

BAB 3. ANALISIS DAN DESAIN SISTEM

Pada bab ini dibahas mengenai Tahap Analisis Data dimana mencakup Pengumpulan Data, Representasi Data, dan Normalisasi Data. Tahap Analisis Sistem dimana mencakup *Preprocessing* Data, Pengambilan Fitur, Model *Neural Network* untuk *Training*, dan Sistem untuk *Testing*. Tahap Desain Sistem dimana mencakup Desain Aplikasi Android, Desain *Web Report Model Neural Network*.

3.1. Analisis Data

Analisis data meninjau permasalahan yang berorientasi pada data, mulai dari Pengumpulan Data, Representasi Data, dan Normalisasi Data. Data adalah bahan utama pada penelitian ini, semua diproses menggunakan data. Data yang digunakan dalam penelitian ini adalah data *raw audio* berformat WAV.

3.1.1. Pengumpulan Data

Data dikoleksi dengan dua cara, dataset pertama diambil dari rekaman langsung melalui *smartphone* dengan konfigurasi 16kHz dan *mono channel*. Data rekaman suara dikondisikan tanpa ada gangguan / *noise*. Data sudah terkumpul sebanyak 349 kalimat dari 14 orang dengan total waktu suara 34 menit. Dataset kedua sebanyak 212 kalimat dari 10 orang diambil dari rekaman luar dari video *Youtube*, rekaman pengumuman gereja dan *speech synthesiser Apple* dan *Bing*, suara diseleksi dengan memilih bagian - bagian suara yang bersih tanpa ada gangguan / *noise*. Dataset asli kedua memiliki konfigurasi standar yaitu 44.1kHz sehingga dilakukan konversi dengan menggunakan Audacity menjadi 16kHz. Dataset yang terkumpul akan dibagi menjadi proporsi training 90%, dev 10%.

3.1.2. Representasi Data

Dataset disimpan dengan kode unik nama dengan dua ekstensi *file* berbeda .WAV dan .TXT, dimana .WAV adalah dataset suara, .TXT adalah target label suara. Charset / set karater yang akan menjadi target label suara dapat berupa grafem / fonem, dimana semua .TXT harus konsisten mengikuti charset / set

karakter. Karena pemodelan grafem sama seperti huruf biasa sehingga charset / set karakter adalah { a, b, c, d ..., z, SPACE }, namun untuk pemodelan fonem tidak bisa mengikuti huruf latin biasa, ada beberapa fonem yang disuarakan berbeda dengan penulisan huruf latin-nya, misal fonem <ng> dimisalkan η, <ny> dimisalkan ñ, fonem <e> dapat berbunyi dua macam yaitu kata menang di simbolkan mΘnaη, sate disimbolkan sate.

3.1.3. Normalisasi Data

Pada normalisasi untuk data suara, digunakan *Peak Normalization*. *Peak Normalization* berarti menormalisasikan *peak data* / amplitudo yang paling maksimum sebagai referensi nol-nya. Dalam data suara dihitung terlebih dahulu desibel sebagai target desibel yang diinginkan dengan rumus (3.1).

Hitung peak dari *wavelength* suara dengan mencari amplitudo maksimum, setelah didapatkan dihitung untuk fungsi normalisasi pada setiap *wavelength* suara pada waktu T dengan rumus (3.2).

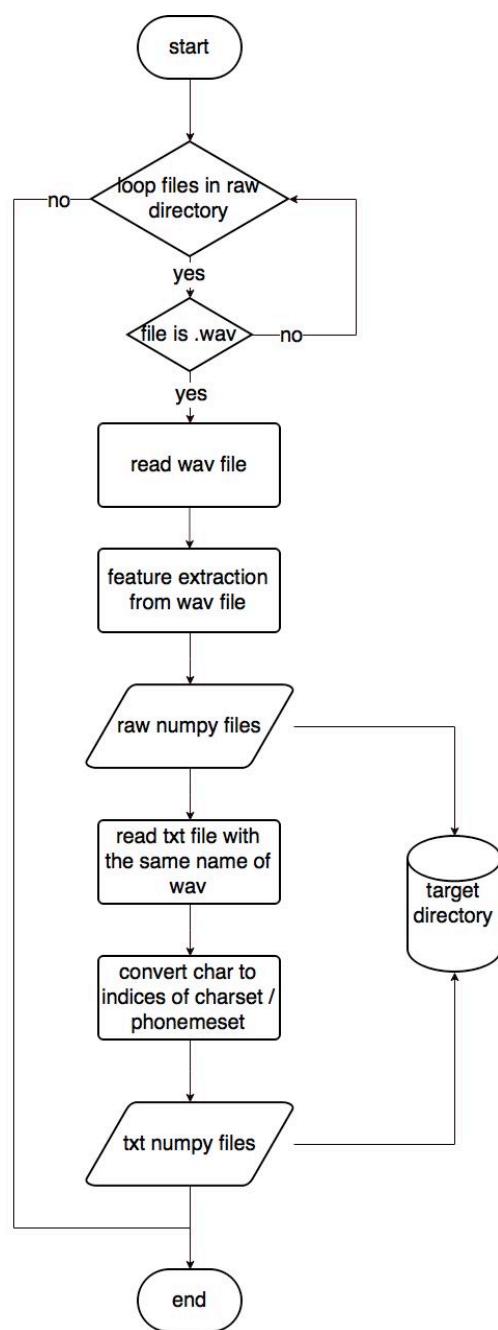
3.2. Analisis Sistem

Analisis sistem membahas permasalahan setelah memproses data. Sistem akan mengolah data suara menjadi sebuah output berupa kata – kata. Sistem mencakup *Preprocessing* Data, Pengambilan Fitur, Model *Neural Network* untuk *Training*, dan Sistem untuk *Testing*.

