan diprediksi dan hasil prediksinya. Gambar 3.15 adalah rancangan halaman input banyak judul.

Gambar 3.16 Rancangan Halaman Input Banyak Data