3.2.1. Preprocessing Data

Dataset pertama memiliki perbedaan channel dengan dataset kedua. Pengambilan fitur / ekstraksi fitur tidak dapat dilakukan ketika dataset pertama dan kedua digabungkan, sehingga perlu dilakukan pengambilan 1 kolom saja untuk dataset suara yang *stereo channel* sehingga dihasilkan keduanya mono channel. Proses preprocessing data juga melakukan konversi data dari .WAV

menjadi .NPY (*numpy file*), agar memudahkan saat training dan dev. *Numpy file* berisi *file* biner yang menyimpan nilai matriks hasil dari ekstraksi fitur. Data yang dihasilkan ada dua *file* dengan format berikut, yaitu <filename>.NPY dan _<filename>.NPY. Karakter <_> pada awal nama *file* menunjukan sebagai dataset target label suara. Pada fitur ekstraksi MFCC (*Mel-Frequency Cepstral Coefficient*), dilakukan pengambilan n konteks frame dari t-n, frame t, dan frame t+n. Dimana ditambahkan pada indeks fitur.



Gambar 3.1. *Flowchart* / diagram alur mekanisme *preprocessing* data

Flowchart / diagram alur pada Gambar 3.1 dimulai dengan memberikan argumen sebagai berikut :

1. *Raw directory*, sebagai tempat *directory / folder* yang akan dibaca dan berisi file .wav dan .txt.
2. *Target directory*, sebagai tempat tujuan / *target directory / folder* yang akan dituju ketika menuliskan hasil outputnya berupa .npy (*numpy file*).
3. *Feature type*, definisi tipe fitur yang akan di ekstrak, misalnya *mfcc / spectrogram*.

Sehingga ketika dijalankan, proses ini akan secara otomatis membaca semua file

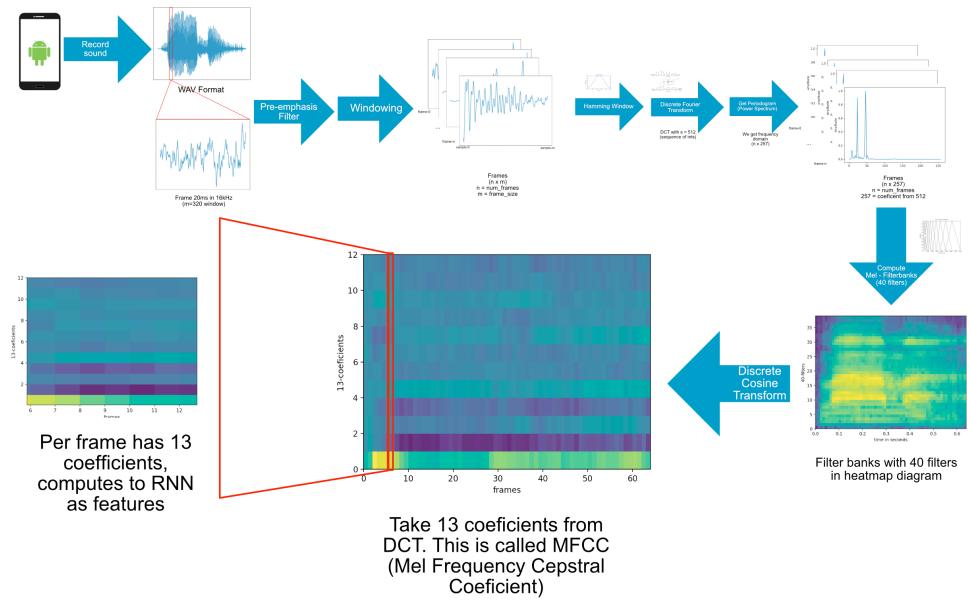
4. *Num of context*, jumlah n konteks yang akan ditambahkan ke dalam indeks fitur (t-n)-t-(t+n).

Sehingga ketika dijalankan, proses ini akan secara otomatis membaca semua file

yang ada di *raw directory* dengan memfilter .wav saja dengan catatan untuk .txt file harus memiliki nama yang sama dengan .wav. Maka akan dihasilkan dua file output ber-ekstensikan .npy (*numpy file*).

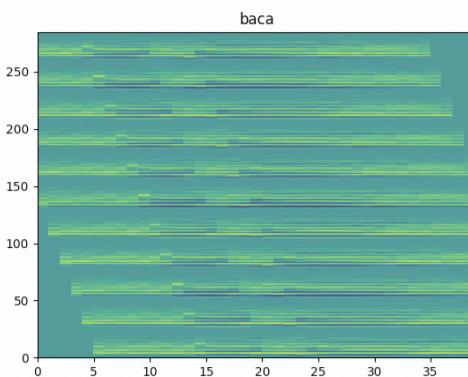
3.2.2. Pengambilan Fitur

Pengambilan fitur / ekstraksi fitur yang digunakan adalah MFCC (Mel-Frequency Cepstral Coefficients). MFCC adalah koefisien dari koleksi yang dihasilkan oleh MFC (Mel-Frequency Cepstrum). Jumlah koefisien yang digunakan adalah 13. Ukuran *frame* yang digunakan setiap waktunya adalah 0.002 detik / 20 milidetik dengan *striding* / pergeseran sebanyak 0.001 detik / 10 milidetik. Pada Gambar 3.2 *flowchart / diagram pengambilan fitur MFCC dari WAV audio file*.



Gambar 3.2. Flowchart / diagram alur pengambilan fitur MFCC dari WAV audio file

Data yang dihasilkan berupa data matriks / array 2 dimensi dengan asumsi baris adalah banyak frame T (*time*), kolom adalah jumlah koefisien (*depth*). MFCC akan dieksten / diperpanjang dengan menambah n frame sesudah dan sebelumnya pada kolom matriks. Sehingga dihasilkan data MFCC yang memiliki fitur n frame sebelum dan sesudahnya (*past-future context*). Misalkan $n = 5$ dengan banyak frame $T = 1000$, sehingga pada frame $t \geq 5$ akan ditambahkan $0 \leq t \leq 4$ dan $6 \leq t \leq 10$ pada kolom matriks frame $t = 5$, jika $t - n < 0$ atau $t + n > T - 1$ maka akan diisi dengan matriks 0 dengan ukuran yang tetap sama. (lihat Gambar 3.3)

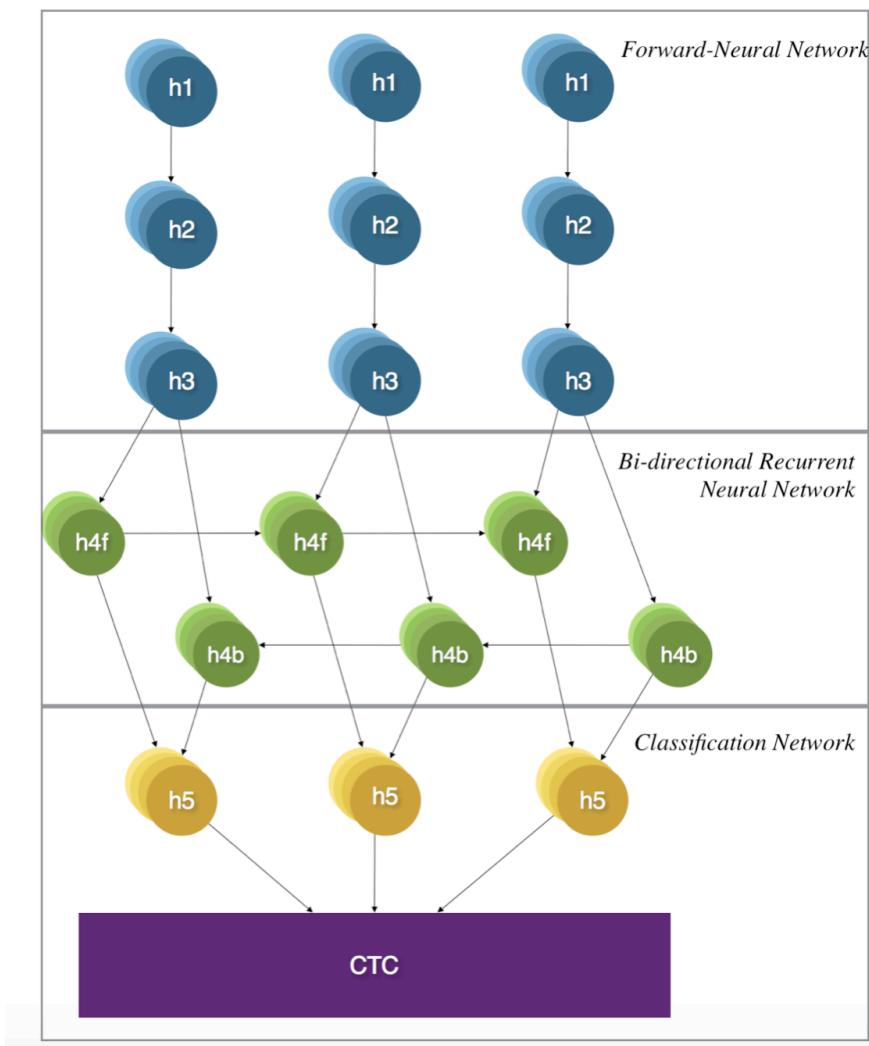


Gambar 3.3. Contoh MFCC *past-future context* dengan mengambil $n = 5$

3.2.3. Model Neural Network

Dataset $x = \{x_{t=0}, x_{t=1}, \dots x_T\}$ adalah fitur matriks yang dihasilkan oleh tahap preprocessing data. Konsep sederhana dari *neural network* adalah memasukan inputan dataset x ke dalam 3 layer, yaitu input, hidden, dan output. Dimana semua layer diberikan *weight*, *bias* dan *activation function*. Pada model *neural network* digunakan *random weight* dan *bias* mengikuti *gauss / normal distribution* dengan *activation function ReLU (Rectifier Linear Unit)* pada rumus (3.3).

Alur jaringan *neural network* pada Gambar 3.4 dibagi 3 bagian besar, yaitu *Forward-Neural Network (hidden layer)*, *Bi-directional Recurrent Neural Network (hidden layer)*, *Classification Network (output layer)*



Gambar 3.4. Alur Jaringan Model *Neural Network*

Pada layer *Forward-Neural Network*, dilakukan *fully connected neural network* dengan *activation function* ReLU. Dengan jumlah *hidden neuron* yang sama pada masing - masing *network*.

Pada layer *Bi-directional Recurrent Neural Network* digunakan LSTM untuk setiap *network forward* dan *backward*-nya. LSTM dilakukan pada saat layer ini bertujuan untuk mencegah *vanishing problem*.

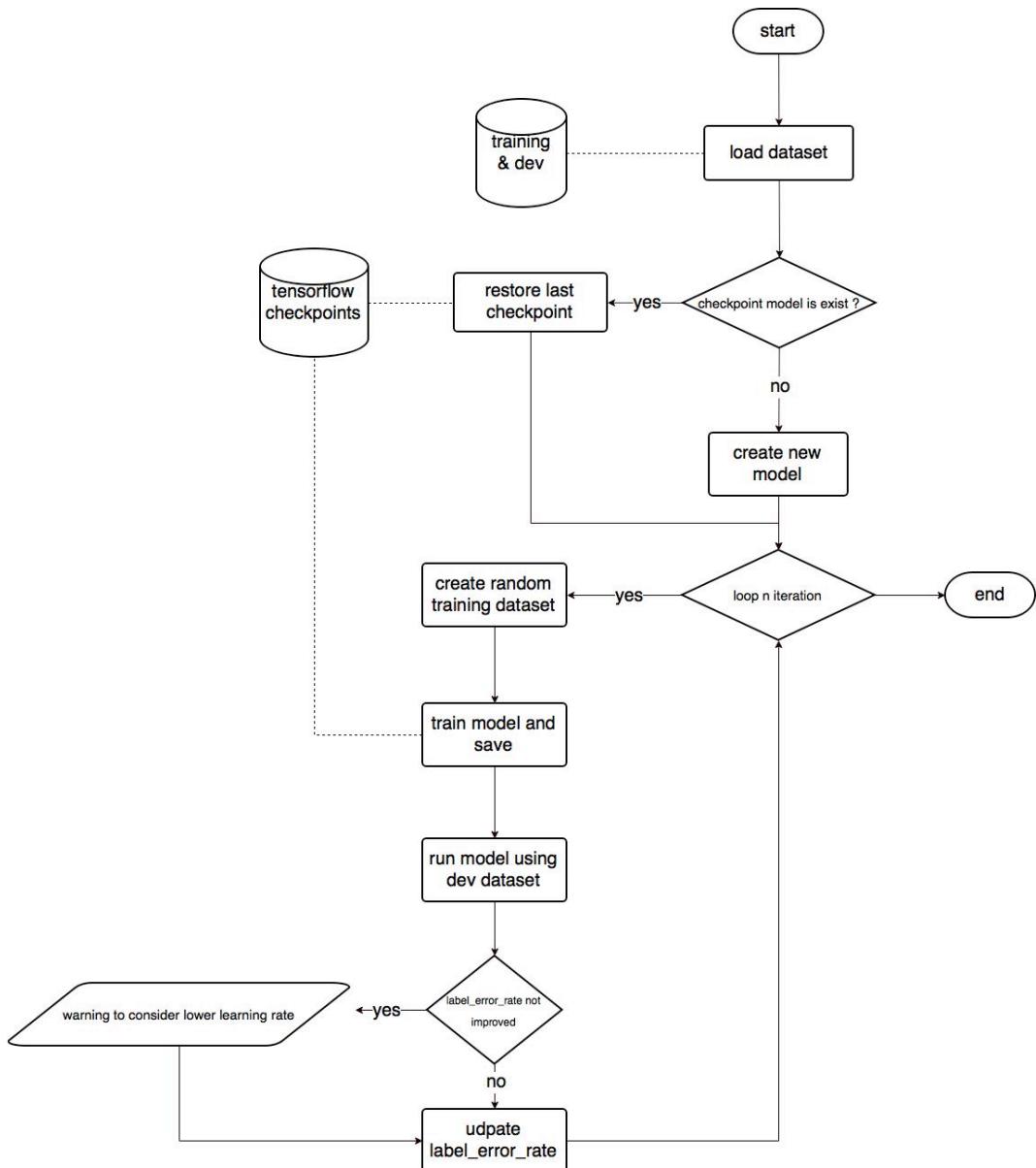
Pada layer *Classification Network*, dilakukan *Softmax function* sebagai *mapping* dari semua *network* terhadap target k karakter (grafem / fonem) yang diinginkan.

Hasil dari *Softmax function* adalah distribusi probabilitas dari *network* terhadap k karakter (grafem / fonem). Hasil ini disebut juga dengan logit. Logit ini digunakan sebagai input pada CTC untuk me-*mapping*-kan urutan karakter tanpa perlu memperhatikan segmentasi data. Setelah keseluruhan *neural network* dijalankan akan dilakukan *backpropagation* menggunakan Adam Optimizer.

3.2.4. Sistem Untuk Training dan Testing

Sistem dibagi menjadi dua bagian, sistem *training* dan sistem implementasi. Pada sistem *training*, dilakukan proses berulang dengan N iterasi untuk dilakukan backpropagation sehingga didapatkan perubahan weight dan bias yang optimum. Backpropagation yang digunakan adalah Adam Optimizer.

Untuk data training dibagi menjadi 2 bagian, *training* dan *dev* dataset. Dengan menggunakan *dev* untuk *dev* dataset untuk menghitung performa training model dengan tolak ukur *Edit Distance / Label Error Rate*. *Flowchart* / diagram alur untuk menjelaskan sistem *training* pada Gambar 3.5.



Gambar 3.5. *Flowchart / diagram alur training model*

Flowchart / diagram alur pada Gambar 3.5 dimulai dengan memberikan argumen sebagai berikut :

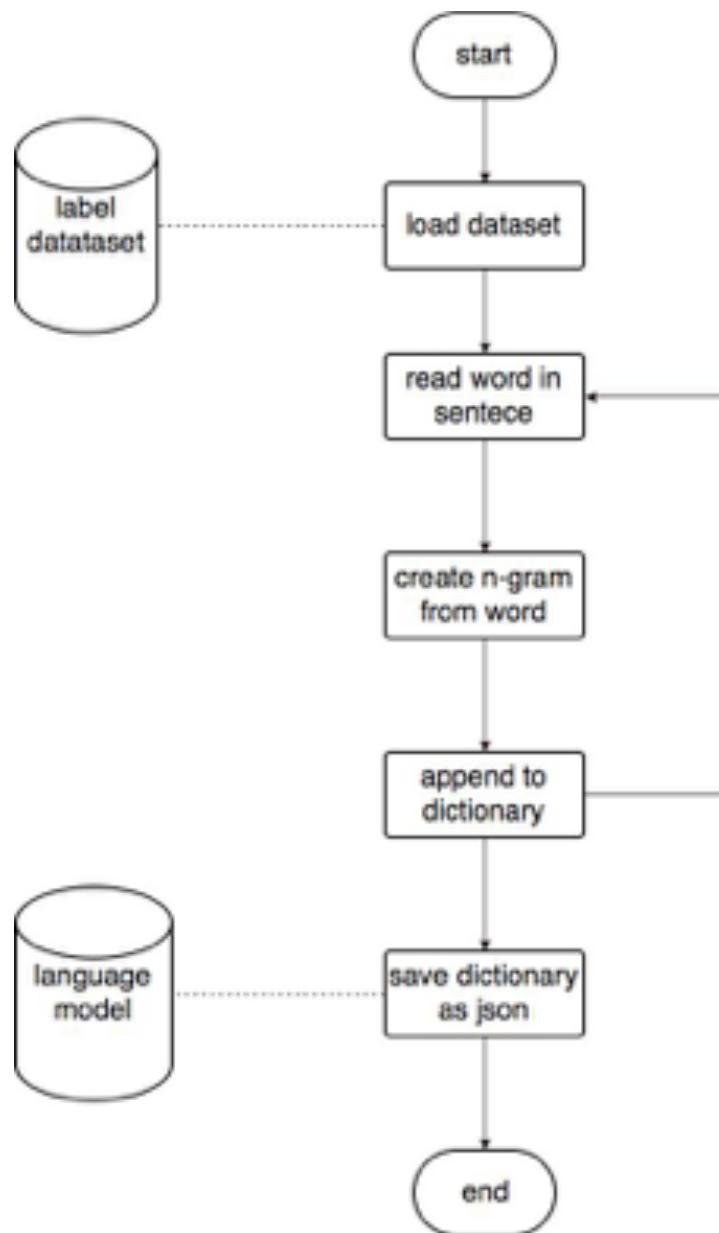
1. *Training directory*, sebagai tempat *directory / folder* berisikan *training dataset* yang dihasilkan saat preprocessing data.
2. *Dev directory*, sebagai tempat *directory / folder* berisikan *dev dataset* yang dihasilkan saat preprocessing data.
3. *Model directory*, sebagai tempat *directory / folder* yang akan dijadikan tempat penyimpanan hasil model, hasil training, dan *checkpoint* terakhir.

4. *Iteration*, jumlah iterasi setiap command dijalankan.
5. *Input Dimension*, jumlah dimensi inputan dalam kasus *context = 5* berarti $(24 * (5 + 1 + 5))$ yaitu 264.
6. *Output Character Recognition*, jumlah karakter yang ingin dikenali.

Pada sistem training ini akan dihasilkan sebuah *tensorflow model* yang berisi semua konfigurasi *weight*, *bias*, dan *graph network* yang telah dibuat saat *training*. *Tensorflow model* ini yang nanti akan digunakan sebagai model untuk menjalankan sistem implementasi di *server*.

Pada sistem implementasi, dilakukan dengan cara mengambil data suara yang direkam melalui *smartphone Android* untuk dikirim ke *server* yang sudah dihasilkan saat sistem *training*. *Server* akan mengolah data .raw audio file yang dikirim oleh *smartphone Android* dan akan menghasilkan prediksi teks untuk ditampilkan di *smartphone Android*. Dalam memprediksi teks ditambahkan satu proses lagi yang dinamakan *language model*.

Dalam proses *language model* dilakukan pengecekan / pengoreksian kata dalam kalimat yang dihasilkan oleh sistem training. Sebelum pengecekan terlebih dahulu akan dilakukan proses pembuatan model / *dictionary* untuk mempermudah proses pengecekan / pengoreksian kata. Flowchart pembuatan model / *dictionary* adalah pada Gambar 3.6.

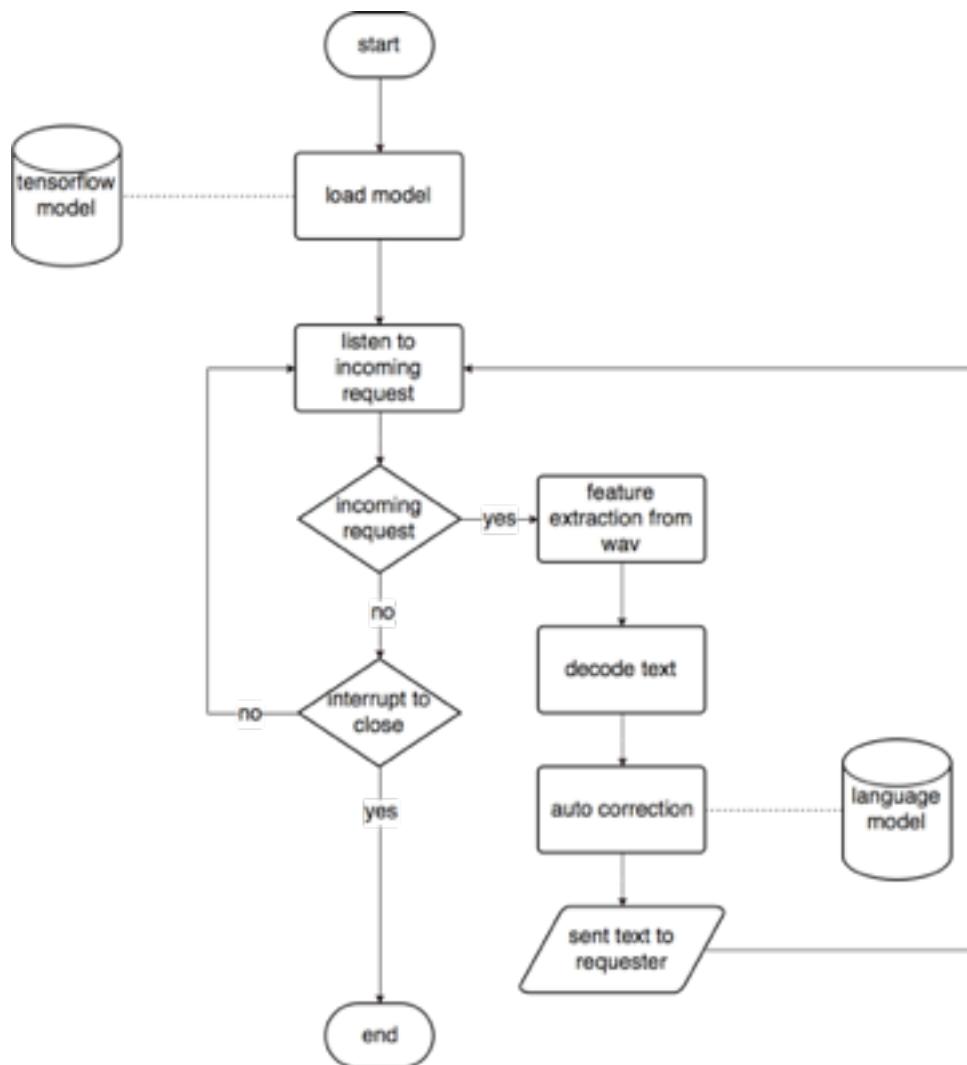


Gambar 3.6. *Flowchart / alur diagram pembuatan model / dictionary untuk language model*

Flowchart / diagram alur diatas dimulai dengan memberikan argumen sebagai berikut :

1. *Dataset Directory*, sebagai tempat *directory / folder* yang bersisikan 2 *folder dev* dan *train*.
2. *N-Gram*, jumlah n pada *n-gram*
3. *Target JSON*, *target filepath* hasil *language model* yang berupa json

Flowchart / diagram alur yang dilakukan saat sistem implementasi pada server adalah pada Gambar 3.7.



Gambar 3.7. *Flowchart / diagram alur sistem implementasi pada server*

Flowchart / diagram alur diatas dimulai dengan memberikan argumen sebagai berikut :

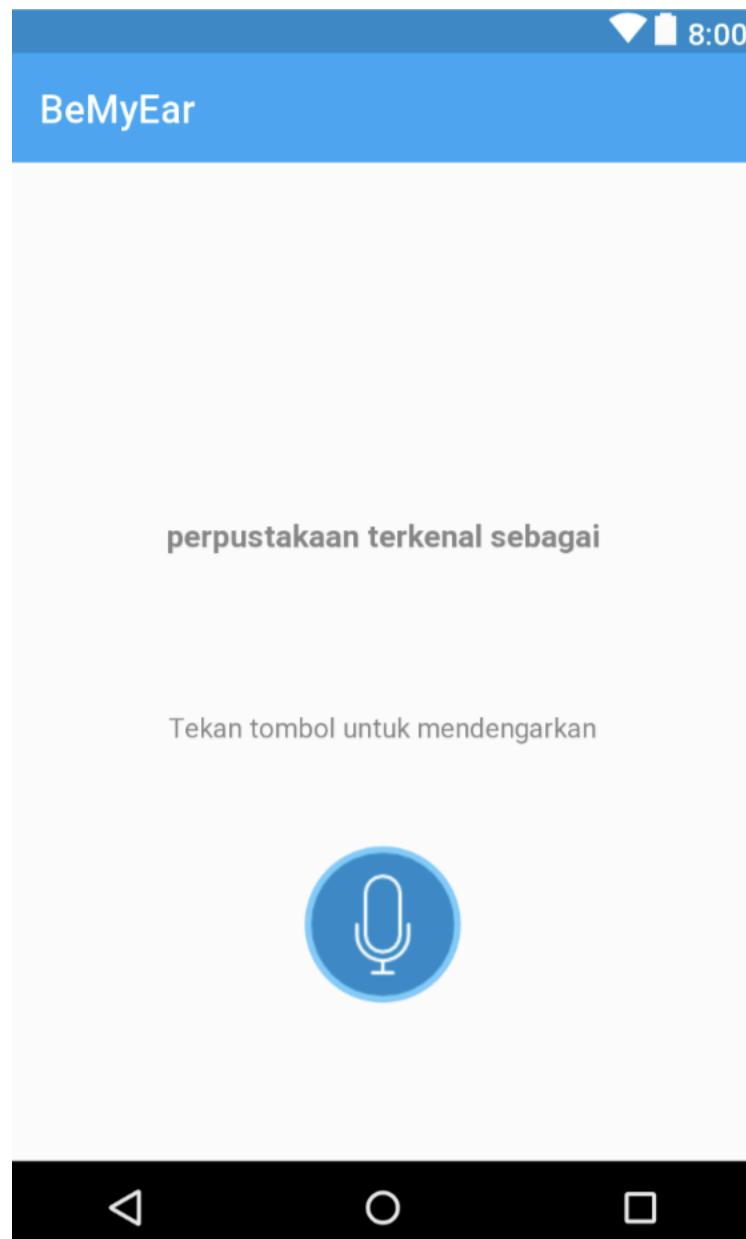
1. *Language Model* filepath, sebagai *filepath / lokasi file* untuk *language model* yang dihasilkan oleh proses pada Gambar 3.6.

3.3. Desain Sistem

Desain sistem membahas tampilan secara grafis yang disebut UI (*User Interface*) dari Aplikasi Android dan *Web Report*.

3.3.1. Desain Aplikasi Android

Pada penelitian ini dibuat desain interface untuk program Android bernama BeMyEar. Aplikasi ini akan mengimplementasikan sistem implementasi dari subbab 3.2.4 untuk system testing. Tampilan aplikasi ini akan dibuat seperti pada 0

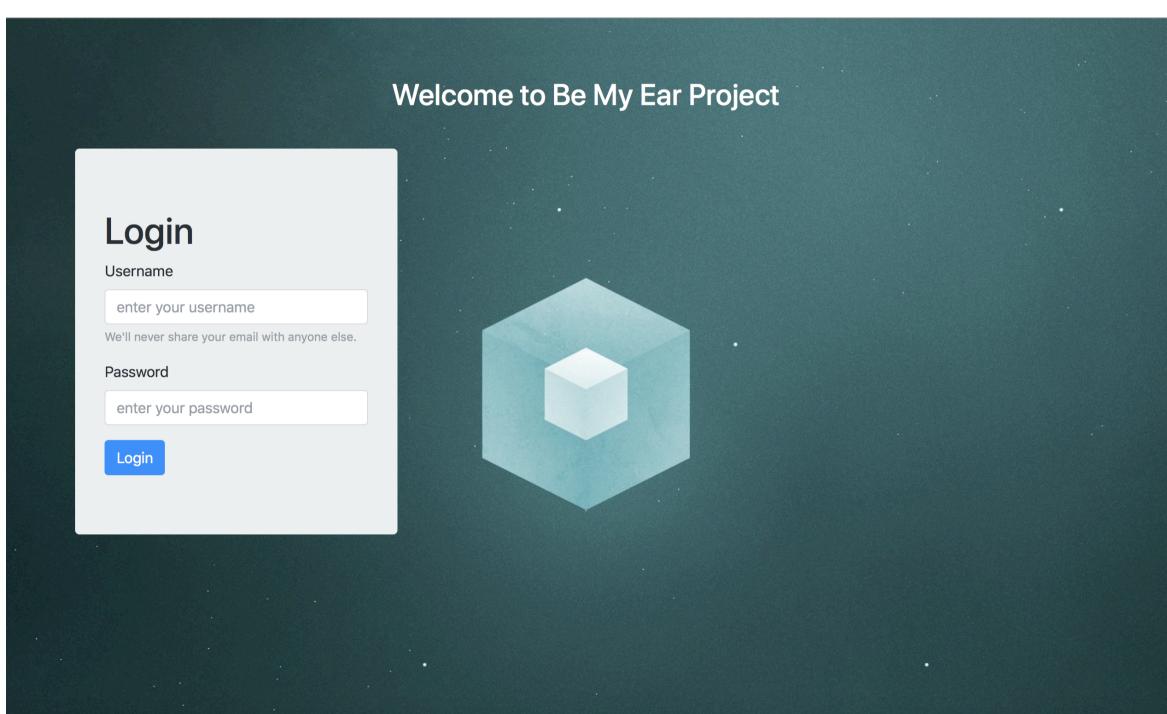


Gambar 3.8. Tampilan halaman awal jika sudah terkoneksi dengan *server*

Pada gambar 3.8, terdapat tombol merupakan tombol yang digunakan untuk merekam suara dan mengirimkan kepada server melalui socket dengan port tertentu. Jika setelah mengatakan sebuah kalimat perlu menekan tombol mic

untuk dilakukan pengenalan suara sehingga akan dihasilkan teks yang akan ditampilkan pada tengah layar **perpustakaan terkenal sebagai**

Program kedua yang dibuat dalam penelitian ini adalah BeMyEar Report, program ini berbasis *website* dan langsung terhubung dengan *server*. Program ini berisikan *dashboard* yang menampilkan daftar model, dataset, hasil training-testing, dan pebanding tiap model. Hasil *report* berupa grafik yang dimana sumbu x adalah jumlah iterasi / batch, sedangkan sumbu y adalah hasil error (CTC / CER). Sebelum masuk ke dalam program ini, diperlukan halaman *login* pada Gambar 3.9.



Gambar 3.9. halaman *login Web Report*

Setelah *login* berhasil, akan ditampilkan halaman *dashboard* dengan *navigation bar* yang dapat dipilih. Adapun *dashboard* awal menampilkan pilihan pertama pada *navigation bar* yaitu Model (lihat pada



Gambar 3.10. Halaman awal *dashboard* berisikan Model yang terdapat pada *server*

Rincian pilihan navigation bar tersebut adalah :

1. Model : berisikan semua model yang ada dalam server
2. Dataset : berisikan semua dataset beserta dengan nama dan labelnya
3. Report : berisikan semua hasil model berupa grafik
4. Compare : fitur pembanding tiap model digabungkan ke dalam satu grafik

Pada menu Dataset, ditampilkan terlebih dahulu semua tipe dataset yang dimasukkan ke dalam folder pada server. Folder ini harus berisi 2 folder lain, yaitu dev dan train. Pada tiap folder harus berisi nama yang sama dalam file .WAV dan .TXT. Tampilan awal pada menu Dataset dapat dilihat pada Gambar 3.11

Datasets

#	Name	Action
1	clean	<button>View</button>
2	clean_human	<button>View</button>
3	clean_human_phoneme	<button>View</button>
4	clean_phoneme	<button>View</button>
5	clean_synth	<button>View</button>
6	clean_synth_phoneme	<button>View</button>
7	mix	<button>View</button>
8	noise_human	<button>View</button>

Gambar 3.11. Tampilan *menu Dataset* dalam menampilkan tipe-tipe *dataset* yang berada pada *server*

Datasets - clean

Training

Total datasets: 510

#	Name	Label	Action
1	0101	bali memang merupakan destinasi favorit untuk berlibur, baik bagi wisatawan domestik maupun mancanegara.	<button>Play</button>
2	0102	ada begitu banyak objek wisata yang ditawarkan pulau eksotis ini, mulai dari panorama alam hingga tempat wisata buatan.	<button>Play</button>
3	0103	nah, berikut ini adalah daftar referensi tempat wisata di bali yang sayang untuk dilewatkan.	<button>Play</button>
4	0107	destinasi pantai di bali yang pertama adalah tanah lot.	<button>Play</button>
5	0108	tempat wisata ini terletak di desa beraban, tabanan dan hanya terpisah sekitar dua puluh empat kilometer dari pantai kuta.	<button>Play</button>
6	0109	untuk rute tercepat yang bisa menghindari macet, anda bisa berkendara melalui jalan tanah lot.	<button>Play</button>
7	0110	keunikan tempat wisata yang satu ini adalah adanya pura yang berada di atas batu karang yang menjorok ke tengah laut.	<button>Play</button>
8	01100	setelah keluar dari bandara, pergilah ke arah utara.	<button>Play</button>
9	01101	anda akan menemukan sebuah pertigaan, belok kanan menuju jalan dewi sartika.	<button>Play</button>

Gambar 3.12. Tampilan kedua setelah tombol  ditekan, pada tampilan ini ditampilkan *dataset* dengan tipe “clean”

Ketika ditekan tombol  akan ditampilkan semua data suara yang telah dimasukan ke dalam server pada folder yang dituju. Tampilan berbentuk tabel dengan kolom nama, label, dan tombol Play.

Pada menu Report, ditampilkan semua model yang terdapat pada server. Tiap model akan berisikan folder reports, dimana di dalamnya akan berisi 2 file

.CSV yaitu result_training.CSV dan result_testing.CSV. Tampilan menu Report dapat dilihat pada Gambar 3.13

Reports

All of the report will be presented by chart

#	Name	Action
3	seventh-gen-1024-attempt-9	<button>View</button>
4	seventh-gen-1024-clean-human-mfcc-5-normalized	<button>View</button>
5	seventh-gen-1024-clean-human-phoneme-mfcc-5-nromalized	<button>View</button>
6	seventh-gen-1024-clean-mfcc-5-normalized	<button>View</button>
7	seventh-gen-1024-clean-phoneme-mfcc-5-normalized using attempt 9	<button>View</button>

Gambar 3.13. Tampilan *menu Report* menunjukkan daftar semua model yang ada pada *server*.

Ketika ditekan tombol  akan ditampilkan dua buah grafik dimana grafik pertama menampilkan hasil *report* CTC (Connectionist Temporal Classification) Loss sedangkan grafik kedua menampilkan hasil *report* CER (Character Error Rate). Tampilan kedua setelah dari halaman report pada Gambar 3.13 akan dimunculkan 2 grafik yang dapat dilihat pada Gambar 3.14 dan Gambar 3.15.

Reports - seventh-gen-1024-mix-mfcc-5-normalized-mini-dataset
CTC (Connectionist Temporal Connectionist) Loss

[Download](#)

Data type
 Training Data
 Testing Data

XY-Axis

Iteration - CTC Loss



CTC (Connectionist Temporal Classification) Loss

Gambar 3.14. Tampilan grafik pada CTC Loss untuk *model seventh-gen-1024-mix-mfcc-5-normalized*

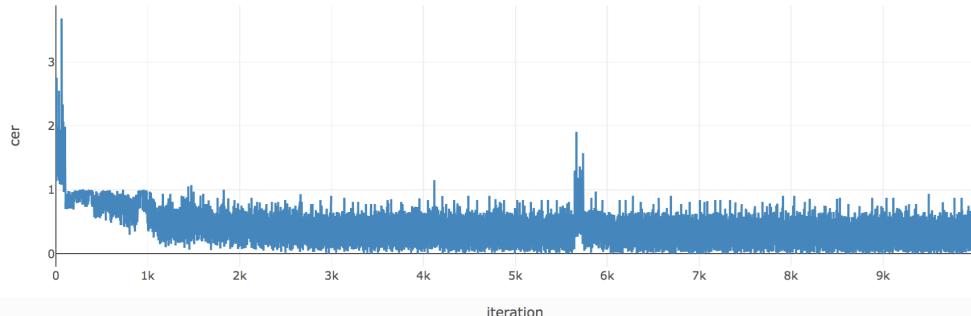
CER (Character Error Rate) Loss

[Download](#)

XY-Axis

Iteration - CER

CER (Character Error Rate)



Gambar 3.15. Tampilan grafik pada CER untuk *model seventh-gen-1024-mix-mfcc-5-normalized*.

Pada menu Compare, ditampilkan list semua model dengan tombol khusus jika ingin membandingkan 1 atau lebih model ke dalam satu grafik. Tampilan menu Compare dapat dilihat pada

#	Name	Action
3	seventh-gen-1024-attempt-9	Add
4	seventh-gen-1024-clean-human-mfcc-5-normalized	Add
5	seventh-gen-1024-clean-human-phoneme-mfcc-5-nromalized	Add
6	seventh-gen-1024-clean-mfcc-5-normalized	Add
7	seventh-gen-1024-clean-phoneme-mfcc-5-normalized using attempt 9	Add

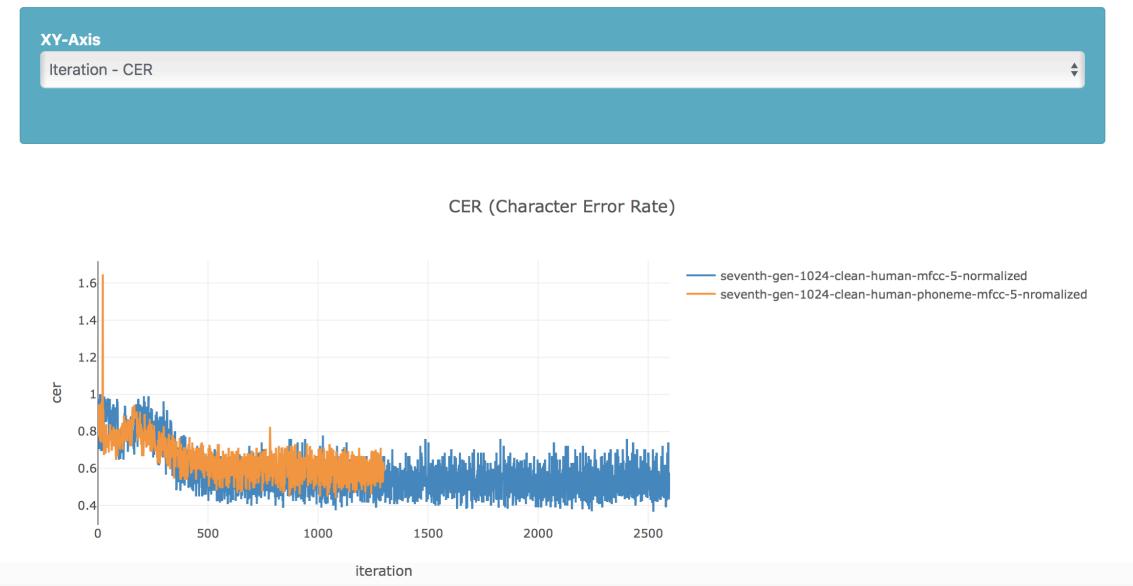
Gambar 3.16. Tampilan daftar *model* pada *server* untuk membandingkan hasil *report* dalam satu grafik.

Ketika tombol  ditekan, akan berubah menjadi  dan sistem akan mengambil data *report* untuk ditampilkan pada grafik. Ketika data *report* berhasil diambil maka akan ditampilkan pada satu grafik yang sama seperti pada



Gambar 3.17. Tampilan pembanding CTC Loss dengan dua *model* *seventh-gen-1024-clean-human-mfcc-5-normalized* dan *seventh-gen-1024-clean-human-phoneme-mfcc-5-normalized*

CER (Character Error Rate) Loss



Gambar 3.18. Tampilan pembanding CER dengan dua *model seventh-gen-1024-clean-human-mfcc-5-normalized* dan *seventh-gen-1024-clean-human-phoneme-mfcc-5-normalized*